IBM

Security for Linux on System z

Learn about the new cryptography functions in the CEX3C

Deploy security-related technologies in Linux on System z

Understand protected key cryptography

Lydia Parziale
Jonathan Barney
Vic Cross
William Johnston
Eduardo Kienetz
Eric Marins
Nilesh Patel
Sri Venkatesen

Redbooks

ibm.com/redbooks

International Technical Support Organization

Security for Linux on System z

January 2013

SG24-7728-01

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

Second Edition (January 2013)

This edition applies to Version 6, Release 2, RSU 1101 of z/VM, SUSE Linux Enterprise Server version 11 Service Pack 2 and Red Hat Enterprise Linux version 6.2.

Contents

Notices

Trademarks

Preface

No IT server platform is 100% secure and useful at the same time. If your server is installed in a secure vault, three floors underground in a double-locked room, not connected to any network and switched off, one would say it was reasonably secure, but it would be a stretch to call it useful.

This IBM® Redbooks® publication is about switching on the power to your Linux on System z® server, connecting it to the data and to the network, and letting users have access to this formidable resource space in a secure, controlled, and auditable fashion to make sure the System z server and Linux are useful to your business. This book is also about ensuring that, *before* you start designing a security *solution*, you understand what the solution has to achieve.

This book is intended for system engineers who want to customize a Linux on System z environment to meet strict security, audit, and control regulations.

Linux on System z is real Linux, based on the standard Linux specification. However, when Linux executes on the System z platform, certain characteristics of that platform are inherited by the Linux operating system images, depending on the environment present on the platform. Linux can run natively in a System z logical partition or it can run in a virtual machine under control of the z/VM® operating system. In addition, Linux runs well in the same server environment, along z/OS® and z/VSE® operating systems. These scenarios all offer additional security and administrative functions to make Linux more secure and easier to administer than a similar Linux configuration running in a distributed environment.

The IBM System z platform, with its 45 years of history, has traditionally been regarded as one of the most secure platforms in the industry. So much so, that today it is the only platform that has obtained the Common Criteria Evaluation Level 5 (EAL5) security branding. The System z10® Enterprise Class (EC[1]) and the z10 Business Class (BC[2]) have also obtained this branding. To consult the official documentation for this, see:

http://www.commoncriteriaportal.org/files/epfiles/0557b_pdf.pdf

System z196 has received the EAL5 certification for security of logical partitions.

The EAL5 ranking on the System z platform should give you confidence to run many applications on various operating systems. An example might be running an application containing confidential data on one System z10 server divided into partitions, or z/VM virtual machines that keep each application's data secure and distinct from one another. The System z architecture prevents the flow of information among logical partitions and virtual machines within a single system. System z mainframes therefore offer a very solid base on which to build a large, consolidated collection of Linux operating system instances.

The following operating systems have received the EAL 4+ with CAPP and LSPP certification: z/OS 1.8 and later, with RACF®; and z/VM 5.3, with RACF. Novell SUSE Linux Enterprise server (SLES) 9, Novell SLES 10, and Red Hat Enterprise Linux (RHEL) 4 are certified for EAL 4+ with CAPP, and RHEL 5 has an EAL 4+ rating with CAPP and LSPP certification.

Additionally, in the zEC12, PKCS #11 and Elliptic Curve Cryptography (ECC) Digital Signatures, DES, AES, SHA-512, PRNG, Tamper-resistant configurable Crypto Express4S, have been designed for PR/SM™ EAL5+ certification.

[1] Common Criteria Evaluation Assurance Level 5 (EAL5) certification for z10 EC received on 29 October 2008.
[2] Common Criteria Evaluation Assurance Level 5 (EAL5) certification for z10 BC servers received 4 May 2009.

Within zEnterprise® 114, the Elliptic Curve Cryptography (ECC) Digital Signatures, DES, AES (192 and 256), SHA-512, PRNG, Tamper-resistant configurable Crypto Express3, PR/SM have been designed for EAL5.

The base for a secure system is tightly related to the way the architecture, and, more specific, virtualization, has been implemented on System z. Since its inception 45 years ago, the architecture has been continuously developed to meet the increasing demands for a more secure and stable platform.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-business development and database management technologies. Lydia is a certified PMP and an IBM Certified IT Specialist. She has an MBA in Technology Management, and has been employed by IBM for over 25 years in various technology areas.

Jonathan Barney is an Enterprise Security Architect in the US. He has 12 years of experience in the Information Services field. He holds a degree in Computer Science from Clarkson University. His areas of expertise include Cloud Security across the Brand, Tape and Disk encryption technologies and PKI. He has written extensively on security concepts in other IBM Redbooks publications and also the thoughtsoncloud.com blog.

Vic Cross is part of the IBM Systems and Technology Group Technical Support Services team in Australia, and is based in Brisbane, Queensland. He has over 20 years of experience in general IT, 15 of which has been directly related to the System z platform and its antecedents. He holds a degree in Computer Science from Queensland University of Technology. His areas of expertise include Linux and Networking on System z. He has written and contributed to several IBM Redbooks publications, including the first edition of this book, *Linux on IBM eServer zSeries and S/390: ISP/ASP Solutions*, SG24-6299 and *Linux on IBM eServer zSeries and S/390: Porting LEAF to Linux on zSeries*, REDP-3627

William Johnston is a Senior Managing Consultant in the United States. He has 30 years of experience at IBM, focusing on mainframe security. He is an adjunct professor at Marist College in Poughkeepsie. Craig interacts with clients globally to bring security best practices to a wide range of implementations. He resides in the Mid-Hudson Valley region of New York State with his wife and three children.

Eduardo Kienetz is a Staff Software Engineer at the IBM Linux Technology Center (LTC) in Brazil. He has over 10 years of experience in the information technology field, 2.5 years of which were in the banking area (Canada), having contributed numerous hours to open source projects and presented at some Linux events. He has worked at IBM for a year, and currently leads the LTC Infrastructure team. This is his second IBM Redbooks publication on System z. His areas of expertise include Linux system's administration, integration, and software development. He holds a Computer Science degree from UNIFRA-RS, Brazil.

Eric Marins is an IT Specialist for IBM Global Technology and Services in Brazil. He has nine years of experience in configuring Linux on System z. He is currently working as an SME for the Linux team in Brazil and also as a Linux Specialist supporting more than 1800 Linux on System z servers for an IBM internal account. He holds a degree in Computer Science from Faculdade Ruy Barbosa and a post-graduate degree in Computer Information Systems

(Database Management). His areas of expertise include Linux, high availability, IP networking, security and server management.

Nilesh Patel is a Senior Identity and Access Intelligence professional in the IBM Security System, Software Group. He is a solution advisor for IBM Security and Compliance Management Solutions. He has extensive experience in design and implementation of Identity and Access Management solutions along with Security Intelligence and Compliance. Nilesh is an IBM ITSO accredited Master Author. He has published many technical papers within the IBM developer domain and customized integration modules on the IBM Open Process Automation Library for the IAM and Security Intelligence products. He has delivered many technical web casts to educate clients on new features and integration of IBM Security products.

Sri Venkatesen is an IT Specialist in the Systems and Technology Group in the US. He has two years of experience in the z/VM and Linux on System z fields. He holds a degree in Computer Science from the University of South Carolina. His areas of expertise include Linux and middleware on System z.

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

Roy P. Costa, Robert Haimowitz, Alfred Schwab (editor)
International Technical Support Organization, Poughkeepsie Center

Bill Bitner
IBM, USA

Thanks to the authors of the previous editions of this book.

- Authors of the first edition, Security for Linux on System z, published in July 2010, were:

  Lydia Parziale, Vic Cross, Shrirang Kulkarni, Guillaume Lasmayous, Nicolas Schmid, Ricardo Sousa, Karl-Erik Stenfors

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

In this chapter, we explain the environment that was used for this IBM Redbooks publication, which was hosted at the ITSO in Poughkeepsie, New York.

1.1 Hardware configuration

We used an IBM zEnterprise EC12 and an IBM System z196 for all the testing for this book. Both the IBM System zEC12 server and the IBM System z196 have four processors, 8 GB of central storage and 2 GB of expanded storage.

The System z servers that we used were at driver level 79F. The general cryptographic feature (FC 3863) covering the usability of the CP Assist for Cryptographic Function (CPACF), and the Crypto Express2 (FC 0863) feature were configured on the system.

1.2 z/VM configuration

We used z/VM v6.2 with RSU 1101 and the preventative service planning (PSP) bucket for 2827. z/VM had the IBM Performance Toolkit, RACF, RSCS, and DIRMAINT features installed and activated.

The book's working examples were developed with multiple Linux for System z distributions. Our directory entries for the Linux virtual machines had 5 GB of memory assigned.

For our Red Hat Enterprise Linux testing and examples, we used the Red Hat Enterprise Linux (RHEL) 6.2 distribution. For testing and examples using SUSE Linux Enterprise Server (SLES), we used the SLES 11 Service Pack 2 distribution. We did not use any other Linux distributions for this book.

1.3 Other software used

In the course of writing this book, the following software was also used:

- phpLDAPadmin Version 1.2.2
- IBM Operations Manager for z/VM Version 1 Release 3

1.4 Disk storage configurations

The majority of our disk storage was provided by an IBM DS8300 configured for CKD volumes. Direct access storage devices (DASD) were assigned to z/VM and the Linux systems allocated minidisks for their use; no dedicated CKD volumes were used.

For our Fibre Channel work, we used an IBM DS8300 Enterprise Storage System, configured for OpenSystems storage. We used two FCP paths out of each of our System z servers. On the z196 we used FICON® Express8 adapters and on the zEC12 we used FICON Express8S adapters. The FCP ports were attached to ports on an IBM 2005-32 FCP switch, which in turn had two connections to the DS8300 system.

2

The z/VM security management support utilities

This chapter provides information about the utilities and products that can help manage security in your z/VM environment in support of Linux workloads.

The use of an external security manager (ESM) is discussed, as well as directory management tools. We introduce the need to provide security over console access, and look at tools that can be used to help with that process.

The implementation of a certificate authority (CA) to issue certificates for use with z/VM service machines is described, as well as the implementation of secured connections to the system through the SSL server. RACF implementation to enhance built-in z/VM security features, and z/VM auditing capabilities are discussed.

2.1 The need for security management in z/VM

One might think that starting a book about security in Linux by talking about z/VM is odd. However, with z/VM as a hypervisor, Linux on System z gains some of its most valuable security and integrity features. Linux running in an LPAR on System z is attractive for certain highly demanding workloads but for flexible server consolidation exercises, particularly those involving multiple security zones, running Linux as a guest of z/VM provides the best way to gain maximum benefit from the System z investment.

Hypervisors on other platforms do not give the level of virtual machine management provided by z/VM. For this reason, discussing the capabilities of z/VM as a hypervisor, the ways that using z/VM benefits Linux, and the ways to set up z/VM to provide the best environment for operating Linux virtual machines are all important.

2.1.1 Scaling up the proof-of-concept

Some organizations want to "start small" with a Linux on z/VM installation, and this is understandable: in many cases z/VM is new to the installation and making the project too complex at the start might give a negative perception. In fact, many organizations start so small that z/VM is not even used, and initial testing with Linux on System z is done in an LPAR.

Often, in this desire to keep things simple, topics such as security management and directory management are deprioritized. This creates a possibility of the project failing because of doubts about whether the system can provide enterprise-level security management. It might seem overly ambitious, but a project that "thinks big" while still starting small is more likely to have answers to problems that may arise when the production migration begins.

The role of directory management in ensuring a secure environment must not be overlooked. Efficient and consistent maintenance of the user directory is important for keeping the system secure and maintainable.

2.2 External security management

In this section, we discuss the security capabilities found in the base function of z/VM, then introduce additional functions provided by an external security manager (ESM).

An ESM is an external software product designed to manage user identities and control access to system resources.

Note: Examples and descriptions provided in this book use the IBM Security Server Resource Access Control Facility (RACF), but many functions will also be available with a third-party ESM. There are notable exceptions to this generalization, for example, the Secure Database Manager (SDBM) backend of the z/VM LDAP server only works with RACF. Security Labels and Mandatory Access Control are capabilities provided only with RACF.

2.2.1 z/VM internal security

z/VM provides isolation and protection of virtual machines from each other, and between virtual machines and the system, overall. These functions are provided by the z/VM Control

Program (CP) and supported by features of the z/Architecture® and System z hardware. Although the core capability of security and integrity is provided by CP without an ESM, the management of this capability is quite basic.

Refer to the *z/VM Security and Integrity* document, found at:

http://www-07.ibm.com/systems/includes/content/z/security/pdf/gm130145.pdf

Passwords in the user directory

Without an ESM, passwords for users and minidisks are managed in the z/VM user directory. In the machine-readable binary directory space, the passwords are kept in an obscured format but the source file for the directory holds the passwords in clear text. z/VM administrators are the only users that can see this source file, so the exposure is limited, but an administrator can easily determine any password on the system.

Because the user directory source is only accessible by a z/VM administrator, users are unable to change their own password directly. Some kind of administration interface can be used to allow users to change a password, but this is not part of the base function.

Consistency across systems

When managing a number of z/VM systems that might be sharing disks, security of these systems must be managed consistently to ensure that the data managed in one environment is not leaked through to another system. For example, multiple z/VM systems may have user definitions that define identical DASD ranges, so that a virtual machine can be started on any available system. If the minidisk definitions are not synchronized between the systems, minidisks can possibly be accessed from an alternate system using weaker credentials.

2.2.2 Reasons to use an ESM

Most administrators start their installation journey with Linux on System z, believing that they do not need an ESM. This is often the case in a proof-of-concept scenario, but by the time they are deploying applications into production it becomes clear that an ESM is required to support the overall business environment that Linux on System z must become a part of.

Regulatory compliance

The fact that any administrator can easily determine any password on the system, and therefore obtain access to any minidisk or virtual machine on the system, may be a significant issue in many regulatory domains. Organizations that must comply with government and industry regulations on the control and management of customer and client data will usually require a level of security protection beyond what can be provided using z/VM internal security mechanisms.

Audit capability

To satisfy the requirements of a security audit, a necessary step is usually to demonstrate that data owned and managed by a system cannot be accessed by a system belonging to a different security profile. This fact can be difficult to demonstrate without an ESM, because minidisks can be attached to any virtual machine with only the minidisk password to protect it.

In addition, providing information detailing which systems successfully accessed certain data is often necessary. z/VM internal security has no simple method of providing such information. (Although deriving access information from z/VM MONITOR data might be possible, this would not be trivial to implement.)

Advanced security features

Using an ESM to protect system resources in z/VM provides for easy access to detailed configuration attributes. For example, granting VSWITCH access to a virtual machine in a persistent manner can be done by using a single RACF command. Although a single CP command can grant VSWITCH access when an ESM is not used, many more steps are necessary to make the change persistent.

Consistency across multiple z/VM systems

An ESM makes providing consistent security configuration and management much easier across many z/VM systems in an organization. The RACF database can be placed on shared DASD, allowing multiple systems to be managed according to the same rules and settings as shown in Figure 2-1. Each system maintains a separate RACF instance, but each instance refers to the same physical RACF database. Changes made to RACF on one z/VM are reflected at all the others sharing the RACF database.

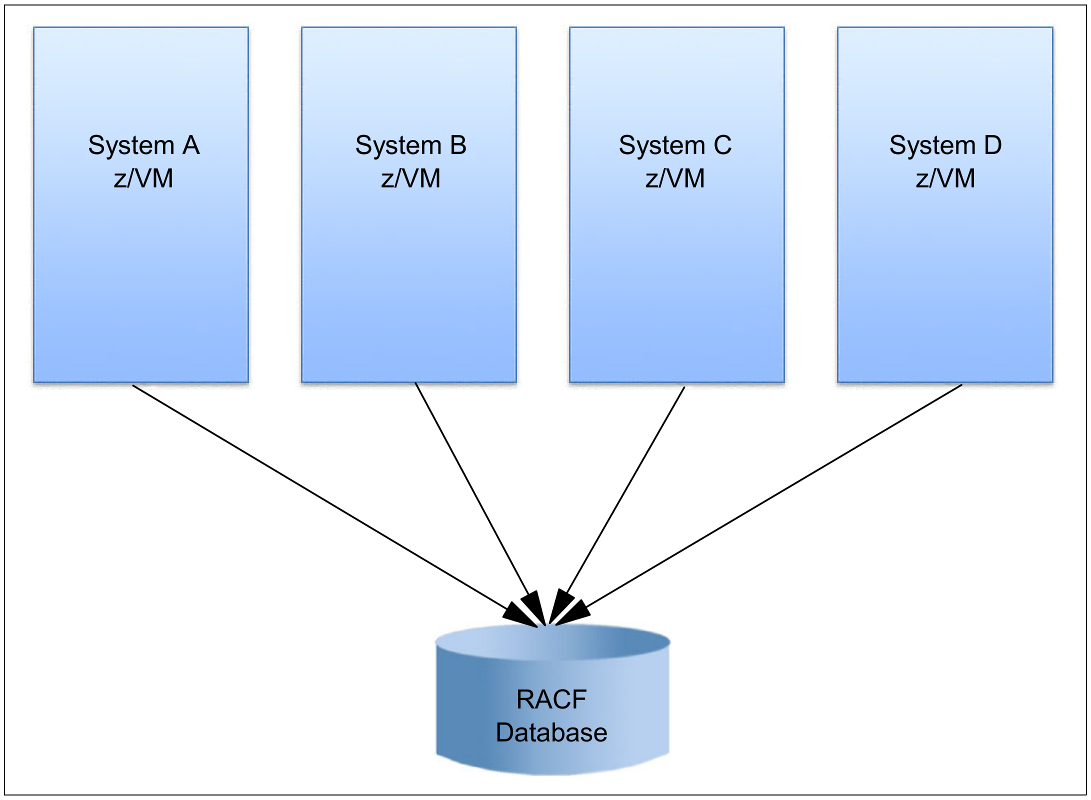


Figure 2-1 Managing multiple z/VM systems using one RACF database

Restriction: Some restrictions apply when sharing a RACF database:

- Classes that use RACLIST need to be refreshed on each z/VM system to affect the changes.
- The total number of shared z/VM systems that can share a RACF database is dependent on the performance of the DASD itself.
- Although it is not advisable, z/OS systems may also share a RACF database with a z/VM system. If this is done, those z/OS systems may not be in RACF DataSharing mode.

2.2.3 Selective enablement of an ESM

By default, the installation documentation for your ESM may enable all functions and features regardless of whether you require those capabilities. For example, the installation process for RACF enables both the VMMDISK class for minidisk protection, and the VMRDR class for unit record devices. If your installation does not require unit-record device protection, you might want to disable the VMRDR class when the installation process has completed.

The RACF installation process generates a series of RACF commands that are based on the contents of the z/VM user directory, including user and minidisk passwords. This set of commands can be modified prior to execution, allowing you to enable only the classes and security settings needed to implement your required security profile.

Caution: When modifying this command list, be careful to ensure that the system is not made too open, or that essential permission commands are not removed, which may cause features and functions to become inoperative. We recommend editing the file based on the removal of an entire unrequired class, rather than specific permissions within the class.

Refer to Chapter 8, “Best practices” on page 265 for more information about how best to enable the parts of your ESM function that you require.

2.3 User directory management

In z/VM, directory management refers to the process that is used to manage the definitions of users and their resources in the z/VM user directory.

Additional information: For more information about the user directory, refer to the z/VM manual, *z/VM: CP Planning and Administration,* SC24-6178-03.

The directory can be managed by using XEDIT and other supporting utilities. This is a manual that can be used when your system has fewer than about 50 virtual machines. As your system becomes more complex, manual directory administration can become cumbersome.

In this section we describe several features of the IBM Directory Maintenance (DirMaint™) facility, which is a priced, optional feature of z/VM. More details about usage of DirMaint are in 2.7, “z/VM Directory Maintenance Facility (DirMaint)” on page 41.

2.3.1 User management

User definitions can be easily added and changed using DirMaint. Through the use of template definitions, new users can typically be added by using a single command. One aspect of security and user management through the use of a directory maintenance system is password management.

The ability for users to set their own password, and to do so without that password being visible to system administrators, should not be underestimated as part of a system’s security profile. Note the following information about using DirMaint to set a user or guest password:

- It avoids unsecure password management practices, such as setting the password to be the same as the user name.

- It increases system integrity through nonrepudiation. Users have more accountability over their actions because they are responsible for activities performed under their identities.

When a directory management utility such as DirMaint is not in use, passwords are easily read in the source file for the directory, and users are unable to change their own passwords. When a directory maintenance utility is in place, users can be given sufficient access to the directory facility to be able to change their own passwords.

2.3.2 Disk management

One of the most useful features provided with directory management systems such as DirMaint is the automatic management of minidisk volumes for users.

Automatic overlap control

When managing the user directory manually, new minidisks must be checked to see if they overlap an existing minidisk allocation. Tools such as DISKMAP and DIRMAP can help with this, but care must be taken to ensure that directory changes are always made based on up-to-date versions of the reports that are generated by these tools.

A directory maintenance utility such as DirMaint allows for the automatic management of DASD space. The utility keeps track of the allocations on each defined volume (referred to as a *region* by DirMaint) and allocates new minidisks into a gap in the region without operator intervention.

Volume groups

DirMaint regions cannot span multiple volumes. Regions can, however, be assigned to *groups*. A group is a collection of DASD regions that are managed by DirMaint as a single area from which minidisks can be assigned. When groups are used, creating a new minidisk for a user simply requires the name of the group to be used and the size of the desired minidisk.

The default behavior of DirMaint, when using groups, is to find the first region in the group with sufficient space available and define the minidisk there. This step results in a region being filled up before DirMaint assigns minidisks on subsequent regions in the group. Alternatively, DirMaint can be configured to allocate minidisks in a round-robin fashion, where sequential minidisk requests are allocated on different regions. This process provides an automatic way to load-level across various physical DASDs, although with modern disk systems, this need is less important.

Automation integration

In z/VM Version 4 Release 3, the Systems Management API (SMAPI) was introduced. It provided a programmatic way to perform many system management tasks, such as creating new users and defining system resources. The user management functions of SMAPI require a directory manager to support their operations.

The full capability of SMAPI (when backed by a directory maintenance utility) enables the writing of other interactive interfaces to perform user and system management tasks.

Scrubbing of released disks

When disk space is released from use by a virtual machine, it is often necessary to ensure that data held on that system is removed before the disks are being reused. This process is usually manual, but directory management utilities provide options that support minidisks being automatically cleaned by the system when they are deleted by a user. For PCI-DSS compliance, as VM minidisks are returned to the DASD pool for reallocation they must be scrubbed of all data. This can be accomplished by using the CMS FORMAT command or other utilities.

Important: If your organization is subject to stringent regulatory requirements, this feature might not scrub data in a way that is compliant. When using DirMaint, refer to the z/VM Directory Maintenance Facility product documentation to learn more about this capability, and ensure that it meets your requirements before use.

2.4 Securing console access to z/VM virtual machines

Every virtual machine running under z/VM has a console device. For a Linux on System z guest, this device is the virtual equivalent of the panel and keyboard attached to a distributed server.

Some virtual machines use this console device more than others. The console on Linux virtual machines is not designed for day-to-day use and is only intended for interactive use during installation or system recovery.

2.4.1 The role of console management in securing your environment

Many installations use the Linux console through z/VM more than they need to. This poses risks to the security and availability of their environment in the following ways:

- The default Linux console driver (the support for the z/VM 3215 emulated line-mode terminal) does not suppress passwords. Any passwords or passphrases that are entered at a prompt through the 3215 console are displayed on the terminal.
- Without an effective method of managing access to the console, the ability to audit system operations becomes compromised. A shared virtual machine password used to access the z/VM virtual machine console, followed by a shared root-account password, provides console access to a Linux system with no audit trail.
- If not configured correctly, a terminal session connected to a virtual machine console can cause the virtual machine to become unresponsive or be logged off z/VM if the console connection is disrupted.
- Connecting to the console of a virtual machine prevents the console messages from being sent to the user's secondary console. If you are using secondary consoles for monitoring or automation, messages appearing at the console while a terminal is connected will not be processed by the monitoring system.

2.4.2 The z/VM LOGONBY function

z/VM provides a logon function that offers a better approach to shared console management. Known as LOGONBY, it allows access to a shared user ID (such as a Linux virtual machine) to be managed using the individual credentials of a system administrator. This improves security by removing the need for system administrators to know (or be able to find) the console password for virtual machines.

With LOGONBY, users access the system with their own credentials but with the authority of the virtual machine.

When RACF is active, VM ignores the LOGONBY directory statement and leaves the access decision to RACF. RACF determines authorization based on the SURROGAT class profiles.

When RACF has been set inactive by the SETRACF INACTIVE command, it defers the shared LOGON authorization decision to z/VM.

Configuring LOGONBY without an ESM

The user directory includes the LOGONBY keyword, which allows up to eight user IDs that can use their own password to log on to the console of the virtual machine.

To make this update using DirMaint, you would issue the following command:

```
DIRM FOR userid LOGONBY admin1 admin2 admin3
```

Where *userid* is the z/VM user name for the virtual machine whose console is being managed, and *admin1* through *admin3* are the administrators who are being given access.

Configuring LOGONBY with an ESM

The ability to log on using another user's credentials is managed by the ESM. In the case of RACF, the class SURROGAT is used to manage the required access. Using the ESM to manage the LOGONBY function removes the limit of eight user IDs allowed on the LOGONBY directory control statement.

Setting up the LOGONBY capability with RACF is achieved with the following steps:

1. Add a discrete profile for the virtual machine whose console is being managed:

   ```
   RAC RDEFINE SURROGAT LOGONBY.userid UACC(NONE)
   ```

2. Permit the relevant user IDs to the resource profile, allowing access:

   ```
   RAC PERMIT LOGONBY.userid CLASS(SURROGAT) ID(admin1) ACC(READ)
   RAC PERMIT LOGONBY.userid CLASS(SURROGAT) ID(admin2) ACC(READ)
   RAC PERMIT LOGONBY.userid CLASS(SURROGAT) ID(admin3) ACC(READ)
   ```

Note: RACF checks for a SURROGAT profile on all logon attempts, both shared and direct, To improve performance, you might want to *RACLIST* the SURROGAT class by entering the following command:

```
RAC SETROPTS RACLIST(SURROGAT)
```

You must then refresh the SURROGAT class each time you change a profile in the class:

```
RAC SETROPTS RACLIST(SURROGAT) REFRESH
```

Making surrogate login mandatory

To strengthen security even further, the ability to log on to a virtual machine's console directly *without* using an administrator's credentials can be removed. This means that the system prevents access to the console of a virtual machine using that machine's own password.

In the user directory, strengthening is done by setting the password of the virtual machine to the special value LBYONLY. Now, any attempt to log on to the virtual machine console without using LOGONBY will be rejected by CP.

When using RACF, strengthening is done by ensuring that the user ID has no permission to its own LOGONBY.userid resource. The process can be enhanced by setting the virtual machine's RACF user entry to NOPASSWORD, which prevents the possibility of the ID being revoked by someone attempting to access the console.

2.4.3 Using a console management utility

Minimizing the use of the virtual machine console, and securing access to the console, improves the security of the system environment. However, system integrity can still be affected by access to the console, as discussed in 2.4.1, “The role of console management in securing your environment” on page 9.

A better solution would be to remove the need to access the console of the virtual machines entirely. A console management facility such as IBM Operations Manager makes it possible to remove any need to access the console of a virtual machine directly, improving manageability and system integrity. Operations Manager is discussed in “IBM Operations Manager” on page 12.

Console management considerations

As introduced in 2.4, “Securing console access to z/VM virtual machines” on page 9, managing information that is generated at the Linux console is an important part of managing the overall environment. Two aspects of managing the virtual machine console are:

- Processing console message output
- Controlling interactive access to the console

Note: Console output is not directly related to security, but could form part of a security response system. For example, a log-file monitoring utility could be set up to send important messages to the console, where they can be processed by a console manager.

Managing access to the system console is very important for system integrity and auditability, however. There is an overlap here with console management, which is why we discuss several of these aspects here.

At least two facilities can help with one or both of the aspects: the z/VM Programmable Operator facility, and the IBM Operations Manager product.

z/VM Programmable Operator

The Programmable Operator (PROP) is a basic message-handling environment that is provided as part of the z/VM product. A virtual machine running a PROP script is set up as the secondary operator for virtual machines to be monitored, enabling it to receive messages sent to the consoles of the monitored users.

PROP scripts are flexible, giving the capability to issue commands back to the originating virtual machine, send messages to other virtual machines, even run commands including CP and CMS commands or REXX execs.

Example of PROP usage

One way in which PROP can provide some enhanced security and integrity is to act as a processing filter for disruptive system commands.

In this case, a PROP script runs in a user ID with a high level of system authority, and the administrator users have a low authority level and cannot issue disruptive commands (such as SHUTDOWN or VARY) on their own. Administrators get their work done by sending commands to the PROP user ID; when PROP receives the message, it invokes a script that checks the validity of the command in the message. The kinds of checks that might be done include a check of the issuing user ID, the time of day, and the scope of the command (for example, the range of device addresses in a VARY OFFLINE command). If the command meets authority, the PROP user issues the command on the administrator’s behalf and sends the results back to the administrator.

The advantages of this approach include:

- A reduction in the privilege level of individual system administrator user IDs
- Greater “sanity checking” of disruptive commands (for example, a SHUTDOWN command issued in the middle of the operating day can be blocked)
- A higher level of audit of commands that change the system state

IBM Operations Manager

The Operations Manager product is a licensed program product from IBM. It provides a number of features that ease the operation of virtual machines under z/VM, including:

- Simple access to virtual machine consoles, without logging on to the virtual machine
- Ability to start and stop virtual machines automatically, with programmed dependencies
- Message interception and automation
- Virtual machine console output logging

All of these functions can be controlled according to the configuration, so certain administrators can be given more access than others (for example: only message view capability, or access to only a subset of virtual machines). In conjunction with an ESM, even finer levels of access authority can be given to individuals or groups, and access authorities can be changed without having to restart Operations Manager.

Figure 2-2 shows an overview of the Operations Manager product.

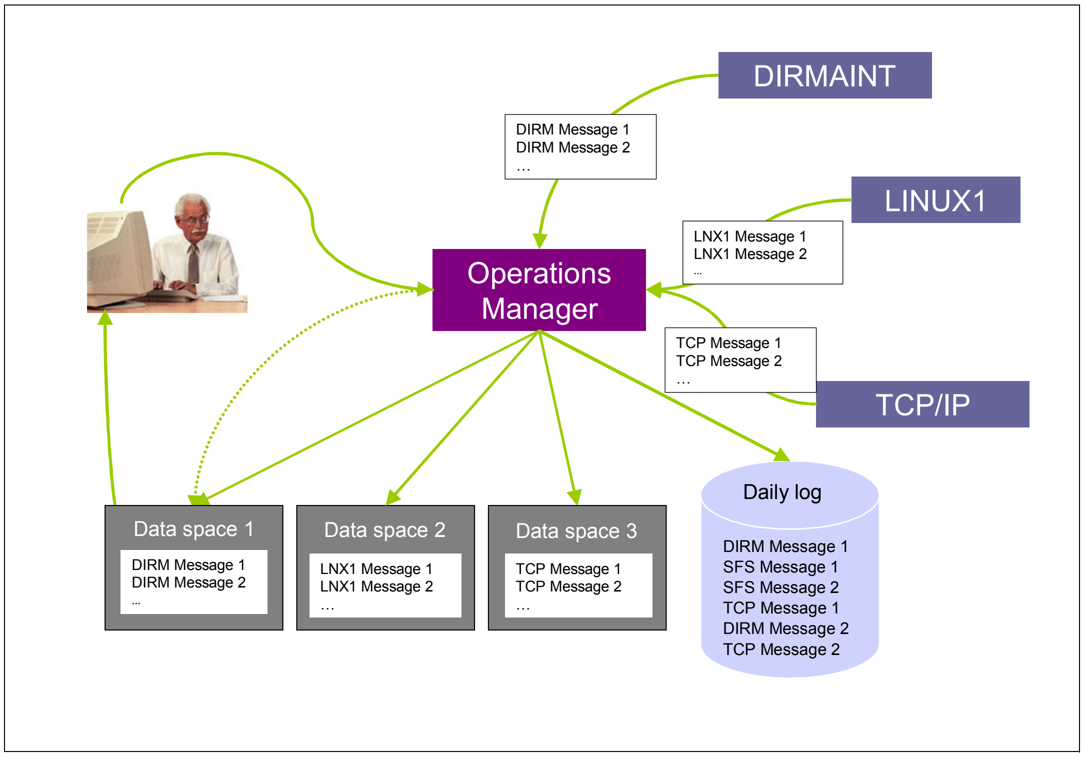


Figure 2-2 Operations Manager overview

Virtual machines in the z/VM system are configured to use Operations Manager as their secondary user. Operations Manager receives these messages and, according to the configuration, logs them. The system administrators issue Operations Manager commands through their own z/VM ID. They can view console output and spool files for virtual machines.

Access to virtual machine consoles

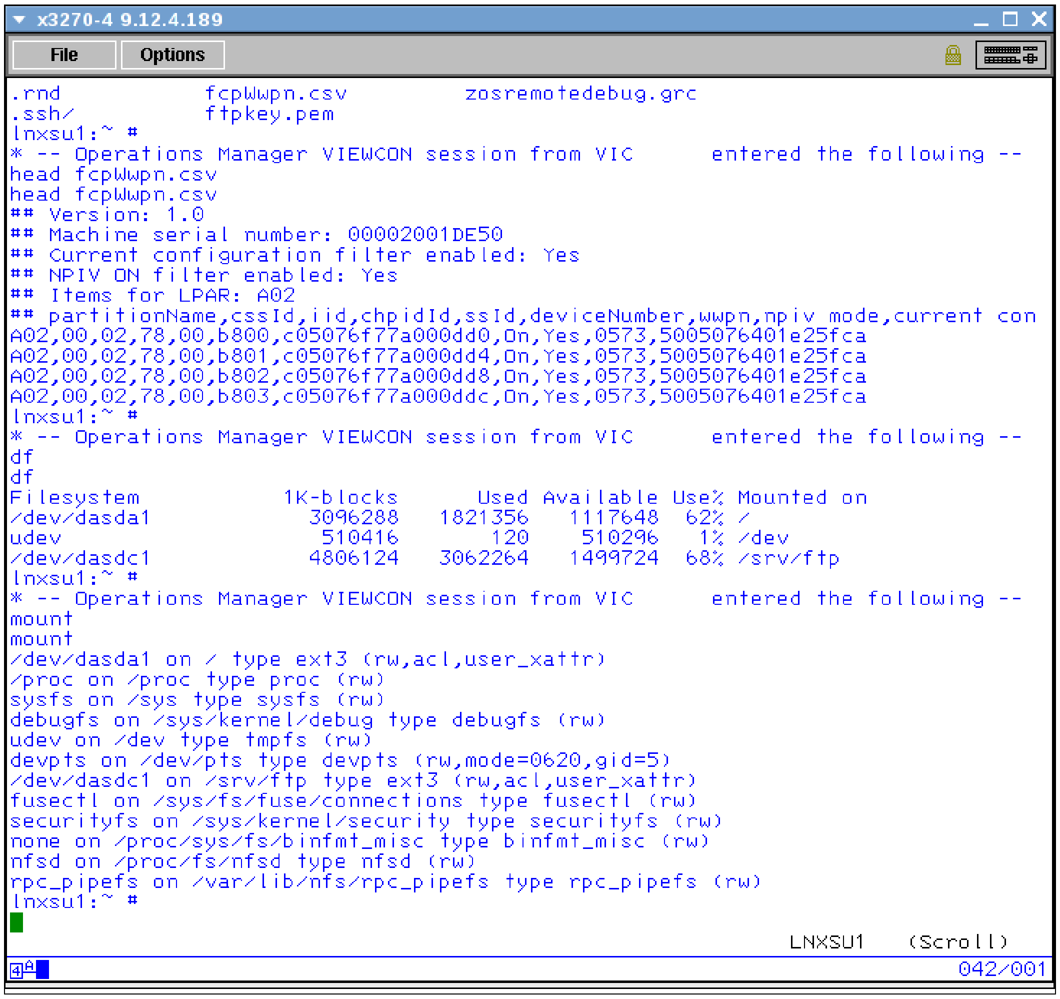
Operations Manager provides a way to access the console of virtual machines without having to log on to the virtual machine. The Operations Manager VIEWCON feature enables an administrator to perform the following functions:

- View current console output in a scrolling window.
- Scroll backward and forward through previous messages.
- Issue commands to the virtual machine, and see the response to those commands.

A VIEWCON session is invoked from an administrator user by issuing a command such as:

```
GOMCMD OPMGRM1 VIEWCON USER(LNXSU1)
```

Figure 2-3 shows the resulting VIEWCON session.



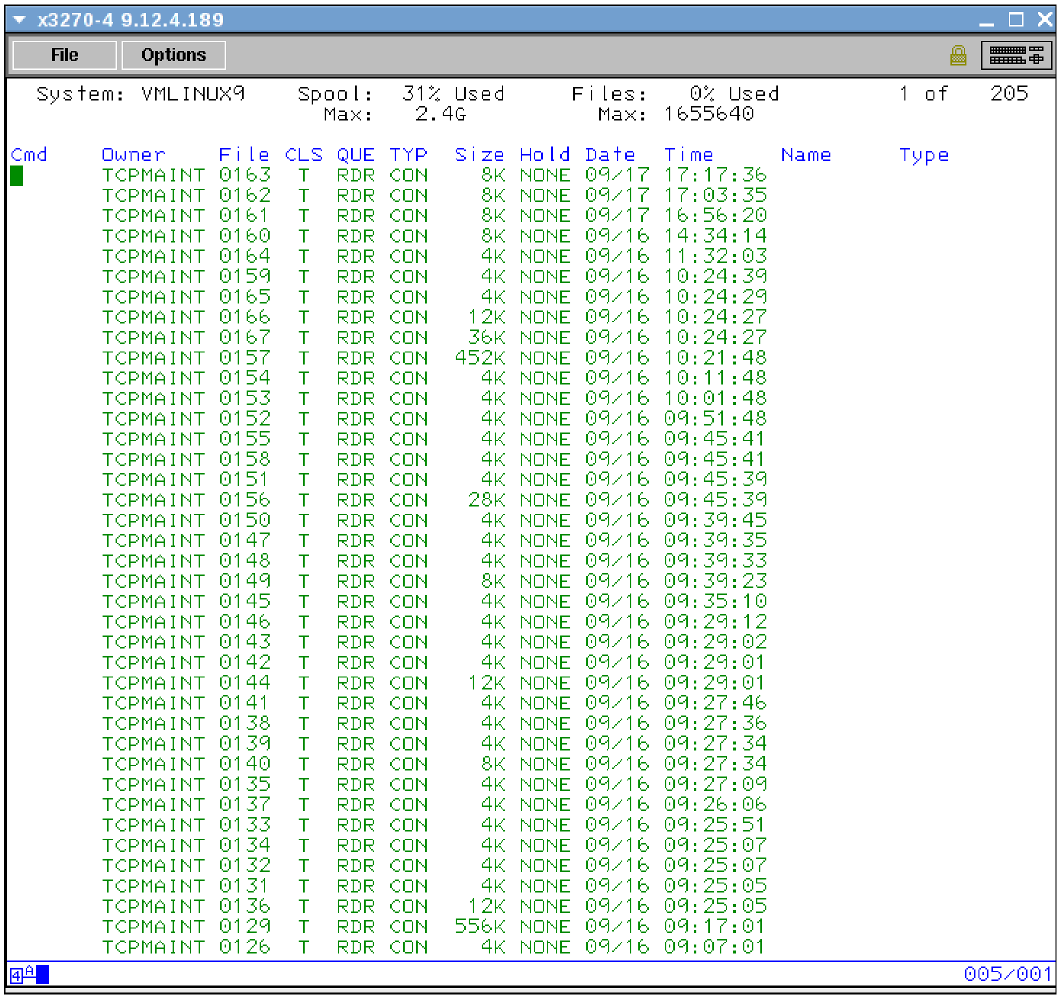
```
.rnd             fcpWwpn.csv          zosremotedebug.grc
.ssh/            ftpkey.pem
lnxsu1:~ #
* -- Operations Manager VIEWCON session from VIC      entered the following --
head fcpWwpn.csv
head fcpWwpn.csv
## Version: 1.0
## Machine serial number: 00002001DE50
## Current configuration filter enabled: Yes
## NPIV ON filter enabled: Yes
## Items for LPAR: A02
## partitionName,cssId,iid,chpidId,ssId,deviceNumber,wwpn,npiv mode,current con
A02,00,02,78,00,b800,c05076f77a000dd0,On,Yes,0573,5005076401e25fca
A02,00,02,78,00,b801,c05076f77a000dd4,On,Yes,0573,5005076401e25fca
A02,00,02,78,00,b802,c05076f77a000dd8,On,Yes,0573,5005076401e25fca
A02,00,02,78,00,b803,c05076f77a000ddc,On,Yes,0573,5005076401e25fca
lnxsu1:~ #
* -- Operations Manager VIEWCON session from VIC      entered the following --
df
df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/dasda1            3096288   1821356   1117648  62% /
udev                    510416       120    510296   1% /dev
/dev/dasdc1            4806124   3062264   1499724  68% /srv/ftp
lnxsu1:~ #
* -- Operations Manager VIEWCON session from VIC      entered the following --
mount
mount
/dev/dasda1 on / type ext3 (rw,acl,user_xattr)
/proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
udev on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/dasdc1 on /srv/ftp type ext3 (rw,acl,user_xattr)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
securityfs on /sys/kernel/security type securityfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
lnxsu1:~ #

                                                       LNXSU1     (Scroll)
                                                                   042/001
```

Figure 2-3 Operations Manager VIEWCON display

Access to virtual machine spool files

Operations Manager can be used to work with the spool files of virtual machines. Figure 2-4 shows the VIEWSPL function being used to look at spool files for the TCPMAINT user.

Note: Spool management is an often underrated and forgotten aspect of running a z/VM system. A utility such as this allows the process to be done more regularly and with greater ease.



```
   System: VMLINUX9     Spool: 31% Used      Files:  0% Used         1 of   205
                          Max:  2.4G           Max: 1655640

Cmd        Owner     File CLS QUE TYP  Size Hold Date  Time      Name      Type
           TCPMAINT 0163  T   RDR CON    8K NONE 09/17 17:17:36
           TCPMAINT 0162  T   RDR CON    8K NONE 09/17 17:03:35
           TCPMAINT 0161  T   RDR CON    8K NONE 09/17 16:56:20
           TCPMAINT 0160  T   RDR CON    8K NONE 09/16 14:34:14
           TCPMAINT 0164  T   RDR CON    4K NONE 09/16 11:32:03
           TCPMAINT 0159  T   RDR CON    4K NONE 09/16 10:24:39
           TCPMAINT 0165  T   RDR CON    4K NONE 09/16 10:24:29
           TCPMAINT 0166  T   RDR CON   12K NONE 09/16 10:24:27
           TCPMAINT 0167  T   RDR CON   36K NONE 09/16 10:24:27
           TCPMAINT 0157  T   RDR CON  452K NONE 09/16 10:21:48
           TCPMAINT 0154  T   RDR CON    4K NONE 09/16 10:11:48
           TCPMAINT 0153  T   RDR CON    4K NONE 09/16 10:01:48
           TCPMAINT 0152  T   RDR CON    4K NONE 09/16 09:51:48
           TCPMAINT 0155  T   RDR CON    4K NONE 09/16 09:45:41
           TCPMAINT 0158  T   RDR CON    4K NONE 09/16 09:45:41
           TCPMAINT 0151  T   RDR CON    4K NONE 09/16 09:45:39
           TCPMAINT 0156  T   RDR CON   28K NONE 09/16 09:45:39
           TCPMAINT 0150  T   RDR CON    4K NONE 09/16 09:39:45
           TCPMAINT 0147  T   RDR CON    4K NONE 09/16 09:39:35
           TCPMAINT 0148  T   RDR CON    4K NONE 09/16 09:39:33
           TCPMAINT 0149  T   RDR CON    8K NONE 09/16 09:39:23
           TCPMAINT 0145  T   RDR CON    4K NONE 09/16 09:35:10
           TCPMAINT 0146  T   RDR CON    4K NONE 09/16 09:29:12
           TCPMAINT 0143  T   RDR CON    4K NONE 09/16 09:29:02
           TCPMAINT 0142  T   RDR CON    4K NONE 09/16 09:29:01
           TCPMAINT 0144  T   RDR CON   12K NONE 09/16 09:29:01
           TCPMAINT 0141  T   RDR CON    4K NONE 09/16 09:27:46
           TCPMAINT 0138  T   RDR CON    4K NONE 09/16 09:27:36
           TCPMAINT 0139  T   RDR CON    4K NONE 09/16 09:27:34
           TCPMAINT 0140  T   RDR CON    8K NONE 09/16 09:27:34
           TCPMAINT 0135  T   RDR CON    4K NONE 09/16 09:27:09
           TCPMAINT 0137  T   RDR CON    4K NONE 09/16 09:26:06
           TCPMAINT 0133  T   RDR CON    4K NONE 09/16 09:25:51
           TCPMAINT 0134  T   RDR CON    4K NONE 09/16 09:25:07
           TCPMAINT 0132  T   RDR CON    4K NONE 09/16 09:25:07
           TCPMAINT 0131  T   RDR CON    4K NONE 09/16 09:25:05
           TCPMAINT 0136  T   RDR CON   12K NONE 09/16 09:25:05
           TCPMAINT 0129  T   RDR CON  556K NONE 09/16 09:17:01
           TCPMAINT 0126  T   RDR CON    4K NONE 09/16 09:07:01
                                                                         005/001
```

Figure 2-4 Operations Manager VIEWSPL list

2.5 Securing network access to z/VM

Transport Layer Security (TLS), and its predecessor Secure Sockets Layer (SSL), are cryptographic protocols that provide end-to-end encrypted communications over unsecure networks, at the transport layer.

In this section, we describe the configuration details for establishing secure connections to a z/VM system. Digital certificates and trust hierarchies are discussed in order to gain an understanding of how they are used to convey trust across unsecured networks.

2.5.1 X.509v3 digital certificates and trust hierarchies

In this section we discuss the X509v3 certificates used by SSL to secure network access to z/VM. The contents of the certificates are out of scope for this document, but their creation and uses are explored here.

X.509v3 digital certificates

X.509v3 digital certificates are used to represent an entity within a system. These entities may be users, servers, or a certificate authority (CA). Within the certificate is information to uniquely identify the entity, a public and private keypair used for encrypting and signing messages, and a signature of the CA that vouches for the authenticity of the certificate.

Digital certificates are generated by either the CA or the using entity. Certificates generated by the CA are sent to the user already signed by the CA. These certificates convey the level of trust that the signing CA bears.

If the using entity, such as a Telnet server or client user, creates the certificate, that certificate is considered *self-signed*. Self-signed certificates convey no trust since their contents have not been vetted by a CA. Users may generate a certificate signing request (CSR) to be sent to the CA. The CA processes the CSR and returns a signed certificate that replaces the certificate in the user's certificate repository.

Trust hierarchies

X.509 digital certificates convey the trust of the certificate signer. A CA digitally signs a certificate by copying a portion of the certificate and encrypting it with its *private* key. The verifier of the signature has a copy of the CA's certificate containing only the *public* key. Since the public and private keys are mathematically related and part of a key pair, what is encrypted with one key can be decrypted with the other. The verifier decrypts the signed portion of the certificate with the CA's public key and compares it to the data that was originally encrypted. If the values are identical, then the verifier can be sure that the signer, the certificate, and the certificate owner are all authentic.

There may be multiple CAs in the chain of certificate authorities. Each CA has its certificate signed by a higher level CA. The top-level CA is trusted by all entities within the hierarchy; it uses a self-signed certificate as its CA certificate. A lower level CA is used to issue server and client certificates as shown in Figure 2-5 on page 16.

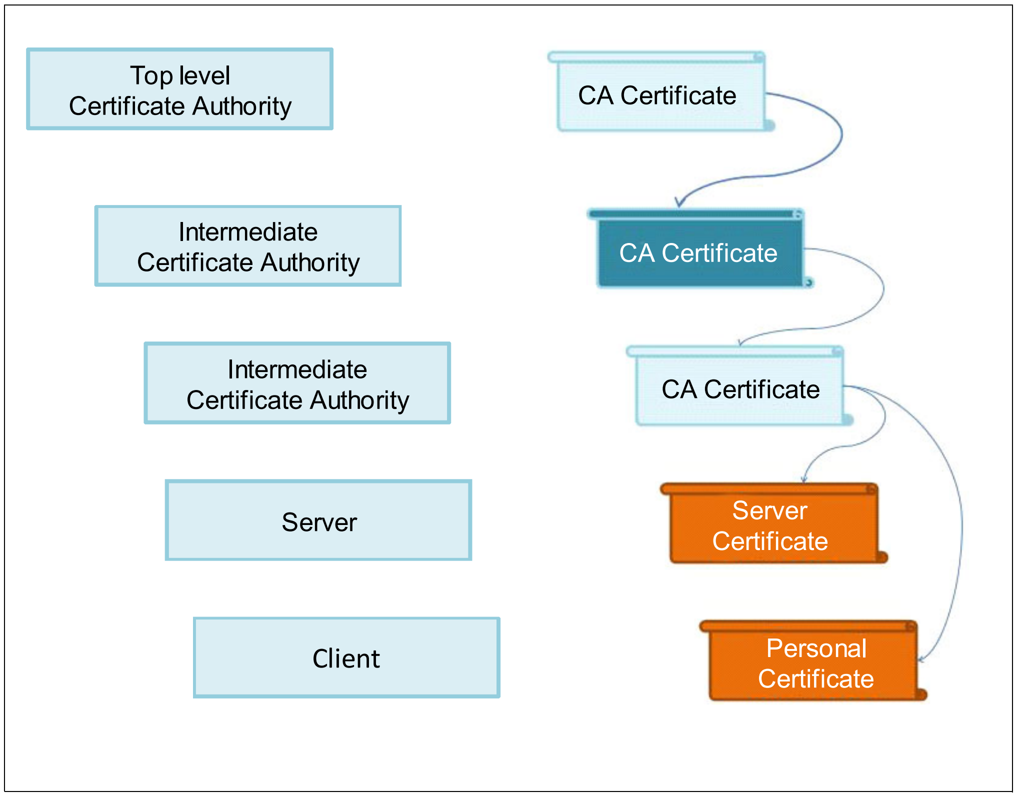


Figure 2-5 Trust hierarchy

A server certificate is used by the TCP/IP-based server to authenticate itself to the client that is attempting to connect to it. The client will have a copy of the CA certificate in order to verify the authenticity of the TCP/IP server. Additionally, if client-side authentication is required, the server verifies the client's certificate in the same manner.

2.5.2 z/VM Telnet server

By default, Telnet 3270 session data flows unencrypted over the network. A machine that is located between the client and the host can then intercept and dump all communications to get the user IDs and passwords. This is a "man in the middle" attack.

To protect the communication between host and clients (IBM Personal Communications, or x3270, for instance), z/VM provides an SSL-capable virtual machine, called SSLSERV. Starting with release V5R4.0, this virtual machine is a pure CMS service machine that provides encrypted communications to clients connecting to the z/VM partition. SSL server code is preinstalled as part of a standard z/VM installation, and simply needs to be configured. For the SSL server to operate properly, it is required to have access to a certificate authority (such as Thawte or Symantec Powered by VeriSign) to be able to provide certificates.

Note: An updated CMS SSL Server is provided in z/VM 6.2, and is available in the service stream for z/VM 5.4 and 6.1. The updated SSL server provides performance improvements over the original CMS SSL Server introduced in z/VM 5.4. More information about the new SSL server can be found at:

`http://www.vm.ibm.com/related/tcpip/tcsslspe.html`

The original and updated SSL servers use the same process for managing the certificate database, so the following processes apply with either.

SSL certificate generation and management

One of the critical aspects in an SSL-enabled server implementation is the generation and management of keys and certificates. The entity that does generation and management of digital certificates is called a certificate authority (CA).

A certificate signed by a CA conveys the trust of that CA to the user community consuming that certificate. Commonly used CAs include Verisign, Entrust, and Thawte. These entities are used to digitally sign a certificate after verifying its contents. Once signed, the certificate conveys the trust of the signer, the CA, to the user of the certificate. In this way, the certificate may be used to authenticate a user ID without the use of a less secure password construct.

When running on System z, clients may choose to implement their own CA to run on:

- z/OS
- z/VM
- Linux on System z

For the purposes of this demonstration, we created a set of certificates in order to mimic the trust hierarchy of a fully functional certificate authority. Using **`gskkeyman`**, we created a CA certificate which was then used to sign the server certificate and the client's personal certificate.

In most cases, a well-known certificate authority such as Verisign or Entrust will be used to sign certificates shared outside an installation boundary. For example, data shared with a business partner may be protected using certificates signed by Verisign. Both you and the partner trust Verisign and therefore trust each other's certificates.

For communications within your installation, it is possible to become your own certificate authority. By implementing PKI Services on z/OS, or another true certificate authority, you can create and sign certificates that convey enough trust for your internal applications.

Note: **`gskkyman`** is the tool used for key and certificate management in z/VM, and that can be seen as the command-line version of the IKEYMAN graphical tool found in other IBM products. **`gskkeyman`** is not a certificate authority itself.

Implementing a certification authority in z/VM

Going into all the details of the implementation and operations of a CA is far beyond the objectives of this book. Nevertheless, we briefly describe the steps that we followed to have a CA certificate sign the server and client certificates for the processes running under z/VM. For details about implementing a CA, refer to:

- *z/VM: TCP/IP Planning and Customization* (for V5R4.0), SC24-6125
- Chapter 15 "SSL Certificate/Key Management" in *z/VM: TCP/IP LDAP Administration Guide* (for V5R4.0), SC24-6140

Create the CA certificate and key

Each CA requires a CA certificate. This certificate contains keys that are used to sign the lower order certificates in the trust hierarchy. To create the CA certificate and its keys:

1. Log on to the GSKADMIN virtual machine.
2. Issue the `gskkyman` command.
3. In the Database Menu, select **1- Create New Database** to create the certificate authority key database.
4. Enter the key database name, for example, `CA.kdb`.

   Note: All files created by `gskkyman` are stored in a Byte File System mounted in `/etc/gskadm`.

5. Enter the password to protect the access to this database.
6. Customize the password expiration period, and database record length if necessary.

   The database is then created, as stated by the following message:

   ```
   Key database /etc/gskadm/SA created.
   ```

7. Create the CA key and certificate, as a self-signed certificate. From the Key Management Menu, select **6** - **Create a self-signed certificate**.
8. Specify the certificate type to be created from one of the user certificate options. The type of user certificate created depends on the security requirements of your installation.
9. Select the type of digest for the signature.
10. Specify the label, subject name, and length of time the certificate is valid as requested.
11. After the certificate is created, set the certificate as the default, as follows:
    a. Select **1 - Manage keys and certificates**.
    b. Select the certificate you created.
    c. Select **3 - Set key as default**.

The CA is now configured, and the certificate and private key are stored in the password-protected database.

Exporting CA certificate as a file

In order for consumers of the certificates signed by the CA certificate to properly verify the trust hierarchy, they need a copy of the CA certificate. Clients needing to authenticate the identity of a server need the CA certificate of the signer of the server's certificate. CA certificates are exported by the CA and made readily available. For the most popular CAs, RACF comes with the CA certificates as part of its installation.

For our `gskkeyman` implementation, we exported the CA certificate to a file to be imported into the server and client keystores. To export the CA certificate as a file:

1. Log on as GSKADMIN user.
2. Issue the `gskkyman` command.
3. Open the CA database (option **2**).
4. From the Key Management Menu, select **1**- **Manage Keys and Certificates.**
5. Select your CA certificate in the Key and Certificate List.
6. In the Key and Certificate Menu, select option **6**, to export the CA certificate as a file.

7. Select the required export format.

Note: For use with IBM Personal Communications, a certificate must be exported as Base64 ASN.1 DER format (option 2).

8. Set the file name. The certificate will be saved in the BFS file pool, by default in the `/etc/gskadm` location.
9. Send the file to the GSKADMIN A disk, as shown in Example 2-1.

Example 2-1 Retrieve the CA certificate onto the GSKADMIN A disk

```
Ready; T=0.01/0.01 17:54:04
openvm list
Directory = '/etc/gskadm'
Update-Dt  Update-Tm Type  Links          Bytes Path name component
09/14/2009 16:26:07   F        1            646 'certreq.arm'
09/14/2009 16:26:53   F        1           1468 'lasmayous.arm'
09/14/2009 16:15:59   F        1           2021 'Cacert.base64.der'
09/14/2009 14:56:36   F        1          70080 'CA.kdb'
09/14/2009 14:38:08   F        1             80 'CA.rdb'
09/14/2009 14:35:54   F        1            129 'CA.sth'
09/14/2009 15:06:42   F        1          75080 'Database.kdb'
09/14/2009 15:06:42   F        1             80 'Database.rdb'
09/14/2009 14:58:45   F        1            129 'Database.sth'
Ready; T=0.01/0.01 17:54:07
openvm get /etc/gskadm/Cacert.base64.der CACERT DER A(BFSLINE NL
Ready; T=0.01/0.01 17:55:10
RUNNING   VMLINUX9
```

Note: Do not forget the option BFSLINE NL, to preserve the text nature of the file.

The certificate is now ready to be retrieved and imported to a third-party client.

Create a certificate request for the SSL server

In most cases, in order to protect the private key of the keypair managed by the certificate, the owner of the certificate (the server or client) will create the certificate locally, then create a Certificate Signing Request (CSR). The CSR is the certificate, formatted for the CA to sign, but does not contain the private key of the keypair. The CSR is sent to the CA. The CA fulfills the request and returns a signed certificate. The user then imports the signed certificate back into its keystore. The import action re-pairs the public key and certificate with the private key in the keystore. This way the private key never left the custody of its owner and the trust of the CA is also maintained.

To create a certificate request for the SSL server:

1. Create a key database by following steps 1 on page 18 to 6 on page 18 in “Create the CA certificate and key” on page 18.

Note: By default, the SSL server is looking for a key database named `/etc/gskadm/Database.kdb`.

2. Create the password stash file, by using option **10 - Store database password**.
3. Instead of creating a self-signed certificate, from the “Key Management Menu”, select option 4 (Create a new certificate request).

4. Select the type of certificate you want to request.
5. Specify the file name of the certificate request, for instance `certreq.arm`.
6. Specify the characteristics of the requested certificate.

> **Note:** Although not enforced by **`gskkyman`**, the following requirement must be followed for an SSL server certificate: the label *must* be no more than eight uppercase alphanumeric characters.

7. The certificate request is saved in `/etc/gskadm` under the name specified.
8. Using **`gskkyman`**, as shown in Example 2-2, sign the certificate request.

Example 2-2 Certificate request signature

```
gskkyman -g -x 365 -cr certreq.arm -ct VMLINUX9.arm -k CA.kdb
Enter database password (press ENTER to cancel):


Certificate created.
Ready; T=0.44/0.44 18:15:04
                                                        RUNNING   VMLINUX9
```

> **Note:** The certificate is signed using the default key in the CA database. If you want to sign the certificate with another key, use the `-l` *`label`* option to specify which label of the key to use for the signature.

Import the certificate in the z/VM key database

Import the certificate to the z/VM key database. For the import to work correctly, the CA certificate must be in the key database.

To import the certificate:

1. Log on to GSKADMIN.
2. Issue the **`gskkyman`** command.
3. Open the `Database.kdb` database.
4. Select option **7** - **Import a certificate**.
5. Enter the name of the CA signed certificate created in “Create the CA certificate and key” on page 18, and the label to use for this certificate.
6. To import the signed certificate request, select **5** - **Receive requested certificate or a renewal certificate**.
7. Select the file that contained your signed certificate.

The SSL server certificate is ready for use; it only has to be added to the configuration file.

SSL server configuration

The z/VM SSL server requires read access to the key database, as well as to the password stash file. By default, only the user gskadmin has access to those files. Therefore, permissions need to be changed to allow security group read access, as detailed in Example 2-3 on page 21.

Example 2-3 Using the openvm permit command to change file permissions

```
openvm permit /etc/gskadm/Database.kdb rw- r-- ---
Ready; T=0.01/0.01 12:00:28
openvm permit /etc/gskadm/Database.sth rw- r-- ---
Ready; T=0.01/0.01 12:00:36
openvm list (own
Directory = '/etc/gskadm'
User ID    Group Name  Permissions Type  Path name component
gskadmin   security    rw- --- ---  F    'CA.kdb'
gskadmin   security    rw- --- ---  F    'CA.rdb'
gskadmin   security    rw- --- ---  F    'CA.sth'
gskadmin   security    rw- r-- ---  F    'Database.kdb'
gskadmin   security    rw- --- ---  F    'Database.rdb'
gskadmin   security    rw- r-- ---  F    'Database.sth'
Ready; T=0.01/0.01 12:00:42
RUNNING   VMLINUX9
```

The last part of the configuration resides in the TCP/IP stack, which must be instructed to use SSL instead of plain-text telnet3270. Example 2-4 shows parts of the TCP/IP configuration file `PROFILE TCPIP D` relevant to SSL server configuration.

Example 2-4 Configuration file PROFILE TCPIP D

```
PROFILE  TCPIP    D1  V 80  Trunc=80 Size=78 Line=17 Col=1 Alt=0
===== * * * Top of File * * *
===== ; ---------------------------------------------------------------------
===== ; - PROFILE TCPIP created by DTCIPWIZ EXEC on 29 Aug 2008
===== ; - Configuration program run by MAINT at 10:22:34
===== ; ---------------------------------------------------------------------
===== ;   %%File Origin Indicator - DO NOT REMOVE OR ALTER the next line%%
===== ;   %%TCPIP%%PROFILE%%STCPIP%%
===== ; ---------------------------------------------------------------------
===== ASSORTEDPARMS
===== PROXYARP
===== ENDASSORTEDPARMS
===== ; ---------------------------------------------------------------------
===== OBEY
===== OPERATOR TCPMAINT MAINT MPROUTE DHCPD REXECD SNMPD SNMPQE LDAPSRV
===== ENDOBEY
===== ; ---------------------------------------------------------------------
===== PORT
=====   20   TCP FTPSERVE  NOAUTOLOG ; FTP SERVER
=====   21   TCP FTPSERVE            ; FTP SERVER
[...]
=====   845  TCP VSMSERVE            ; VSM API Server
=====   962  TCP INTCLIEN SECURE 2NDLINX9
===== ; 2049 UDP VMNFS               ; NFS Server
===== ; 2049 TCP VMNFS     NOAUTOLOG ; NFS Server
===== ; 9999 TCP SSLSERV             ; SSL SERVER - ADMINISTRATION
[...]
===== ; ---------------------------------------------------------------------
===== AUTOLOG
===== FTPSERVE   0
===== SSLSERV    0
===== ENDAUTOLOG
```

```
===== SSLSERVERID SSLSERV TIMEOUT 60
===== INTERNALCLIENTPARMS
=====  SECURECONNECTION REQUIRED
=====  TLSLABEL 2NDLINX9
=====  PORT 23
=====  PORT 962
===== ENDINTERNALCLIENTPARMS
PROFILE  TCPIP    D1  V 80  Trunc=80 Size=82 Line=79 Col=1 Alt=8
====>
                                                      X E D I T  1 File
```

Because of the `SECURECONNECTION REQUIRED` statement, no unencrypted Telnet 3270 sessions are allowed. Standard connections on port 23 are still possible for clients capable of issuing the STARTTLS Telnet command. Other clients can connect to port 962, starting an SSL connection on this port before issuing Telnet 3270 commands.

Note: We did our tests in a second-level z/VM, which was running as a guest of the z/VM that was hosting our residency. This approach is a very convenient way to do experiments before moving the configuration to the production system.

When everything has been tested successfully, the TCP/IP stack must be restarted.

Note: Before restarting TCP/IP, make sure to have a working connection to the TCPMAINT virtual machine, either through VM/Pass-Through Facility (PVM) or the Hardware Management Console (HMC), for instance.

Example 2-5 shows the SSL-related messages:

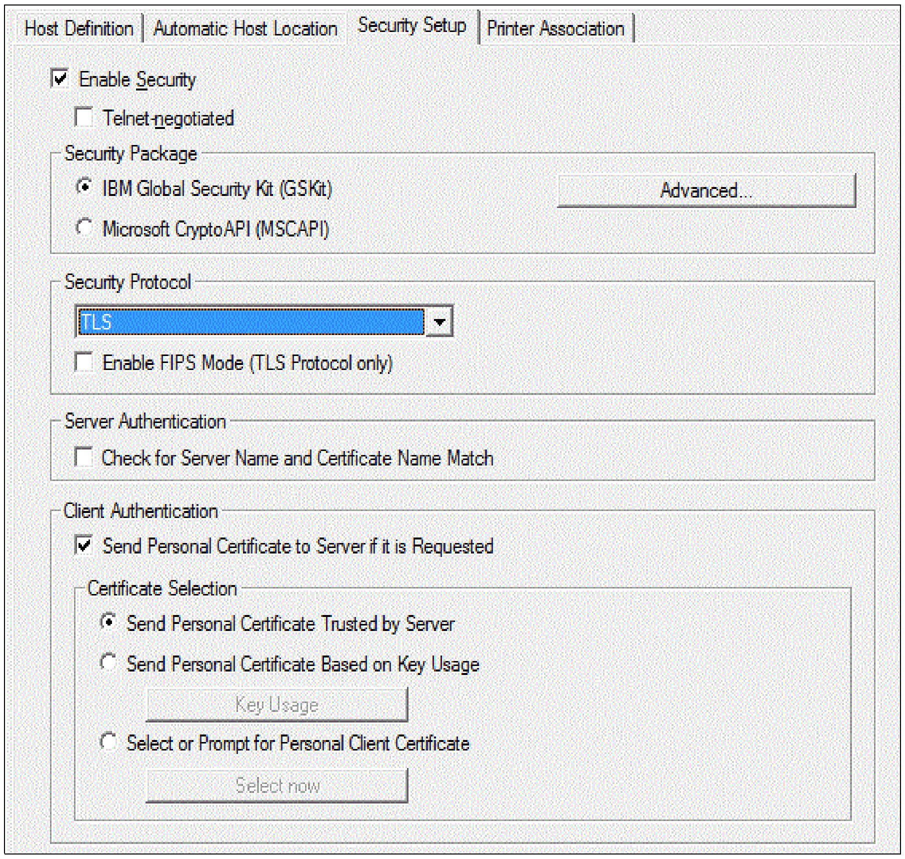
Example 2-5 SSL-related messages when the TCP/IP stack is restarted

```
xautolog tcpip
Command accepted
Ready; T=0.01/0.01 09:39:23
TCPIP   : NIC 3020 is created; devices 3020-3022 defined
AUTO LOGON  ***       TCPIP    USERS = 24
HCPCLS6056I XAUTOLOG information for TCPIP: The IPL command is verified by the I
PL command processor.
TCPIP   : z/VM V5.4.0    2009-09-09 09:47
TCPIP   : DMSACP723I D (198) R/O
TCPIP   : DMSACP723I E (591) R/O
TCPIP   : DMSACP723I F (592) R/O
TCPIP   : Ready; T=0.01/0.01 09:39:23
TCPIP   : DTCRUN1022I Console log will be sent to default owner ID: TCPMAINT
[...]
TCPIP   : DTCRUN1011I No parameters in use
TCPIP   : DTCTCP001I z/VM TCP/IP Level 540
[...]
TCPIP   : 09:39:23 DTCIPI053I SSLServerID specified, SSL will be autologged befo
re any other servers
TCPIP   : 09:39:23 DTCQDI001I QDIO device DEV@3020 device number 3020:
[...]
TCPIP   : AUTO LOGON  ***       SSLSERV  USERS = 24
SSLSERV : z/VM V5.4.0    2009-09-09 09:47
SSLSERV : DMSACP723I D (198) R/O
SSLSERV : DMSACP723I E (591) R/O
SSLSERV : DMSACP723I F (592) R/O
SSLSERV : Ready; T=0.01/0.01 09:39:33
SSLSERV : DTCRUN1022I Console log will be sent to default owner ID: TCPMAINT
SSLSERV : DTCRUN1011I Server started at 09:39:33 on 16 Sep 2009 (Wednesday)
SSLSERV : DTCRUN1011I Running server command: VMSSL
SSLSERV : DTCRUN1011I Parameters in use:
SSLSERV : DTCRUN1011I  KEYFile /etc/gskadm/Database.kdb
SSLSERV : DTCSSL2423I Using server module: SSLSERV MODULE E2 - 6/05/09 18:44:14
SSLSERV : DTCSSL002I SSLSERV main() - PROGMAP:
SSLSERV : Name        Entry       Origin      Bytes       Attributes
SSLSERV : SSLSERV     0F8E4F00    0F8E4F00    000420FB    Amode 31  Reloc
SSLSERV : DTCSSL002I DEBUG settings: Debug: 0
SSLSERV : DTCSSL002I main() started...
SSLSERV : DTCSSL015I Server initialization in progress (z/VM level 540 - PK80387
)
SSLSERV : DTCSSL100I This software incorporates the RSA algorithm
SSLSERV : DTCSSL132I Server ID: SSLSERV
SSLSERV : DTCSSL002I mainSSL() started...
SSLSERV : DTCSSL002I main(): calling CQadminMain()...
TCPIP   : 09:39:34 DTCSSL044I The SSL Server is available to handle secure conne
ctions
TCPIP   : 09:39:35 DTCSTM237I Telnet server: Using port 23
TCPIP   : 09:39:35 DTCSTM237I Telnet server: Using port 962
TCPIP   : 09:39:35 DTCSTM298I Telnet server: Line mode input will be presented w
hen requested
TCPIP   : 09:39:35 DTCSTM296I Telnet server: Erase All Unprotected (EAU) data wi
ll be transmitted
TCPIP   : 09:39:35 DTCSTM239I Telnet server: No inactivity timeout
TCPIP   : 09:39:35 DTCSTM240I Telnet server: Every 600 seconds a timing mark opt
ion packet will be sent.
TCPIP   : 09:39:35 DTCSTM299I Telnet server: EOJTIMEOUT is 120 seconds
TCPIP   : 09:39:35 DTCSTM269I Telnet server: SCANInterval is 60 seconds
TCPIP   : 09:39:35 DTCSTM255I Telnet server: Suppress-Go-Ahead enabled
TCPIP   : 09:39:35 DTCSTM291I Telnet server: TN3270E is enabled
TCPIP   : 09:39:35 DTCOTC050I ConnectExit is disabled for TN3270E sessions
TCPIP   : 09:39:35 DTCSTM256I Telnet server: Line-mode terminal names will be p
efixed with "TCPIP"
TCPIP   : 09:39:35 DTCSTM263I Telnet server: Will use logical device addresses i
n the range 00000000 through 00000FFF
TCPIP   : 09:39:35 DTCSTM305I Telnet server: Secure Connections are PREFERRED
TCPIP   : 09:39:35 DTCSTM309I Telnet server: TLS Label is SSLSRV
TCPIP   : 09:39:35 DTCSTM243I ****************************************
TCPIP   : 09:39:35 DTCSTM244I Log of IBM TCP/IP Telnet Server Users started on 0
9/16/09 at 09:39:35
TCPIP   : AUTO LOGON  ***       PORTMAP  USERS = 24
TCPIP   : 09:39:35 DTCMON402W TCP/IP was very low on Small data buffers. Of 12 b
locks, 11 are free after 11 more were allocated.
TCPIP   : 09:39:35 DTCSTM213I Telnet server: Global connection to *CCS CP System
 Service established
TCPIP   : 09:39:35 DTCSTM216I Telnet server: First line of *CCS logo is: z/VM ON
LINE
TCPIP   :          --VMLINUX9--PRESS BREAK KEY TO BEGIN SESSION
TCPIP   : 09:39:35 DTCSTM326I Telnet server: Ready to handle secure connections.
```

Note: Only direct connections to the system will be encrypted. Internal connections are still made in clear text. If dialing into the system through PVM, for instance, this connection remains unencrypted.

Telnet 3270 client configuration

Depending on the 3270 client that you are using, client-side certificates might be required. For instance, a Linux x3270 client makes no use of a client-side certificate, but an IBM Personal Communications 3270 client requires a client certificate. Client side authentication may be required by the 3270 server. To configure PComm TN3270 to comply with client-side authentication requests, follow the Security Setup panel under the link configuration dialog as shown in Figure 2-6.



[✓ Enable Security]
[ ] Telnet-negotiated

Security Package
[✓ IBM Global Security Kit (GSKit)]
Microsoft CryptoAPI (MSCAPI)

Security Protocol
TLS
[ ] Enable FIPS Mode (TLS Protocol only)

Server Authentication
[ ] Check for Server Name and Certificate Name Match

Client Authentication
[✓ Send Personal Certificate to Server if it is Requested]

Certificate Selection
[✓ Send Personal Certificate Trusted by Server]
Send Personal Certificate Based on Key Usage
Select or Prompt for Personal Client Certificate

Figure 2-6 PComm TN3270 security panel

Although detailing each client configuration specifics is not the purpose of this book, here are the steps to follow when you want a client-side certificate:

1. Get the CA certificate from the CA.
2. Issue a certificate request from the client, and send it to the CA.
3. Have the certificate request signed by the CA.
4. Send the CA certificate and signed certificate request back to the client.
5. Update the client configuration with these certificates.

Note: When the CA is hosted in z/VM, Linux client certificate requests will have to be created by using the IBM tool **gsk7ikm** (or the equivalent CLI tool **gsk7cmd**). Depending on the middleware, exporting the certificate out of the key database might be required.

2.5.3 z/VM FTP server

The z/VM FTP server is one of the first TCP/IP servers that the user configures on a system. Exchanging data between a workstation and the z/VM systems, for instance to upload the Linux boot files to the 191 MDISK of a guest, is extremely useful.

By default, the z/VM FTP server does not allow any secured connections. Its configuration can be updated to allow or require secure connections. The FTP protocol uses at least two ports. By default, port 20 is used for control connections, and port 21 is used for transferring data. If secured data connections are configured, control connections must be secured as well.

To configure z/VM FTPSERVE to accept SSL-secured connections, the `SRVRFTP CONFIG E` file must be updated. If no previous configuration was done, the sample file `SRVRFTP SCONFIG E` can be renamed and edited. The relevant part of the `SRVRFTP CONFIG` file is shown in Example 2-6.

Example 2-6 The SRVRFTP CONFIG file

```
SRVRFTP  CONFIG   E1  V 80  Trunc=80 Size=310 Line=164 Col=1 Alt=0

===== ; ------------------------------------------------------------------------
===== ; The TLSLABEL statement specifies the label of the FTP server
===== ; certificate for securing connections using TLS.  There is no default
===== ; label.
===== ; ------------------------------------------------------------------------
===== ;
===== TLSLABEL SSLSRV
===== ;
===== ;
===== ; ------------------------------------------------------------------------
===== ; The SECURECONTROL statement specifies the FTP server-wide minimum
===== ; security level for control connections.
===== ;
===== ;  NEVER    - specifies that secure control connections are not allowed.
===== ;             Attempts by the client to secure the control connection
===== ;             will receive an error reply. This is the default.
===== ;
===== ;  ALLOWED  - specifies optional TLS security on control connections.
===== ;             When ALLOWED, secure control connections are created when
===== ;             the client asks for them.
===== ;
===== ;  REQUIRED - specifies control connections must be secured using TLS.
===== ;             When REQUIRED is specified, clear control connections
===== ;             are not allowed.
===== ; ------------------------------------------------------------------------
===== ;
===== SECURECONTROL ALLOWED
===== ;
===== ; ------------------------------------------------------------------------
```

```
===== ; The SECUREDATA statement specifies the FTP server-wide minimum
===== ; security level for data connections.
===== ;
===== ;  NEVER    - specifies that secure data connections are not allowed.
===== ;             Attempts by the client to secure data connections
===== ;             will receive an error reply. This is the default.
===== ;
===== ;  ALLOWED  - specifies optional TLS security on data connections.
===== ;             When ALLOWED, secure data connections are created when
===== ;             the client asks for them.
===== ;
===== ;  REQUIRED - specifies data connections must be secured using TLS.
===== ;             When REQUIRED is specified, clear data connections
===== ;             are not allowed.
===== ; ------------------------------------------------------------------
===== ;
===== SECUREDATA ALLOWED
```

The TLSLABEL refers to the certificate label that is to be used by the server for establishing an FTP-over-TLS connection. Refer to “Create a certificate request for the SSL server” on page 19 to create a certificate for the FTPSERVE user ID.

Note: Each time a new certificate is added to the certificate database, the SSL server configuration must be refreshed. Log on as TCPMAINT and use the SSLADMIN REFRESH command to reflect the changes in the certificate database.

The `SECURECONTROL ALLOWED` statement enables unsecured or secured control connections; the SECUREDATA ALLOWED does the same for data connections.

Upon startup, FTPSERVE confirms that the secure control and data connections are configured, as shown in Example 2-7.

Example 2-7 FTPSERVE startup messages

```
TCPIP   : AUTO LOGON  ***       FTPSERVE USERS = 25
FTPSERVE: z/VM V5.4.0    2009-09-09 09:47
FTPSERVE: DMSACP723I D (198) R/O
FTPSERVE: DMSACP723I E (591) R/O
FTPSERVE: DMSACP723I F (592) R/O
FTPSERVE: Ready; T=0.01/0.01 09:39:45
FTPSERVE: DTCRUN1022I Console log will be sent to default owner ID: TCPMAINT
FTPSERVE: DTCRUN1011I Server started at 09:39:45 on 16 Sep 2009 (Wednesday)
FTPSERVE: DTCRUN1011I Running server command: SRVRFTP
FTPSERVE: DTCRUN1011I No parameters in use
FTPSERVE: DTCFTS0018I VM TCP/IP Server-FTP Level 540 09:39:45 EDT WEDNESDAY 2009
-09-16
FTPSERVE: DTCFTS0002I Using translate table STANDARD TCPXLBIN.
FTPSERVE: 09:39:45 DTCFTS0014I Using port FTP control (21).
FTPSERVE: 09:39:45 DTCFTS0015I Inactivity time is 300.
FTPSERVE: 09:39:45 DTCFTS0371I Default list format is VM
FTPSERVE: 09:39:45 DTCFTS0373I Default automatic translation is turned OFF
FTPSERVE: 09:39:45 DTCFTS0414I Default secure control connection level is ALLOWE
D
FTPSERVE: 09:39:45 DTCFTS0415I Default secure data connection level is ALLOWED
FTPSERVE: 09:39:45 DTCFTS0416I TLS label is SSLSRV
```

```
FTPSERVE: 09:39:45 DTCFTS7003I Diagnose 88 authorization confirmed
FTPSERVE: 09:39:45 DTCFTS7003I Diagnose D4 authorization confirmed
FTPSERVE: 09:39:45 DTCFTS8466I SSL server is available and TLS label SSLSRV has
been verified
FTPSERVE: 09:39:45 DTCFTS0021I Server-FTP: Initialization completed 09:39:45 EDT
 WEDNESDAY 2009-09-
FTPSERVE: 09:39:45 16
```

Note: If secured connections are required, an FTP Secure-capable client will be necessary to connect to the FTP Server. We used the FileZilla client for our tests, using FTP over the explicit TLS/SSL option.

2.6 Securing z/VM resources

Securing the access to the z/VM partition is a first step to securing z/VM resources. Most of the work in securing a z/VM partition is to secure the access to the resources managed by z/VM.

2.6.1 Built-in security features

The z/VM hypervisor has a set of built-in functions that allow a systems administrator to define groups of users according to their needs, to secure access to the resources used by the virtual machines under the control of the CP, and to perform user authentication and authorization.

CP privilege classes

Default z/VM installations come with a set of commands divided into eight groups, or privilege classes. Depending on their functions, users are part of one group or another. Table 2-1 details the predefined roles and classes that are available on a z/VM system, and, where available, examples of commands that are available to a privilege class.

Table 2-1 Default z/VM privilege classes

Class	User function	Example command
A	System Operator: The class A user is responsible for the overall performance of the z/VM system.	SHUTDOWN
B	System Resource Operator: The class B user controls all the real resources of a z/VM system.	VARY CHPID, VARY PROCESSOR
C	System Programmer: The class C user updates or changes system-wide parameters of a z/VM system.	DUMP (Host storage)
D	Spooling Operator: The class D user controls spool files and the system's real reader, printer, and punch.	SPXTAPE, CHANGE
E	System Analyst: The class E user examines and saves system operation data in specified z/VM storage areas.	SAVESYS, SAVESEG
F	Service Representative: The class F user obtains and analyzes in detail data about I/O devices connected to a z/VM system. This class is reserved for IBM use.	-

Class	User function	Example command
G	General User: The class G user controls functions associated with a given virtual machine.	IPL
ANY	These commands are available to any user.	-
H	Reserved for IBM use.	-
I-Z and 1-6	These classes are available to customers to redefine privilege as required by their organization.	-

According to the privilege class of the user, CP allows or prevents the execution of a command. Example 2-8 on page 28 demonstrates what happens when a class G user tries to execute a class A command, such as **shutdown**. CP prevents it, informing the user that the particular command is unknown.

Example 2-8 Class G user trying to execute a class A command

```
L LNXRH1
ENTER PASSWORD  (IT WILL NOT APPEAR WHEN TYPED):

NIC C200 is created; devices C200-C202 defined
z/VM Version 5 Release 4.0, Service Level 0902 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0003 RDR,   NO PRT,   NO PUN
LOGON AT 11:07:29 EDT FRIDAY 09/11/09
z/VM V5.4.0    2009-09-09 09:47
Ready; T=0.01/0.01 11:07:29

CMS
q privclas
Privilege classes for user LNXRH1
        Currently: G
        Directory: G
Ready; T=0.01/0.01 11:07:35
shutdown
Unknown CP/CMS command
                                                      RUNNING   VMLINUX9
```

There are 14 unused privilege classes available for clients to define their own classes, according to their organizational needs. For more information about how to define new privilege classes, refer to “Planning a new user class structure” in *z/VM: CP Planning and Administration,* SC24-6083 (for V5R4.0).

Protecting z/VM guest LANs

Unless specified otherwise, z/VM guest LANs are created in UNRESTRICTED mode, meaning that anybody can connect to the virtual LAN. Specifying the highly recommended RESTRICTED option on the DEFINE LAN command, as demonstrated in Example 2-9, enables LAN access restriction.

Note: When you define a z/VM Virtual Switch, it is automatically created in RESTRICTED mode.

Example 2-9   Creating a RESTRICTED guest LAN

```
define LAN REST OWNERID SYSTEM RESTRICTED
LAN SYSTEM REST is created
Ready; T=0.01/0.01 11:45:13
q LAN REST
LAN SYSTEM REST         Type: HIPERS  Connected: 0    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    IP        MFS: 16384      Accounting: OFF
  IPTimeout: 5
  Isolation Status: OFF
Ready; T=0.01/0.01 11:45:20
                                                            RUNNING   VMLINUX9
```

Regular class G users are unable to connect to the LAN created, as shown in Example 2-10.

Example 2-10   Trying to access a restricted guest LAN without proper authorization

```
CMS
couple c300 to SYSTEM REST
HCPNDF6011E You are not authorized to COUPLE to SYSTEM REST
Ready(06011); T=0.01/0.01 11:51:01
                                                            RUNNING   VMLINUX9
```

To be able to access a restricted guest LAN, users have to be explicitly granted to do so, as shown in Example 2-11.

Example 2-11   Granting user LNXRH1 access to REST guest LAN

```
set lan rest ownerid system grant lnxrh1
Command complete
Ready; T=0.01/0.01 11:54:43
                                                            RUNNING   VMLINUX9
```

When explicitly granted, users are allowed to couple to the guest LAN as demonstrated in Example 2-12.

Example 2-12   Accessing a restricted guest LAN after proper granting

```
CMS
couple c300 to system rest
NIC C300 is connected to LAN SYSTEM REST
Ready; T=0.01/0.01 11:57:45
                                                            RUNNING   VMLINUX9
```

Protecting access to minidisks

Each virtual machine defined in z/VM User Directory is given access to a set of disks or minidisks. These disks (or minidisks) are defined with a given access mode: they can be set up for exclusive use by a virtual machine, or they can be sharable between users. In the latter case, the system administrator has the ability to set a password on the disk, which will be required each time a user accesses a disk.

The password is set in the z/VM user directory entry for the user, as shown in Example 2-13.

Example 2-13   MDISK password in the z/VM user directory

```
01804 USER SSLSERV SSLSERV 256M 2G G
01805  INCLUDE TCPCMSU
```

```
POSIXINFO UID 7 GNAME security
IUCV ALLOW
OPTION ACCT MAXCONN 1024 QUICKDSP SVMSTAT APPLMON
SHARE RELATIVE 3000
LINK 5VMTCP40 491 491 RR
LINK 5VMTCP40 492 492 RR
LINK TCPMAINT 591 591 RR
LINK TCPMAINT 592 592 RR
LINK TCPMAINT 198 198 RR
MDISK 191 3390 2347 001 LX9W02  MR RSSLSERV WSSLSERV MSSLSERV
```

When you want to link to a SSLSERV 191 disk, you must provide the password corresponding to the mode that you want to use for the link. In the case of Example 2-14, the password is MSSLSERV. If the password that is provided is incorrect, the link fails.

Example 2-14 Linking to the SSLSERV 191 disk

```
link SSLSERV 191 555 MW
ENTER MULT PASSWORD:

HCPLNM114E SSLSERV 0191 not linked; mode or password incorrect
Ready(00114); T=0.01/0.01 13:34:02
link SSLSERV 191 555 RR
ENTER READ PASSWORD:

Ready; T=0.01/0.01 13:34:24
                                                    RUNNING   VMLINUX9
```

Note: This example works only for MDisks (either ECKD™ or FBA). If your installation is using dedicated disks, these are completely transparent to z/VM, which only acts as a pass-through.

2.6.2 Securing z/VM resources with RACF

The z/VM Resources Access Control Facility (RACF) Security Server is an IBM System z security product.

RACF performs the following operations:

- Authenticates the identity of users connecting to the system.
- Authorizes user access to system resources under its surveillance.
- Records and reports accesses to the system.

A resource is any piece of information stored on your system, and the means to access this information. For instance, a virtual machine minidisk is a resource, and so is the virtual switch that is connected to get network access.

RACF provides all the tools and capabilities to implement security policies. Among these capabilities, RACF includes a predefined set of classes that can be used to protect a specific type of resource, as detailed in Table 2-2.

Table 2-2 RACF predefined classes

RACF Class	Resources protected
VMMDISK	When this class is activated, RACF will prevent unauthorized links and attachements to mdisks.

RACF Class	Resources protected
VMLAN	When this class is activated, all users must be explicitly authorized to connect to the virtual LANs managed by z/VM, even when granted by CP.
VMRDR	Activated, this class allows the security administrator to protect virtual unit record devices (readers, punches, and printers).
VMBATCH	When activated, this class is used to control alternate IDs. Alternate user IDs are used, for instance, by the FTP server, where you log on to a machine by using another user's identify.
SECLABEL	This class is required when implementing mandatory access control. Used in correlation with the VMMAC resource class.

For more details about how to secure z/VM resources with RACF, or to check how resources are currently being secured, refer to Chapter 8, "Best practices" on page 265.

Implementation of RACF

Many books have been written about RACF and how to implement it. This book is not intended to be a detailed RACF implementation guide; RACF is discussed in many chapters, because it is the cornerstone for securing z/VM and Linux virtual machines.

For more details about RACF implementation, refer to the following documentation:

- *Program Directory for RACF Security Server*, GI11-2894 (for z/VM function level 540)
- *Security on z/VM*, SG24-7471

For reference, here are the steps that the authors followed to proceed with RACF implementation on their system. This procedure is not intended to replace the documentation, but more as a checklist of the main steps to go through.

Note: Only the steps relevant to our setup are mentioned here. Refer to the aforementioned documentation for details.

The procedure is as follows:

1. Check the user directory for statements that are unacceptable for RACF:
   - NOLOG passwords
   - DUPLICATE user IDs (if you plan to share the RACF database)
   - ALL passwords for MDISKs on IBM Open Systems Adapter Support Facility (OSA/SF) for VM machines (not in use in our setup)
2. Create the RPIDIRCT SYSUT1 file, which contains all statements to populate the RACF database with the content of the directory.
3. Review and update RPIDIRCT SYSUT1 if required.
4. Customize RACF SMF record processing.
5. Delete or replace the ICHRCX02 exit. (optional step)
6. Install the CP part of RACF.
7. IPL the RACF-enabled CP module.

Note: If AUTO_WARM_IPL is set in SYSTEM CONFIG, it has to be disabled before you re-IPL the RACF-enabled CP module. This step can be done by editing the SYSTEM CONFIG file, located on the CF2 PARM minidisk.

8. Update the RACF database with the content of the USER DIRECTORY.
9. Set the RACF options of your choice.
10. Determine control and audit options.
11. Set up dual registration, if you are using DirMaint.
12. Put RACF into production.

2.6.3 Securing TCP/IP service machines with RACF

The z/VM FTP Server is in use in our installation to provide an easy way to exchange data between our workstations and the z/VM system. To enable RACF with the FTP Server, we followed the steps detailed in “Appendix A - Using TCP/IP with an External Security Manager” in the guide *z/VM: TCP/IP Planning and Customization* (for V5R4.0), SC24-6125.

Note: In step 7 of that appendix, you must modify the DTCPARMS server entries to enable the required RACF interfaces, as follows:

```
:ESM_Enable.YES
```

This statement must be added to the `SYSTEM.DTCPARMS` file. Do not modify the `IBM.DTCPARMS` file that is provided by IBM.

For instance, for the FTP Server, SYSTEM DTCPARMS would appear like this example:

```
.*******************************************************************
.* SYSTEM DTCPARMS created by DTCIPWIZ EXEC on 29 Aug 2008
.* Configuration program run by MAINT at 10:22:23
.*******************************************************************
.*:nick.TCPIP    :type.server
.*               :class.stack
.*               :attach.3020-3022
:nick.LDAPSRV   :type.server  :class.ldap
                :ESM_Enable.YES
:NICK.FTP       :TYPE.CLASS
                :ESM_ENABLE.YES
```

In the example, LDAPSRV is also modified to take advantage of RACF capabilities.

2.6.4 Centralized authentication

z/VM user authentication is being kept in a central repository: either the User Directory, DirMaint control files, or the RACF database, depending on the z/VM configuration. To extend user authentication to an entire IT environment, the most common way is to rely on Directory Services. A directory is a set of objects, with attributes, organized in a hierarchical manner. An object can be a user, and attributes can be its first and last names, and password.

The protocol used to access a directory is Lightweight Directory Access Protocol (LDAP). Directory servers are commonly referred to as LDAP servers. On System z, several options are available:

- Running the LDAP server in z/OS. This might be the preferred option if your installation already uses z/OS.
- Running the LDAP server in z/VM. This is the option chosen by the authors, and is further described in Chapter 3, "Configuring and using the System z LDAP servers" on page 51.
- Running the LDAP server in Linux. This is the option if your requirements cannot be met with the two previous options. This topic is discussed in 3.8, "Using an OpenLDAP server with the z/VM LDAP server" on page 86.

2.6.5 Centralized audit

When discussing the security of an IT environment, one of the key aspects to keep in mind is auditability. No matter how many resources (people, hardware, software, or money) have been involved in defining and implementing security procedures, they are useless if they cannot be audited.

IT managers must have a detailed understanding of who is doing what on the system (logs). A company's CFOs or CIOs must know at any time whether their company complies with the recent regulations, and whether adjustments are required to their company's policies (audit).

Note: Logs and audits are different:

- Logs are written by each of the machines to provide information about the operations being performed in the machine (return codes). Often, log format is specific to a virtual machine.
- An audit is collected system-wide, and is written in a standardized way (System Management Facility, SMF, record in z/VM), depending on the event and virtual machine.

RACF provides all the capabilities required to establish and enforce a company's security rules, as well as to record data to be used as input for reports and dashboards. The RACF administrator defines sets of rules to ensure user identification and authentification, and control access to the system and virtual machine resources. The administrator also provides auditing responsibility to a user or group of users who will be responsible for verifying the accuracy of the security rules in place and to ensure, for instance, that the installation is compliant with the regulations demanded by the company's business.

To help in the audit process, RACF provides a set of routines, functions, and utilities:

- Logging routines that record the information required
- Audit control functions that let the auditor specify which information has to be logged
- Utilities to convert RACF records into human readable format

Recording information that your installation needs

For an audit to be efficient, the system must record all the information required to verify that the security rules of your installation are correctly enforced.

By default, RACF logs the following events, because knowing about them for efficient auditing is crucial:

- Every use of the RVARY or SETROPTS command

- Every time the request `RACROUTE REQUEST=VERIFY` fails
- Every time the console operator grants access to a resource as part of the failsoft processing performed when RACF is inactive

In contrast, some events are never logged, because they do not provide relevant audit information. These events are the use of the following RACF commands: LISTDSD, LISTGRP, LISTUSER, RLIST, LDIRECT, LFILE, SRFILE, SRDIR, and SEARCH.

All other events can be logged. Note the following information:

- Owners of resources can specify logging options in the resource profile, to indicate which levels of access to log (READ, UPDATE, ALTER, or CONTROL), and which conditions to log (success, failures, or both). This is called owner-controlled logging.
- The auditor can specify extra logging options, to record additional events, for instance (but not limited to):
  - Changes to any RACF profiles
  - All RACF commands that a SPECIAL user issues
  - All unauthorized attempts to use RACF commands

  Note: For the complete list of events that can be logged, see *z/VM: RACF Security Server Auditor's Guide*, SC24-6143.

  The auditor can bypass owner-controlled logging and force a proper logging level, if required. This is referred to as auditor-controlled logging.

Setting audit controls

RACF can provide a specific user or group of users enough privileges to run the audit of an installation. To do so, the RACF security administrator (by default, user SYSADMIN) has to set the AUDITOR attribute for this user, as shown in Example 2-15.

Example 2-15 Granting a user the AUDITOR attribute

```
rac alu GUIGUI AUDITOR
```

Keep in mind that separating powers and authorities is an absolute must. The security administrator is responsible for implementing the security rules in an installation. A user with AUDITOR attribute is only responsible for auditing the system; the user should not be able to modify the rules that are in place. Responsibilities should not be intertwined.

To check the audit functions activated system-wide, the auditor can use the **`rac setropts list`** command. As shown in Example 2-16, RACF is configured to audit the USER, GROUP, VMDISK, VMLAN, and SURROGAT classes.

Example 2-16 Querying audit functions

```
rac setropts list
ATTRIBUTES = INITSTATS NOWHEN(PROGRAM) SAUDIT CMDVIOL NOOPERAUDIT
STATISTICS = NONE
AUDIT CLASSES = USER GROUP VMMDISK VMLAN SURROGAT
ACTIVE CLASSES = DATASET USER GROUP SECLABEL VMMDISK VMRDR VMCMD VMBATCH
                 VMLAN VMMAC FACILITY SURROGAT XFACILIT GXFACILI
GENERIC PROFILE CLASSES =  NONE
GENERIC COMMAND CLASSES =  NONE
GENLIST CLASSES =  NONE
GLOBAL CHECKING CLASSES =  NONE
```

```
RACLIST CLASSES =  SECLABEL FACILITY XFACILIT
[...]
```

To add another class to the list of audited classes, issue the command shown in Example 2-17, replacing the VMRDR class with the one required:

Example 2-17   Adding class VMRDR to the list of audited classes

```
Ready; T=0.01/0.01 14:54:51
rac setropts audit(VMRDR)
Ready; T=0.01/0.01 14:55:03
rac setropts list
ATTRIBUTES = INITSTATS NOWHEN(PROGRAM) SAUDIT CMDVIOL NOOPERAUDIT
STATISTICS = NONE
AUDIT CLASSES = USER GROUP VMMDISK VMRDR VMLAN SURROGAT
ACTIVE CLASSES = DATASET USER GROUP SECLABEL VMMDISK VMRDR VMCMD VMBATCH
                 VMLAN VMMAC FACILITY SURROGAT XFACILIT GXFACILI
GENERIC PROFILE CLASSES =  NONE
GENERIC COMMAND CLASSES =  NONE
GENLIST CLASSES =  NONE
GLOBAL CHECKING CLASSES =  NONE
RACLIST CLASSES =  SECLABEL FACILITY XFACILIT
[...]
```

Many classes, by default, are defined to create audit data when a failure to access a resource occurs. The AUDITING field in Example 2-18 reflects this.

Example 2-18   Querying MAINT.CF1 profile logging options

```
rac rlist VMMDISK MAINT.CF1 ALL
CLASS      NAME
-----      ----
VMMDISK    MAINT.CF1

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    MAINT           NONE               NONE    NO

[...]
AUDITING
--------
FAILURES(READ)

GLOBALAUDIT
-----------
NONE
[...]
CREATION DATE  LAST REFERENCE DATE  LAST CHANGE DATE
 (DAY) (YEAR)       (DAY) (YEAR)      (DAY) (YEAR)
-------------  -------------------  ----------------
  261    09          261    09         261    09

ALTER COUNT   CONTROL COUNT   UPDATE COUNT   READ COUNT
-----------   -------------   ------------   ----------
  000000        000000        000000         000000

USER      ACCESS   ACCESS COUNT
----      ------   ------ -----
MAINT     ALTER       000000
VSMSERVE  READ        000000
```

```
VSMWORK1  READ       000000
VSMWORK2  READ       000000
VSMWORK3  READ       000000

   ID     ACCESS  ACCESS COUNT  CLASS                 ENTITY  NAME
-------- -------  ------------ -------- ---------------------------------------
NO ENTRIES IN CONDITIONAL ACCESS LIST
Ready; T=0.01/0.01 14:59:01
```

This step might not be sufficient to ensure proper auditing. When a profile must be updated, two options are available, from the audit perspective:

- The security administrator updates the AUDITING value to fit the auditing needs. Example 2-19 shows how the security administrator updated a VSWITCH profile to record every coupling request to VSWITCH1, whether it is a success or failure.

  Example 2-19 Updating a VSWITCH profile to ensure proper auditing

  ```
  rac ralter vmlan system.vswitch1 audit(all(update))
  Ready; T=0.01/0.01 16:37:05
  rac rl vmlan system.vswitch1
  CLASS      NAME
  -----      ----
  VMLAN      SYSTEM.VSWITCH1

  LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
  -----  --------   ----------------  -----------  -------
   00    SYSADMIN        NONE              ALTER    NO

  INSTALLATION DATA
  -----------------
  NONE

  APPLICATION DATA
  ----------------
  NONE

  AUDITING
  --------
  ALL(UPDATE)

  GLOBALAUDIT
  -----------
  NONE

  NOTIFY
  ------
  NO USER TO BE NOTIFIED
  Ready; T=0.01/0.01 16:37:18

                                                       RUNNING   VMLINUX9
  ```

- If updating a resource profile is not possible, the auditor can still circumvent default auditing options by using the **rac ralter** command with the **GLOBALAUDIT** attribute, as shown in Example 2-20.

Example 2-20 Modifying MAINT CF1 to log every access attempt

```
rac ralter vmmdisk maint.cf1 globalaudit(ALL)
Ready; T=0.01/0.01 15:00:32
rac ralter vmmdisk maint.cf1 globalaudit(ALL(UPDATE))
Ready; T=0.01/0.01 15:00:32
rac rlist VMMDISK MAINT.CF1 ALL
CLASS      NAME
-----      ----
VMMDISK    MAINT.CF1
[...]
AUDITING
--------
FAILURES(READ)

GLOBALAUDIT
-----------
ALL(UPDATE)
[...]
```

Note: From the auditor point of view, no difference exists between auditing the accesses to a minidisk or to a LAN. Both are RACF resources and are treated equally.

Only the security administrator can update the AUDIT value in a resource profile. The auditor is responsible for updating the GLOBALAUDIT value.

For more information about configuring audit controls, refer to *z/VM: RACF Security Server Auditor's Guide*, SC24-6143.

Processing audit records on z/VM

Under normal conditions, RACF logs a record for each event occurring on the system, if this event is configured to be logged. These audit records are SMF records (same as z/OS SMF records) type 80, 81, or 83:

- An SMF record type 80 is a RACF processing record, created for all other events.
- An SMF record type 81 is created to log RACF initialization data.
- An SMF record type 83 is created to log LDAP server operations.

Upon startup, RACF relies on the file `SMF CONTROL`, located on disk RACFSMF's 191 disk, to locate the minidisk on which to store its SMF records. By default, two minidisks are configured to host RACF SMF records: the RACFVM 301 and 302 minidisks. When RACFSMF decides to use the second disk, it updates the SMF CONTROL file to reflect the change in the configurations. Audit SMF records are stored in a file called `SMF DATA`. See Example 2-21.

Example 2-21 SMF CONTROL file

```
SMF      CONTROL  H1  F 100  Trunc=100 Size=1 Line=0 Col=1 Alt=0

00000 * * * Top of File * * *
00001 CURRENT 301 K PRIMARY 301 K SECONDARY 302 K 10000 VMSP CLOSE 001 SEVER NO
0 RA
00002 * * * End of File * * *
```

The `SMF DATA` file is first stored on the RACFVM 301 minidisk. When this disk becomes full, RACFSMF does a rotation of the log file, and then starts filling in the 302 minidisk.

The z/VM RACF Security server provides a number of tools or solutions to process the RACF SMF audit records:

- Data Security Monitor, or DSMON

  This tool provides reports about the current state of the RACF implementation.

- RACF SMF Unload utility

  This tool creates a sequential file from RACF auditing data. In turn, this file can be processed so it can be:

  - Viewed directly
  - Used as input to your own processing utilities
  - Manipulated with sort/merge programs
  - Output as XML-formatted file for viewing in a web browser
  - Used as input to a database for producing customized dashboards

- RACF Report Writer

  This tool creates easy-to-read reports based on the SMF data, which can be filtered to produce documents detailing access to a particular resource, or user and group activity for instance.

  Note: This tool is no longer recommended. It does not support some of the SMF records produced by the latest versions of RACF.

- Merging the records

  Merge the records with z/OS SMF records, and analyze those records by using the same procedures as for z/OS SMF records.

- Archiving the records

  The archiving task is handled by the RACFSMF service machine, either on a regular basis or when the disk that is holding the `SMF DATA` file is full.

Using the RACF SMF Unload utility (RACFADU)

This utility can be used by the auditor to export the SMF records to a file for further processing or analysis. The output can be either a flat file or XML. To use the RACFADU tool, the auditor must have been granted read access to the RACFVM 305 minidisk (which holds the binary of the tool), as well as 301 and 302 minidisks, where the SMF DATA file is located.

To use RACFADU:

1. Log on as the user that has the AUDITOR attribute.
2. Link RACFVM 301 and 302 as read-only. Additionally, 305 should be accessed as B, as shown in Example 2-22.

   Note: Auditor has been given the authorization by the security administrator to link to RACFVM 301, 302, and 305.

Example 2-22 RACFADU setup

```
 ICH70001I GUIGUI   LAST ACCESS AT 09:47:29 ON THURSDAY, SEPTEMBER 24, 2009
z/VM Version 5 Release 4.0, Service Level 0902 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0001 RDR,   NO PRT,   NO PUN
LOGON AT 11:46:11 EDT THURSDAY 09/24/09
z/VM V5.4.0    2009-09-09 09:47
```

```
Ready; T=0.01/0.01 11:46:12
link racfvm 305 305 rr
DASD 0305 LINKED R/O; R/W BY RACFVM      ; R/O BY SYSADMIN
Ready; T=0.01/0.01 11:46:18
link racfvm 301 301 rr
DASD 0301 LINKED R/O; R/W BY RACFVM
Ready; T=0.01/0.01 11:46:26
link racfvm 302 302 rr
Ready; T=0.01/0.01 11:46:34
acc 305 B
DMSACP723I B (305) R/O
Ready; T=0.01/0.01 11:48:37
RUNNING   VMLINUX9
```

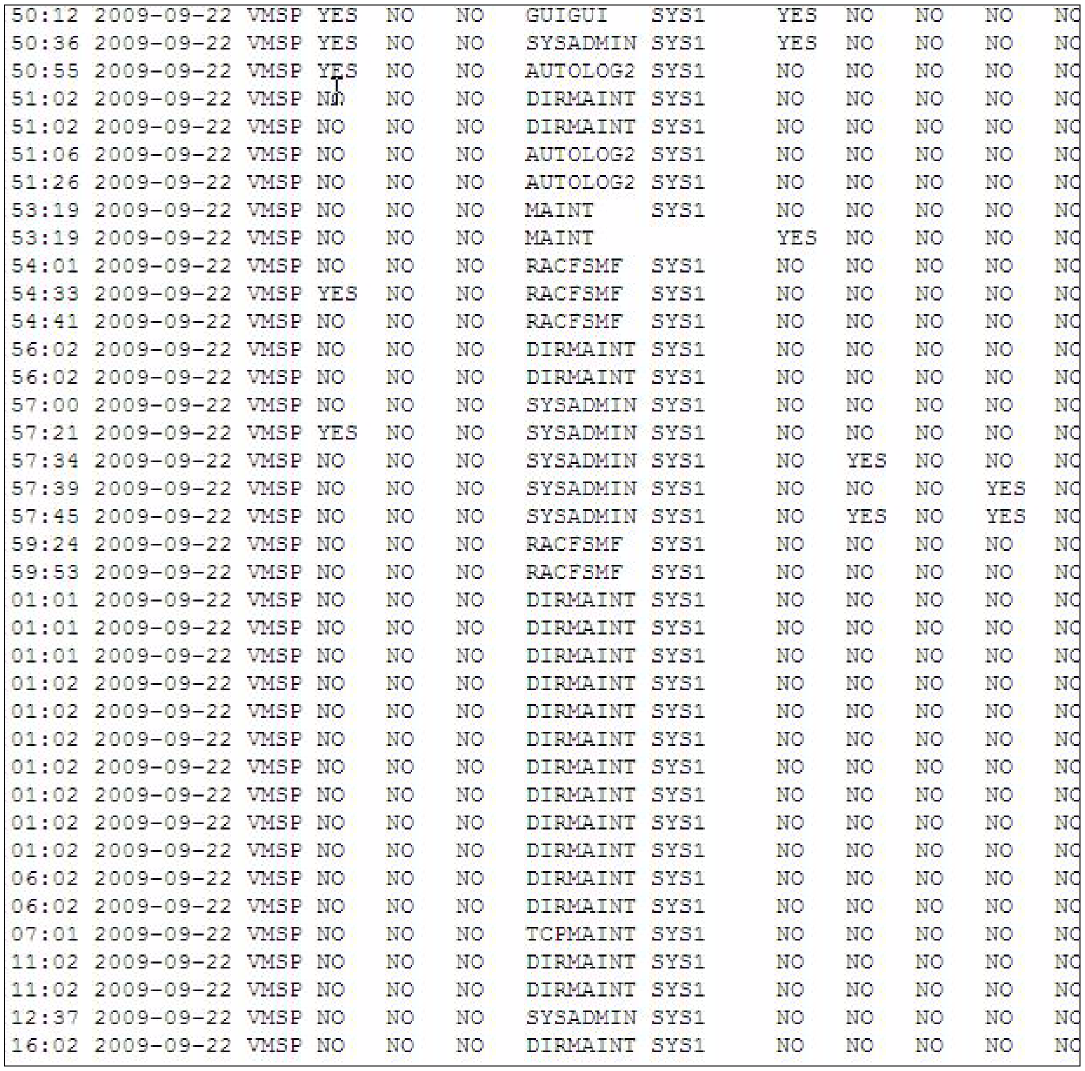
3. Make sure that enough free space is available on the disk to hold the output file.
4. Issue the RACFADU command and fill in the required fields. The panel looks as captured in Example 2-23.

Example 2-23 RACFADU panel

```
                     RACF SMF Unload Utility - Input Panel




  . Virtual address of input SMF data minidisk       0301

  . Virtual address of output minidisk               0191

  . Filename and filetype of  sequential             RACFADU   OUTPUT
    output file

  . Filename and filetype of  XML easily readable
    output file

  . Filename and filetype of  XML compressed         ________  ________
    output file

                   PF1 = Help    PF2 = Execute   PF3 = Quit
                          ENTER = Verify input fields

Enter CP/CMS Commands below:
====>
```

5. Retrieve the output file for analysis or further processing.

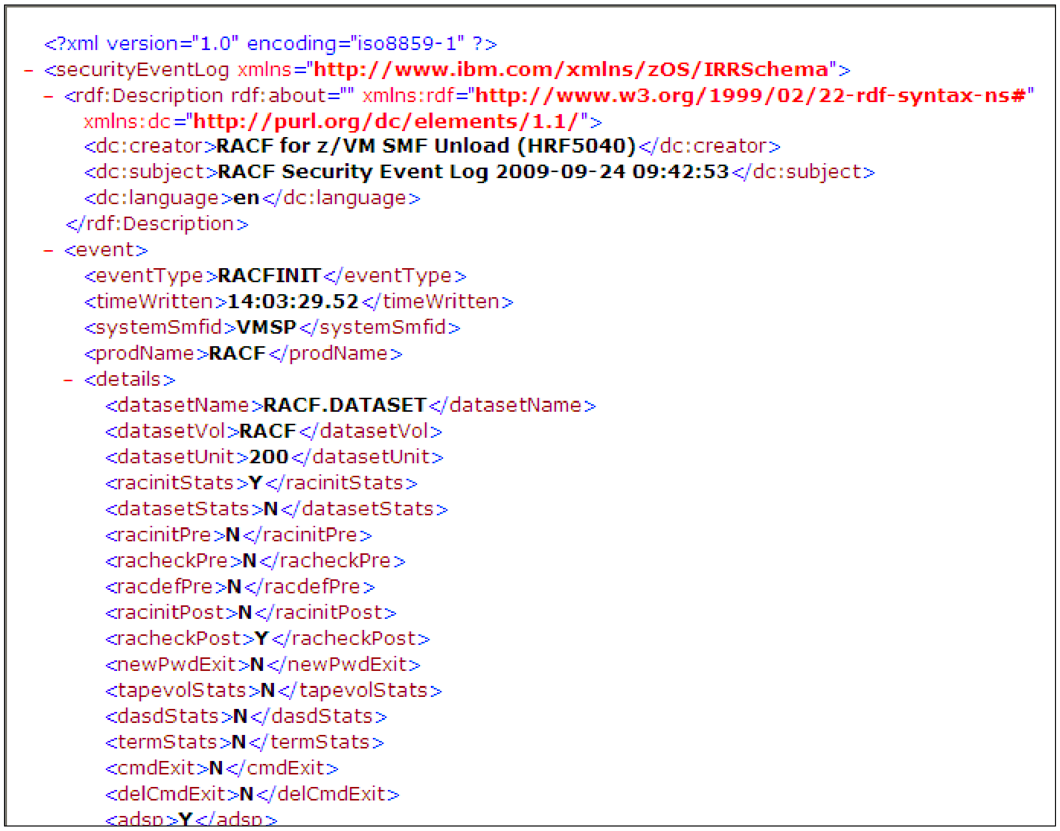
The flat file output looks like the example in Figure 2-7 on page 40.



```
50:12 2009-09-22 VMSP YES  NO   NO   GUIGUI   SYS1     YES  NO   NO   NO   N
50:36 2009-09-22 VMSP YES  NO   NO   SYSADMIN SYS1     YES  NO   NO   NO   N
50:55 2009-09-22 VMSP YES  NO   NO   AUTOLOG2 SYS1     NO   NO   NO   NO   N
51:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
51:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
51:06 2009-09-22 VMSP NO   NO   NO   AUTOLOG2 SYS1     NO   NO   NO   NO   N
51:26 2009-09-22 VMSP NO   NO   NO   AUTOLOG2 SYS1     NO   NO   NO   NO   N
53:19 2009-09-22 VMSP NO   NO   NO   MAINT    SYS1     NO   NO   NO   NO   N
53:19 2009-09-22 VMSP NO   NO   NO   MAINT             YES  NO   NO   NO   N
54:01 2009-09-22 VMSP NO   NO   NO   RACFSMF  SYS1     NO   NO   NO   NO   N
54:33 2009-09-22 VMSP YES  NO   NO   RACFSMF  SYS1     NO   NO   NO   NO   N
54:41 2009-09-22 VMSP NO   NO   NO   RACFSMF  SYS1     NO   NO   NO   NO   N
56:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
56:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
57:00 2009-09-22 VMSP NO   NO   NO   SYSADMIN SYS1     NO   NO   NO   NO   N
57:21 2009-09-22 VMSP YES  NO   NO   SYSADMIN SYS1     NO   NO   NO   NO   N
57:34 2009-09-22 VMSP NO   NO   NO   SYSADMIN SYS1     NO   YES  NO   NO   N
57:39 2009-09-22 VMSP NO   NO   NO   SYSADMIN SYS1     NO   NO   NO   YES  N
57:45 2009-09-22 VMSP NO   NO   NO   SYSADMIN SYS1     NO   YES  NO   YES  N
59:24 2009-09-22 VMSP NO   NO   NO   RACFSMF  SYS1     NO   NO   NO   NO   N
59:53 2009-09-22 VMSP NO   NO   NO   RACFSMF  SYS1     NO   NO   NO   NO   N
01:01 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:01 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:01 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
01:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
06:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
06:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
07:01 2009-09-22 VMSP NO   NO   NO   TCPMAINT SYS1     NO   NO   NO   NO   N
11:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
11:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
12:37 2009-09-22 VMSP NO   NO   NO   SYSADMIN SYS1     NO   NO   NO   NO   N
16:02 2009-09-22 VMSP NO   NO   NO   DIRMAINT SYS1     NO   NO   NO   NO   N
```

Figure 2-7 RACFADU output file

If XML processing is required, RACFADU can be used to create an XML output file, which can then be viewed from a web browser, as shown in Figure 2-8 on page 41.


```
  <?xml version="1.0" encoding="iso8859-1" ?>
- <securityEventLog xmlns="http://www.ibm.com/xmlns/zOS/IRRSchema">
  - <rdf:Description rdf:about="" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:creator>RACF for z/VM SMF Unload (HRF5040)</dc:creator>
      <dc:subject>RACF Security Event Log 2009-09-24 09:42:53</dc:subject>
      <dc:language>en</dc:language>
    </rdf:Description>
  - <event>
      <eventType>RACFINIT</eventType>
      <timeWritten>14:03:29.52</timeWritten>
      <systemSmfid>VMSP</systemSmfid>
      <prodName>RACF</prodName>
    - <details>
        <datasetName>RACF.DATASET</datasetName>
        <datasetVol>RACF</datasetVol>
        <datasetUnit>200</datasetUnit>
        <racinitStats>Y</racinitStats>
        <datasetStats>N</datasetStats>
        <racinitPre>N</racinitPre>
        <racheckPre>N</racheckPre>
        <racdefPre>N</racdefPre>
        <racinitPost>N</racinitPost>
        <racheckPost>Y</racheckPost>
        <newPwdExit>N</newPwdExit>
        <tapevolStats>N</tapevolStats>
        <dasdStats>N</dasdStats>
        <termStats>N</termStats>
        <cmdExit>N</cmdExit>
        <delCmdExit>N</delCmdExit>
        <adsp>Y</adsp>
```
Figure 2-8 RACFADU XML output

Note: To be able to visualize the XML output file, you must change the encoding of the file to read:

```
encoding=”iso8859-1”
```

To find the details of each SMF record, refer to *z/VM V5R4.0 RACF Security Server Macros and Interfaces,* SC24-6147.

2.7 z/VM Directory Maintenance Facility (DirMaint)

As stated in 2.3, “User directory management” on page 7, the Directory Maintenance Facility for z/VM (DirMaint) is an IBM licensed program product that is included with z/VM. It gives z/VM administrators an easy way to administer the z/VM user directory. In this section, we describe additional features of DirMaint, how to customize it and how to use it.

2.7.1 DirMaint features

DirMaint provides well-organized and secure interactive facilities for maintaining the z/VM system directory. Directory management is simplified by DirMaint’s command interface and automated facilities. DirMaint provides support for all the z/VM directory statements. Most DirMaint directory commands have the same names and format as the VM directory statements they support. DirMaint also provides additional utilities to help manage minidisk assignments and allocations, and provide a level of security regarding command authorizations and password monitoring.

Note the following information:

- DirMaint release supports the z/VM Security strategy.
- Access to minidisks is controlled by either passwords or explicit link authorization, as determined by the minidisk owner. Minidisk passwords are now optional for controlling minidisk directory links.
- DirMaint also supports control of minidisk links by an external security manager (ESM), such as RACF/VM or other vendors' ESM.
- Online information for the DirMaint Release base feature includes multicultural support.

2.7.2 Customizing DirMaint

The Program Directory for DirMaint explains the steps necessary for taking the pre-installed software and activating it for your system.

Being a licensed product, DirMaint is enabled by using a PRODUCT statement in the `SYSTEM CONFIG` file. You can verify the enabled products on your system by using the QUERY PRODUCT command, as shown in Example 2-24.

Example 2-24 Sample of QUERY PRODUCT on z/VM

```
query product
Product  State    Description
5VMDIR40 Enabled  10/21/08.10:22:47.MAINT    Install/service DirMaint using mini
disk
5VMPTK40 Enabled  09/16/09.09:01:18.MAINT    PERFKIT Minidisk Install and Servic
e
5VMRAC40 Enabled  09/18/09.11:37:01.MAINT    RACF Feature of z/VM, FL540
5VMRSC40 Enabled  10/07/08.09:20:03.MAINT    Install/Service RSCS FL540
Ready; T=0.01/0.01 15:24:21
```

DirMaint provides features that help you more easily and consistently add new user definitions to a z/VM system. Using these features increases the security of your environment by reducing the possibility of errors in virtual machine configurations.

Controlling DASD allocation

DirMaint provides a number of ways for disk space to be automatically managed.

Without DirMaint, z/VM administrators must maintain their own records for the allocation of disk space. When creating minidisks, administrators must manually define the starting point and extent of minidisks, creating a possibility for errors in calculations that could result in data corruption when the disks are used.

DirMaint provides automatic allocation capabilities that allow an administrator to simply instruct DirMaint to allocate a disk of a certain size. Depending on the level of control that the administrator requires, the allocation can be done either on a particular volume or anywhere on a set of predefined volumes. When sets of volumes are used, DirMaint can be instructed to fill up each available volume with minidisks before using the next volume, or to allocate minidisks in sequence across the volumes in the set.

EXTENT CONTROL file

The `EXTENT CONTROL` file tells DirMaint the number, type, and size of DASDs that are attached to the system and available for allocation. Basically, this file tells DirMaint where it can allocate minidisks. The file is divided into sections, with each section describing one part of the DASD management task. Example 2-25 shows the `EXTENT CONTROL` file from our z/VM system.

Example 2-25 EXTENT CONTROL file

```
* *******************************************************************
* CopyRight Notice
*  LICENSED MATERIALS - PROGRAM PROPERTY OF IBM.
*  RESTRICTED MATERIALS OF IBM.
*  5741-A05  (C) COPYRIGHT IBM CORPORATION 1979, 2004.
*  All rights reserved.
*  US Government Users Restricted Rights -
*  Use, duplication, or disclosure restricted by GSA ADP
*  schedule contract with IBM Corporation.
*  Status: 510
*  PITS:   @Z----ID
*  APAR:   @VA-----
*
* Purpose:   Default Extent Control file.
*
* Change Activity: (IBM)
* @V715DSR - New for DirMaint Version 1 Release 5 Mod 0.
* @VA61019 - Support for Multiprise 2000 variable size DASD.
* @VA61648 - Support 1084 cylinder emulated 3390 DASD.
* @VRA8FHA - Remove AUTOBLK and DEFAULTS sections.
*            Note: Valid TYPE values for REGIONS are now defined
*            in the DEFAULTS DATADVH file.
*            Overrides may still be included here.
* @VRFCWRS - Re-structure the REGIONS section header to support
*            10 digits for RegStart & RegEnd.
* ... (reserved for future change activity) ...
[...]
* Change Activity: (Local)
* 20090925 - VJC - Added REGION LXBC00
*
* *******************************************************************
:REGIONS.
  *RegionId  VolSer    RegStart      RegEnd  Dev-Type  Comments
LX9RES      LX9RES     0001          3338    3390-03
LX9W01      LX9W01     0001          3338    3390-03
LX9W02      LX9W02     0001          3338    3390-03
LX9U1R      LX9U1R     0001          3338    3390-03
LXD81E      LXD81E     0001          10016   3390-09
LXD81F      LXD81F     0001          10016   3390-09
LXDF1B      LXDF1B     0001          10016   3390-09
LXDF1C      LXDF1C     0001          10016   3390-09
LXBC00      LXBC00     START         END     9336
:END.
:GROUPS.
  *GroupName RegionList
ANY LX9U1R
:END.
```

```
:EXCLUDE.
* USERID ADDRESS
MAINT 0124
MAINT 0125
:END.
:AUTOBLOCK.
  * IBM supplied defaults are contained in the AUTOBLK DATADVH file.
  * The following are customer overrides and supplements.
  *
  *DASDType BlockSize Blocks/Unit Alloc_Unit Architecture
:END.
:DEFAULTS.
  * IBM supplied defaults are contained in the DEFAULTS DATADVH file.
  * The following are customer overrides and supplements.
  *
  *DASDType Max-Size
:END.
```

The first section, REGIONS, identifies the areas on individual DASDs that can be used for minidisk allocation. Regions usually map to individual DASD volumes; however, defining more than one region on a volume is possible. The following parameters are required for a region definition:

- Region ID (RegionId)

  An identifier for a region, which is used in a group definition (if the region forms part of a group)
- Volume serial (VolSer)

  The label of the volume on which the region exists
- Region extent (RegStart and RegEnd)

  The region start and end values define the start and end points of the region, given in whatever is the allocation unit for the type of disk device (cylinders for CKD, blocks for FBA). The special keywords START and END can be used if the allocation either starts at the beginning or finishes at the end (or both) of the volume.
- Device type (Dev-Type)

  The type of DASD on which the region is present. Note that this *must* correspond with the physical disk device that the volume lives on. The types are predefined to DirMaint.

The next section, GROUPS, allows administrators to treat a set of regions as a single pool of disk space when allocating minidisks. For example, you can create a group called SYSMDG for minidisks that are related to system user IDs, or a group called LINMDG for minidisks that are related to virtual machines for Linux. This grouping would allow you to separate user minidisks from system-related minidisks. Groups can be defined to reflect your security label configuration, ensuring that minidisks for *secure* machines are not allocated on volumes that also contain *non-secure* minidisks.

Updating the EXTENT CONTROL file

Although the `EXTENT CONTROL` file can be updated by editing the file directly on DirMaint's disk, DirMaint has to be shut down. A more convenient way is to edit the file locally in your own administrator ID by using the DirMaint SEND and FILE commands.

Note: The method of editing `EXTENT CONTROL` directly on DirMaint disk is described in *Getting Started with Linux on System z*, SC24-6096.

The following steps show an example of updating the `EXTENT CONTROL` file. In this example, four additional DASD volumes (two emulated FBA DASDs, labelled LXBC00 and LXBC01; and two ECKD volumes, labelled LXD0F0 and LXD0F1) are being made available for creating minidisks. New groups are also being created for allocations onto these volumes.

1. Have DirMaint send you a copy of the current `EXTENT CONTROL` file; see Example 2-26.

Example 2-26 Using DIRM SEND to obtain the EXTENT CONTROL file

```
dirm send extent control
DVHXMT1191I Your SEND request has been sent for processing.
Ready; T=0.02/0.02 17:01:04
 DVHREQ2288I Your SEND request for VIC at * has been accepted.
RDR FILE 0009 SENT FROM DIRMAINT PUN WAS 0619 RECS 0036 CPY  001 A NOHOLD NOKEEP
 DVHREQ2289I Your SEND request for VIC at * has completed; with RC = 0.
```

2. Receive the file from your reader; see Example 2-27.

Example 2-27 Receiving EXTENT CONTROL for editing

```
receive 9
File EXTENT CONTROL A1 created from EXTENT CONTROL E1 received from DIRMAINT at
VMLINUX9
Ready; T=0.01/0.01 17:04:43
```

3. Edit the `EXTENT CONTROL` file on your A-disk:

```
xedit extent control a
```

4. Find the section labelled `:REGIONS`. From the XEDIT command line, type the following command and press the Enter key:

```
====> /:regions
```

5. Move the cursor to the prefix area for the line containing `*RegionID`. Type the letter `i` in the prefix area and press the Enter key. See Example 2-28.

Example 2-28 RegionID line

```
00034 * *****************************************************
00035 :REGIONS.
i       *RegionId     VolSer     RegStart   RegEnd     Dev-Type
```

This step inserts a line below the headings.

6. Type information, as follows, into RegionId, VolSer, RegStart, RegEnd, and Dev-Type:

```
RLBC00 LXBC00 START END 9336
RLBC01 LXBC01 START END 9336
RLD0F0 LXD0F0 START 10000 3390-9
RAD0F0 LXD0F0 10001 END 3390-9
RLD0F1 LXD0F1 START 10000 3390-9
RAD0F1 LXD0F1 10001 END 3390-9
```

The regions starting with RA are used to ensure the maximum usage of the volumes. A 3390 Model 9 has 10017 usable cylinders, and because we usually just allocate minidisks with whole-number cylinder counts, those last 16 cylinders are likely never to get used. By using a different region for those last few cylinders, we can use that space for small minidisks.

7. Move the cursor to the prefix area for the line containing `*GroupName`, then type `i2` in the prefix area, and add these lines:

```
GLFBA1D (ALLOCATE ROTATING)
GLFBA1D RLBC00 RLBC01
GLCKD1A RAD0F0 RAD0F1
GLCKD1S (ALLOCATE ROTATING)
GLCKD1S RLD0F0 RLD0F1
```

> **Note:** The keywords `ALLOCATE ROTATING` in these group definitions instruct DirMaint to make subsequent minidisk allocations by rotating through the available regions in the group. In this example, for the group `GLFBA1D`, a minidisk will be allocated onto LXBC00, the next on LXBC01, and so on. If `ALLOCATE ROTATING` is not specified, shown for the `GLCKD1A` group, minidisks will be allocated on the first region in the group until there is no available space left in the region, after which the next region will be used.

8. Save the file: from the XEDIT command line, type this command and press the Enter key:

```
====> file
```

9. Send the edited control file back to DirMaint; see Example 2-29.

Example 2-29 Sending the EXTENT CONTROL file to DirMaint

```
dirm file extent control a
PUN FILE 0010 SENT TO   DIRMAINT RDR AS  0621 RECS 0040 CPY  001 0 NOHOLD NOKEEP
DVHXMT1191I Your FILE request has been sent for processing.
Ready; T=0.01/0.02 17:09:40
 DVHREQ2288I Your FILE request for VIC at * has been accepted.
 DVHRCV3821I File EXTENT CONTROL E1 has been received; RC = 0.
 DVHREQ2289I Your FILE request for VIC at * has completed; with RC = 0.
```

10. If the changes are required immediately, signal DirMaint to reload the edited control file; see Example 2-30.

Example 2-30 Reloading DirMaint's extent configuration

```
dirm rldextn
DVHXMT1191I Your RLDEXTN request has been sent for processing.
Ready; T=0.01/0.02 17:11:10
 DVHREQ2288I Your RLDEXTN request for VIC at * has been accepted.
 DVHILZ3510I Starting DVHINITL with directory: USER DIRECT E
 DVHILZ3510I DVHINITL Parms: BLDMONO BLDDASD BLDLINK
 DVHIZD3528W One or more DASD volume control files (LX9PG1) were
 DVHIZD3528W created using default values for device characteristics -
 DVHIZD3528W $PAGE$ 0A03
 DVHREQ2289I Your RLDEXTN request for VIC at * has completed; with RC =
 DVHREQ2289I 0.
```

Directory prototypes

DirMaint allows user prototypes to be set up, which enables you to more easily add user definitions. A prototype directory file is similar to a profile entry in the user directory, but can include all the statements that are required to define a new guest.

Note: DirMaint provides samples of directory profiles and prototype files. You can get the LINDFLT and LINUX samples from DirMaint by using these DirMaint commands:

```
DIRM SEND LINDFLT DIRECT
DIRM SEND LINUX PROTODIR
```

A prototype is added to DirMaint by creating a file that defines it and then adding that file to DirMaint. Example 2-31 shows a prototype file.

Example 2-31 Example prototype definition LXEXA PROTODIR

```
USER LXEXA NOLOG
INCLUDE LINDFLT
MDISK 191 3390 AUTOG 0001 GLCKD1A MR
MDISK 200 3390 AUTOG 2000 GLCKD1S MR
MDISK 201 3390 AUTOG 4000 GLCKD1S MR
MDISK 2A0 9336 AUTOG 20000000 GLFBA1D MR
NICDEF 800 TYPE QDIO LAN SYSTEM VSWDEV1
```

If you have created this file on your A-disk, and you are authorized to store files on DirMaint, you would use the DirMaint command FILE to store it for use:

```
DIRM FILE LXEXA PROTODIR
```

You can then refer to this prototype definition by using the LIKE keyword on the DirMaint command to add a new user.

Prototyping an existing virtual machine

You can create a prototype file from an existing user definition. The following steps illustrate an example of this:

1. Get a copy of the directory entry for the user you want to prototype:

   ```
   DIRM GET LXSMB01 NOLOCK
   ```

2. Receive the file, using the name of the new prototype you want to create and a file type of PROTODIR:

   ```
   RECEIVE number LXSMB PROTODIR
   ```

3. Xedit the received file. Make the following changes to the file:

   – Change the name on the USER line to match the prototype name, and the user password to NOLOG.

   – Update *all* MDISK entries to AMDisk commands with the appropriate AUTOG parameters to allow DirMaint to automatically allocate minidisks when the prototype is used.

   Note: You may also use CLONEDisk instead of AMDisk, which would perform a copy of existing minidisks rather than simply allocating space. For an introduction, refer to “Using DirMaint to CLONEDISK” on page 48.

4. Lodge the file to DirMaint.

   ```
   DIRM FILE LXSMB PROTODIR
   ```

You can now use DIRM ADD LIKE to create virtual machines with the same resource definitions as your existing virtual machine.

2.7.3 Using DirMaint

This section discusses several aspects of using DirMaint to manage user definitions in z/VM.

Note: Refer to the z/VM Directory Maintenance Facility documentation for detailed information about using DirMaint.

Important commands in DirMaint

DirMaint has a corresponding command for every z/VM directory statement. Command authorization is controlled by assigning DirMaint commands to privileged command sets. Example 2-32 lists some important commands.

Example 2-32 Sample of important commands in DirMaint

```
Add - Add a new user or profile directory entry
REView - Review a user or profile directory entry
AMDisk - Adds a new minidisk
CLAss - Change the CP class for a directory entry
DEDicate - Add or delete an existing dedicate statements
DMDisk - Removes a minidisk
LOGONBY - Allows users to use their own password to logon to different IDs
FILE - Add or replace a DirMaint control file
MDisk - Change the access mode and passwords for minidisks
PURGE - To remove the entry for a userID
RLDCode - Cause DirMaint to reload its resident operating procedures
RLDExtn - Cause DirMaint to reload its CONFIG* DATADVH file
SEND - Request a copy of a DirMaint control file
STorage - Change logon storage size
SETOptn - Add, change or delete CP options
SPEcial - Add or delete an existing special statement
TMDisk - Transfer ownership of a minidisk from one userid to another
```

The command DIRM HELP allows you to find more information about any of the DirMaint commands.

Creating a new guest image using LIKE

By using the setup we provided as examples in 2.7.2, “Customizing DirMaint” on page 42, adding an entry to the directory for a new user ID becomes very easy.

In this example, we are using DirMaint to define a new Linux virtual machine called LXWEB01, and using the prototype LXEXA. Type this command and then press Enter:

```
DIRM ADD LXWEB01 LIKE LXEXA PW new_password
```

Using DirMaint to CLONEDISK

We can create a new minidisk by cloning an existing minidisk. The new disk will have the same device type and size as the cloned disk, and will also contain all of the data from the existing disk. CLONEDISK can also be used to *reset* an existing minidisk to be identical to another disk of the same type and size.

DirMaint's CLONEDISK command performs the following operations when copying a minidisk:

1. If a new destination minidisk is to be created, DirMaint uses the allocation parameters that are provided with the command to create a new minidisk in the destination user ID. The size of the disk to be created is identical to the size of the source disk.
2. DirMaint attaches the source and destination minidisks to the DATAMOVE user. If a stable access cannot be obtained to the destination, DirMaint abandons the request.
3. DATAMOVE performs the copy operation, using either FlashCopy® or DASD Dump Restore (DDR), according to the system default.
4. The source and destination minidisks are detached from DATAMOVE.

Note: The disk specified as the source for a CLONEDISK operation should not be in use. Especially for Linux file systems, the copy might not be valid if the disk is being used by an active virtual machine.

The syntax of the DirMaint CLONEDISK command is shown in Example 2-33.

Example 2-33 Sample CLONEDISK command

```
>--DIRMaint--.--------------------.--CLONEDisk--vaddr--source_owner--------->
              |                (1)|
              '-Prefix Keywords----'

>--source_addr--.--------------------------------.------------------------><
                '-| MDISK Allocation Parameters |-'

MDISK Allocation Parameters:
|--.-startloc--volid-------.----------------------------------------------->
   |---AUTOG----groupname--|
   |---AUTOR----regionname-|
   |---AUTOV----volid------|
   '-DEVNO--raddr----------'

                         (3)
>----mode--.-----------.------.----------------.----------------------------|
           |        (2)|      '-| PW Options |-'
           '-suffix----'

PW Options:
|--PWs--readpass--.-------------------------.------------------------------|
                  '-writepass--.-----------.-'
                               '-multipass-'
```

An example of cloning a minidisk using DirMaint CLONEDISK is shown in Example 2-34.

Example 2-34 CLONEDISK

```
DIRM FOR LXWEB01 CLONEDISK 301 LNXORG 201
```

Set password using DirMaint

The SETPW operand of the DirMaint command is used to change the user logon password. See Example 2-35. When entering the SETPW command password or passphrase on the VM command line, note that a passphrase containing embedded blanks must be surrounded by single quotation marks. The password or passphrase does not appear in log files maintained by DirMaint.

Example 2-35 Sample of the SETpw command

```
>>--DIRMaint--.--------------------.--SETpw--password--.---------------.----->
              |                 (1)|                   '-VPW--verifypw-'
              '-Prefix Keywords----'

>--.-----------.-----------------------------------------------------------><
   '-nnn--DAYS-'
```

> **Note:** Only the password in the user directory is updated with this command. If an ESM is in place and active, the ESM will be the first point of verification for logon passwords.

Example 2-36 shows an example of using SETPW.

Example 2-36 DirMaint SETpw command

```
dirm for lnxrh1 setpw linuxrh1 vpw linuxrh1 30 days
```

2.8 Other ESM and directory manager security observations in this book

Chapter 8, “Best practices” on page 265 presents points to consider when securing your environment, and how either an ESM, a directory manager, or both, can help you. The recommendations in this chapter cover resource security management, including disks and network connections.

In addition, 6.11, “Protecting z/VM minidisks” on page 237 discusses an aspect of data management and how an ESM and directory management can assist in securing your data.

3 Configuring and using the System z LDAP servers

In this chapter, we introduce the z/VM and z/OS LDAP servers. We show how they interact with other parts of the host operating system (such as RACF) and present how they can be used to provide standard LDAP services to Linux guests.

We also introduce high availability topics for LDAP, and demonstrate how the z/VM LDAP server can be used in conjunction with a Linux-based OpenLDAP server in an SSI cluster to provide high availability for the LDAP service.

We focus on the use and operation of the z/VM LDAP server, but illustrate differences between the server running in z/VM and z/OS.

3.1 The z/VM and z/OS LDAP servers

With z/VM Version 5 Release 3, the LDAP server was introduced into z/VM. The server was included from z/OS Security Server Version 1 Release 8, using a new technology in z/VM that allows z/OS binaries to run in a z/VM virtual machine.

With each z/VM release since Version 5 Release 3, the LDAP server has been upgraded to the corresponding z/OS Security Server level; in z/VM 6.2 it was updated to the z/OS Security Server V1R12 level.

3.1.1 z/VM LDAP server backends

The z/VM LDAP server can operate concurrently with multiple database instances, referred to as backends. It supports a variety of backend types.

RACF backend (SDBM)

The SDBM backend uses RACF as its backing store. All data presented by SDBM comes from the RACF database. The data in the SDBM backend is organized using the RACF LDAP schema.

Because RACF is used directly as the backing store for SDBM, full auditing of all operations is automatically managed according to your RACF configuration.

Byte File System backend (LDBM)

The LDBM backend stores the LDAP data in a directory in the z/VM Byte File System (BFS).

The schema used by LDBM can be extended, allowing other applications that use LDAP to be supported by the z/VM LDAP server. Refer to 3.4, “Extending the LDBM schema” on page 59 for information about extending the LDAP schema used by LDBM.

DB2 backend (TDBM)

Available in the z/OS LDAP server only, the TDBM backend provides equivalent function to LDBM. TDBM uses a DB2® subsystem for the LDAP storage instead of the BFS used by LDBM, yielding a higher level of performance. If the size of the LDAP database is expected to be larger than 500,000 records, TDBM is recommended over LDBM for better performance.

Logging backend (GDBM)

To provide a similar level of auditing and logging capability as SDBM, a separate backend can be configured to log changes to LDBM. This logging backend is configured separately to LDBM, but is updated each time a change is made to a record in LDBM.

GDBM appears as a standard LDAP database with a schema similar to that of LDBM. Normal LDAP search operations are supported, allowing administrators to easily search for details of updates made to particular records or changes made by a particular administrator.

Configuration backend (CDBM)

New in the z/VM LDAP Server with z/VM 6.2, the CDBM backend provides an LDAP interface to configuration information. This database is especially useful in clustered setups, allowing configuration changes in the master LDAP to be automatically copied to the other servers in the cluster using normal LDAP replication.

Audit backend (ICTX)

The z/VM LDAP server can provide an internal interface to RACF that is used for recording audit information from the Linux audit daemon. The ICTX backend is actually a plug-in to the LDAP server that provides the interface to which the Linux audit daemon connects. It is not an actual LDAP database as such.

Note: More information about ICTX and the RACF audit capability can be found in “Centralizing Linux audit information with z/VM RACF” on page 92.

3.1.2 The relationship between the LDAP servers and RACF

The z/VM LDAP server is a separate component from RACF, and runs in its own service virtual machine (SVM). The LDAP server does not require RACF (or another ESM) to be present in order to function. However, some functions (such as native authentication with LDBM) are not available without RACF.

On z/OS, the LDAP server is a started task separate from RACF. Since it is unlikely that a z/OS system would run without an ESM, the ESM-related functions are available in almost every installation. They are not compulsory, however, and it is possible to configure a z/OS LDAP server without using the ESM-related features.

3.2 Setting up the z/OS LDAP server

The z/OS LDAP server includes a configuration utility that generates the required files for operating the server. We used this utility, called **dsconfig**, to create a configuration for our z/OS LDAP server.

3.2.1 Using dsconfig

To use the **dsconfig** utility, we created a setup file that defined the environment we expected to operate our LDAP server in. Key parts of our configuration file are shown in Example 3-1. We configured our z/OS LDAP server with LDBM and SDBM backends, without the TDBM backend, and to run with a started task name of LDAPSRV.

Example 3-1 Portion of the **dsconfig** *configuration file ds.config*

```
# ***********************************************************************
# This is the main input file for the configuration utility for the LDAP
# server.
#
# Refer to publication:
#   IBM Tivoli Directory Server Administration and Use for z/OS
#   (SC23-5191)
# ***********************************************************************
#------------------------------------------------------------------------
# ADMINDN <distinguished name>
#
ADMINDN = "cn=Admin,dc=itso,dc=ibm,dc=com"
# -----------------------------------------------------------------------
# ADMINPW <password>
ADMINPW = secret
# -----------------------------------------------------------------------
# LDBM_SUFFIX <suffix>
LDBM_SUFFIX = 'ou=poughkeepsie_L,dc=itso,dc=ibm,dc=com"
# -------------------------------------------------------------------------
# LDBM_DATABASEDIRECTORY <directory>
LDBM_DATABASEDIRECTORY = /var/ldap/ldbm
# -------------------------------------------------------------------------
# SDBM_SUFFIX <suffix>
SDBM_SUFFIX = "ou=poughkeepsie_S,dc=itso,dc=ibm,dc=com"
# -------------------------------------------------------------------------
# TDBM_SUFFIX <suffix>
TDBM_SUFFIX =
# -------------------------------------------------------------------------
# SCHEMAPATH <directory>
SCHEMAPATH = /var/ldap/schema
# -------------------------------------------------------------------------
# PROG_SUFFIX <suffix>
PROG_SUFFIX = LO
# -------------------------------------------------------------------------
# LDAPUSRID <user_id>
LDAPUSRID = LDAPSRV
# -------------------------------------------------------------------------
# OUTPUT_DATASET <data_set_name>
OUTPUT_DATASET = VIC.GLD.CNFOUT
...
```

Example 3-2 shows the execution of the **dsconfig** utility.

Example 3-2 Running the dsconfig utility

```
VIC @ SC60:/u/vic>export STEPLIB=SYS1.SIEALNKE
VIC @ SC60:/u/vic>export PATH=/usr/lpp/ldap/sbin:$PATH
VIC @ SC60:/u/vic>export NLSPATH=/usr/lpp/ldap/lib/nls/msg/%L/%N:$NLSPATH
VIC @ SC60:/u/vic>export LANG=En_US.IBM-1047
VIC @ SC60:/u/vic>dsconfig -i ds.profile
120830 11:46:22.612762 GLD2002I Directory Server configuration utility has start
ed.
120830 11:46:24.291679 GLD2003I Directory Server configuration utility has ended
.
VIC @ SC60:/u/vic>
```

The **dsconfig** utility allocates the partitioned data set (PDS) named in the `OUTPUT_DATASET` statement, with members that are used to create the environment for running the z/OS LDAP server. The **dsconfig** utility generates files containing APF authorization statements, RACF updates, and the JCL to run to make the changes. The configuration output also includes the LDAP server configuration file and environment file, and the started task JCL member that correctly starts the LDAP server to use those configuration and environment files.

Note: Changes to the z/OS TCP/IP configuration may also be required to run the LDAP server. See the TCPIP PROFILE changes for the z/VM LDAP server in Example 3-3 on page 55 for indicative changes.

The full method of enabling the z/OS LDAP server is described in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191.

3.3 Setting up the z/VM LDAP server

The manuals *z/VM TCP/IP Planning and Customization (version 6 release 2)*, SC24-6238-02, and *z/VM TCP/IP LDAP Administration Guide (version 6 release 2)*, SC24-6236-01, provide much information about enabling and configuring the z/VM LDAP server. The publication *Security on z/VM*, SG24-7471, also discusses the setup of the LDAP server on z/VM.

For our environment, we used the following configuration:

- SSL/TLS, enabled for secure connection
- LDBM with Native Authentication
- GDBM
- SDBM

In this section we investigate how we arrived at our desired configuration.

3.3.1 Activating the z/VM LDAP server

To activate the z/VM LDAP server, we followed the steps described in *z/VM TCP/IP Planning and Customization (version 6 release 2)*, SC24-6238-02. In summary, we updated the `AUTOLOG` and `PORT` statements in the TCP/IP PROFILE file, and updated the `DS CONF` and `DS ENVVARS` files for the server attributes we required.

Note: We created `DS CONF` and `DS ENVVARS` from the samples provided with TCP/IP. If you are not familiar with the configuration options available, use the samples and review the comments contained there.

Example 3-3 shows the changes to `PROFILE TCPIP`.

Example 3-3 Changes to PROFILE TCPIP

```
PORT
   ...
   389 TCP LDAPSRV            ; LDAP Server
   636 TCP LDAPSRV            ; LDAP Server
...
AUTOLOG
   ...
   LDAPSRV 0
   ...
ENDAUTOLOG
```

Note: The z/VM LDAP server does not use the system SSL server to provide SSL/TLS support. When reserving the secure LDAP port (636), the SECURE keyword on the PORT statement should not be used.

Some examples show the use of the `NOAUTOLOG` keyword on the reservation for port 636. This is done to ensure that TCP/IP does not try to restart LDAPSRV if the dedicated SSL port is not configured in the LDAP server. In order to support as many possible clients, we believe configuring port 636 for SSL connections is good practice. However, if you will not be using port 636 (that is, all your secure clients will connect using TLS via the standard port) then you will need to add `NOAUTOLOG` to the port reservation for port 636.

Example 3-4 shows the statements we provided in DS CONF.

Example 3-4 Statements from DS CONF

```
adminDn "cn=Admin,dc=itso,dc=ibm,dc=com"
adminPW secret
allowAnonymousBinds on
audit on
audit all,add+delete
audit error,modify+search+bind
audit none,unbind
commThreads 10
listen ldap://:389
listen ldaps://:636
maxConnections 4096
sendV3StringsOverV2As UTF-8
sizeLimit 500
timeLimit 3600
validateIncomingV2strings on

sslAuth serverAuth
sslCertificate LDAPSRV
sslKeyRingFile /etc/ldap/key.kdb
sslKeyRingPWStashFile /etc/ldap/key.sth

database LDBM GLDBLD31
suffix "ou=cambridge_L,dc=itso,dc=ibm,dc=com"

database GDBM GLDBGD31
changeLogging on
```

Note: As you can see, we specified `adminPW` in our configuration file. This is not recommended as a way to run your live LDAP server. Do this only to get your database initially configured. Set `adminDN` to the name of a DN that appears in one of your backends (either a valid RACF ID if using SDBM, or a DN that you add to LDBM) so that proper password checking can be done for admin access to the LDAP server. When you have an ID available in your database, remove the `adminPW` statement from your DS CONF file.

The statements we entered in `DS ENVVARS` are shown in Example 3-5.

Example 3-5 Statements from DS ENVVARS

```
NLSPATH=/usr/lib/nls/msg/%L/%N
LANG=En_US.IBM-1047
TZ=EST5EDT
```

Enabling SSL support for LDAP connections

The LDAP server uses its own internal SSL engine for providing secure connectivity for clients. As noted earlier, this means that the System SSL service is not used, and the TCP/IP port definition for the LDAP server must not be defined to use System SSL.

The keywords in `DS CONF` related to SSL are shown in Example 3-6.

Example 3-6 SSL keywords in DS CONF

```
sslAuth serverAuth
```

```
sslCertificate LDAPSRV
sslKeyRingFile /etc/ldap/key.kdb
sslKeyRingPWStashFile /etc/ldap/key.sth
```

These keywords are explained here:

sslAuth	The type of SSL session to be used. “serverAuth” means that only the certificate of the server is to be verified when SSL is brought up. “clientAuth” means that the client must have a valid certificate, and this certificate is also verified before the SSL connection is enabled.
sslCertificate	The label of the SSL certificate within the certificate database that the LDAP server will present as its server certificate.
sslKeyRingFile	The BFS location of the certificate database containing the LDAP server’s certificate.
sslKeyRingPWStashFile	The saved password for the certificate database, used by the LDAP server to unlock the database.

Providing the server certificate to LDAPSRV

Since the LDAP server does not use the System SSL services, we believe creating a separate database for the LDAP server provides additional security for SSL certificates in the following ways:

- The LDAP server’s internal SSL engine will not have access to certificates used for servers managed through the System SSL services (such as TN3270).
- System SSL will not be able to access the LDAP server certificate.

Following the process described in “Implementing a certification authority in z/VM” on page 17, we used **`gskkyman`** to create a new database for the LDAP server. We used this database to generate a certificate request, export and sign the request, then import the signed certificate back to the LDAP server certificate database.

Note: As you can see in our configuration files shown, we used the filename `/etc/ldap/key.kdb` for the LDAP server certificate database. We had to create the `ldap` directory under `/etc/` to do this. Since `/etc` is in the system root BFS, which is the same file system mounted by default by GSKADMIN, we did not have to perform any other BFS setup to build this key database.

The LDAP server does have its own BFS space, mounted at `/var/ldap`. This BFS is used mainly for logging. You can create your certificate database under `/var/ldap`, and this will have a slight advantage in protecting the LDAP server’s certificate database from other users on the system. However, since GSKADMIN does not have access to this BFS by default, you will need to take additional steps to get the files into that directory.

Depending on the directories you use and their respective access controls, you may not be able to access both BFS file systems from the same user ID. If this is the case, you will need to use OPENVM GETBFS to copy the BFS files to a CMS file system, and then use a CMS mechanism (for example SENDFILE, SFS, or FTP) to transfer the files between user IDs. Once the CMS files are at the user ID with access to the LDAPSRV BFS file system, you can use OPENVM PUTBFS to copy them into the LDAPSRV user’s BFS.

Note: Remember to take care to preserve the file format when copying the files between file systems using CMS and network utilities! Utilities like SENDFILE and FTP have options for treating files as binary or ASCII, and using unnecessary translations may corrupt the contents. Use a BINARY mode to transfer these certificate store files.

This is different from transferring the certificate files as discussed in “Exporting CA certificate as a file” on page 18, because those files are text files and have to be transferred in text or ASCII mode.

Activating RACF support in the LDAP server

At the time we first set up the LDAP server, RACF was not enabled on our system. This meant that when RACF became available, we had to update part of the configuration.

We made the following changes to switch to a RACF-enabled configuration:

- Set the **`.ESM_Enable`** tag in the `SYSTEM DTCPARMS` file.
- Added the listener for the RACF Program Call interface to DS CONF.
- Added the SDBM backend to DS CONF.
- Permitted the LDAP server user to connect to RACF.

DTCPARMS update

The update we made to SYSTEM DTCPARMS is shown in Example 3-7.

Example 3-7 DTCPARMS tag for RACF activation of the LDAP server

```
:nick.LDAPSRV   :type.server  :class.ldap
                :ESM_Enable.YES
```

LDAP server configuration changes

Example 3-8 shows the updates made to DS CONF to enable the RACF functions of the LDAP server (the changes are in bold).

Example 3-8 DS CONF changes for RACF enablement

```
adminDn "cn=Admin,dc=itso,dc=ibm,dc=com"
allowAnonymousBinds on
audit on
audit all,add+delete
audit error,modify+search+bind
audit none,unbind
commThreads 10
listen ldap://:389
listen ldaps://:636
listen ldap://:pc
maxConnections 4096
sendV3StringsOverV2As UTF-8
sizeLimit 500
timeLimit 3600
validateIncomingV2strings on

sslAuth serverAuth
sslCertificate LDAPSRV
sslKeyRingFile /etc/ldap/key.kdb
sslKeyRingPWStashFile /etc/ldap/key.sth
```

```
database LDBM GLDBLD31
suffix "ou=cambridge_L,dc=itso,dc=ibm,dc=com"
nativeAuthSubtree all
nativeUpdateAllowed on
useNativeAuth all

database SDBM GLDBSD31
suffix "ou=cambridge_S,dc=itso,dc=ibm,dc=com"
```

LDAP server RACROUTE authority

In order for the LDAP server to connect to RACF and issue RACROUTE requests, we had to give authority to the user that was running the LDAP server (LDAPSRV). From a RACF administrator user, we issued the following command to grant the required access, which is described in *z/VM TCP/IP Planning and Customization (version 6 release 2)*, SC24-6238-02:

```
RAC PERMIT ICHCONN CLASS(FACILITY) ID(LDAPSRV) ACC(UPDATE)
```

Note: You might have to create the ICHCONN resource in the FACILITY class prior to issuing this command. Also, if the FACILITY class is RACLISTed, you will have to refresh the class for this command to take effect. For more information about RACROUTE authorization, refer to the “Authorization to Issue RACROUTE Requests” topic in *z/VM: Security Server RACROUTE Macro Reference*, SC24-6150.

3.3.2 Adding schema supplied by IBM to LDBM

To use the LDBM backend, the definitions of all valid objects and attributes that can be contained in the database must be added to the server. These definitions are contained in the database *schema*. All LDAP databases have a schema, but for some databases (such as SDBM) the schema is predefined.

The schema used by LDBM must be populated before the database can be used. IBM provides two sample schema files that create definitions for many common objects and attributes found in an LDAP database. We followed the instructions in *z/VM TCP/IP Planning and Customization (version 6 release 2)*, SC24-6238-02 for using the sample schema files (supplied by IBM) to define the schema for our LDBM backend. The process is also described in *Security on z/VM*, SG24-7471.

The same instructions applied to the z/OS LDAP server. On a z/OS system, the files are found in the `/usr/lpp/ldap/etc` directory as `schema.user.ldif` and `schema.IBM.ldif`. However, we found that the files from the z/VM system worked the same, and we could run the **`ldapadd`** command from z/VM to add the schema files to the z/OS LDAP server.

3.4 Extending the LDBM schema

One of the interesting capabilities offered by the LDBM backend of the z/VM LDAP server is the ability to extend its schema. Extending allows the server to provide support for a wider variety of objects and attributes than SDBM, and also enables the z/VM LDAP server to provide LDAP database support to applications that have to work with a fixed database schema.

Full information about the schema support of the z/VM LDAP server can be found in the “LDAP directory schema” chapter of *z/VM TCP/IP LDAP Administration Guide (version 6* *release 2)*, SC24-6236-01. For the z/OS LDAP server, information is found in the "LDAP directory schema" chapter of the manual *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191.

3.4.1 LDAP schema dependencies for Linux

Basic support for LDAP in Linux is provided by modules added to these system components:

- The Name Server Switch (NSS), a component of the GNU C Library
- Pluggable Authentication Module (PAM)
- System Security Services Daemon (SSSD), a new security subsystem used in RHEL 6

The nss_ldap set of extensions provides LDAP lookup capabilities for many of the databases provided by NSS, including passwd, protocols, hosts, and others; pam_ldap provides a module that can be added to the PAM configuration to provide LDAP authentication for PAM-enabled applications.

Configuring nss_ldap and pam_ldap

The configuration of nss_ldap and pam_ldap is usually consolidated into a single `/etc/ldap.conf` file. Among other configuration settings that are controlled there, the `ldap.conf` file provides a way for the default LDAP attributes used by nss_ldap and pam_ldap to be changed. Example 3-9 lists some of the default values from the `ldap.conf` file on our Linux system.

Example 3-9 Default NSS and PAM configuration values from /etc/ldap.conf

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

# The user ID attribute (defaults to uid)
#pam_login_attribute uid

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

# Enable support for RFC2307bis (distinguished names in group
# members)
nss_schema      rfc2307bis

# attribute/objectclass mapping
# Syntax:
#nss_map_attribute      rfc2307attribute       mapped_attribute
#nss_map_objectclass    rfc2307objectclass     mapped_objectclass
```

Both nss_ldap and pam_ldap assume that the LDAP schema in use is compatible with the definition contained in RFC 2307. The last two lines of Example 3-9 show how the `nss_map_attribute` and `nss_map_objectclass` statements would be used to change the default mappings of object types and attributes from what the modules expect to what the LDAP server actually provides. Example 3-10 shows another portion of the `ldap.conf` file, this time showing how it would be set up for the schema used on the Microsoft Active Directory database.

Example 3-10 The ldap.conf file for configuration with MS Active Directory

```
# AD mappings
nss_map_objectclass posixAccount user
nss_map_objectclass shadowAccount user
nss_map_attribute uid sAMAccountName
nss_map_attribute homeDirectory unixHomeDirectory
nss_map_attribute shadowLastChange pwdLastSet
nss_map_objectclass posixGroup group
nss_map_attribute uniqueMember member
pam_login_attribute sAMAccountName
pam_filter objectclass=User
pam_password ad
```

Deciding to change the database schema or ldap.conf

The two options to choose from when using pam_ldap and nss_ldap with a *nonstandard* LDAP server are:

- Modify the `ldap.conf` file to change the Linux `objectclass` and `attributetype` mappings.
- Update the schema of the LDAP server to allow Linux to work unmodified.

There is no recommendation either way. Perhaps changing a small number of LDAP servers is easier than a large number of Linux virtual machines. However, if the change to `ldap.conf` is built into the *standard build* of your Linux virtual machines, the actual amount of work that is required to change them is minimal.

One factor to consider is the potentially large number of Linux applications that you might want to have use LDAP at some time (so far we have only discussed basic Linux authentication), and each of these is likely to have its own configuration. For example, the Apache Web server provides its own support for LDAP authentication (with mod_authnz_ldap extension), and even then application programming environments such as PHP and Perl running under Apache using mod_php or mod_perl have their own LDAP functionality. Having to configure each of these individually and map schemas separately could become difficult to maintain.

Because of the greater flexibility that can be obtained, we would tend to believe that extending the z/VM LDAP server schema is a better approach. However, this is not a firm suggestion, and you should consider your mix of server and application before you decide which approach to use.

3.4.2 Extending the schema of the z/VM LDAP server

If you decide to extend the schema of the z/VM LDAP server, then this section is for you.

Obtaining schema definitions

OpenLDAP is a project that develops Open Source LDAP applications and development tools. The project produces the OpenLDAP Suite, which includes the most common LDAP server implementation found on Linux systems, **`slapd`**.

Information: The Web site for the OpenLDAP Suite is:

http://www.openldap.org/

OpenLDAP includes a number of schema files that implement the directory objects described in many Internet RFCs. Also, products such as pam_ldap, nss_ldap, and Samba that rely on or support LDAP databases are usually packaged on Linux with schema files that are designed for OpenLDAP.

The z/VM LDAP server provides an interface to extend the LDBM schema, but it is missing certain syntax definitions and matching rules that OpenLDAP provides (and that many schemas use).

Converting OpenLDAP schema files for the z/VM LDAP server

The definition of the schema in the z/VM LDAP server LDBM backend is contained in the database itself under a separate Directory Information Tree (DIT) called `cn=schema`. The schema DIT is updated with the **`ldapmodify`** utility, using source contained in an LDAP Data Interchange Format (LDIF) file. The LDIF file is read only once, because the **`ldapmodify`** operation reads the data from the LDIF file into the LDBM database.

Many LDAP servers, such as OpenLDAP version 2.3 and higher and the Fedora Directory Server, use LDIF files for schemas. A number of scripts on the Internet perform translation of files in the older OpenLDAP schema format to LDIF for various LDAP servers. None of these scripts, however, support the automatic management of the syntax definitions and matching rules missing from the various servers.

Note: The DIT that is used for the schema in OpenLDAP differs from that used for the z/VM LDAP server, so taking an OpenLDAP schema LDIF file and applying it to the z/VM LDAP server directly is not possible. In addition, the missing syntax definitions and matching rules also prevent OpenLDAP schema LDIF files from being used with the z/VM LDAP server.

In general, we found that the process of converting OpenLDAP schema files can require some effort. We converted two schema files in OpenLDAP format to see how the process works:

- The NIS schema, which provides the object classes and attributes usually used by UNIX and Linux systems for authentication.
- The **inetOrgPerson** schema (usually supplied with OpenLDAP) is described in RFC 2798 and extends the basic **person** and **account** objects from previous RFCs.

Translating OpenLDAP LDIF schema to z/VM LDIF

Even though schema files in OpenLDAP's LDIF cannot be directly applied to the z/VM LDAP server, we tried converting the format to that which is acceptable to z/VM LDAP. Table 3-1 shows the differences we found between the OpenLDAP LDIF schema and the z/VM LDAP LDIF schema.

Table 3-1 LDIF differences between OpenLDAP OLC format and z/VM LDAP format

	OpenLDAP OLC	**z/VM LDAP**
DN of schema updates	cn=schema,cn=config	cn=schema
Name of object class for attribute definitions	olcAttributeTypes	attributeTypes
Name of object class for object class definitions	olcObjectClasses	objectClasses

We describe the process we used in the following steps:

1. Replace the opening LDIF syntax with corrected syntax (adding a new schema object to the “cn=schema,cn=config” tree is changed to adding attributes to the “cn=schema” object);
2. Search and replace “olcAttributeTypes” with “attributeTypes”;
3. Search and replace “olcObjectClasses” with “objectClasses”;
4. Insert “add:” lines at appropriate points to reflect the type of attribute (either **`attributeTypes`** or **`objectClasses`**) being added in the next section of the file. Ideally you would group all the new attributeTypes attributes and commence that block with an “add: attributeTypes” line, followed by the objectClasses attributes commencing with an “add: objectTypes” line. However, there might be dependencies in the schema definitions that require the new attributes be mixed; if your file requires this, you must have a new “add” line when you change from adding one attribute type to another.

Once you have completed these steps, you have a file that should update the schema of the z/VM LDAP server as needed. However, the schema may be using LDAP syntaxes or matching rules that are not present in the z/VM LDAP server.

In converting the NIS schema for z/VM LDAP, we found that most of the prerequisite attribute and object definitions were supplied in the IBM schema files provided with the z/VM LDAP server (which we describe in “Adding schema supplied by IBM to LDBM” on page 59). A couple of definitions missing from the base IBM schema prevent the NIS schema from being loaded, however, so we had to find the missing schema components and add them. We used phpLDAPadmin to display the schema loaded in the z/VM LDAP server, and compared that with the “core” and “cosine” schema listed as the dependencies of the NIS schema. We used those results to create an LDIF file with the differences to be applied, which is shown in Example 3-11.

Example 3-11 LDIF for additional schema definitions from “core” and “cosine” schema

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.5.4.65 NAME ‘pseudonym’
  DESC ‘X.520(4th): pseudonym for the object’
  SUP name )
-
add: objectClasses
objectClasses: ( 0.9.2342.19200300.100.4.19 NAME ‘simpleSecurityObject’
  DESC ‘RFC1274: simple security object’
  SUP top AUXILIARY
  MUST userPassword )
objectClasses: ( 1.3.6.1.1.3.1 NAME ‘uidObject’
  DESC ‘RFC2377: uid object’
  SUP top AUXILIARY MUST uid )
```

After we applied this schema LDIF to our z/VM LDAP server, we were able to add the NIS schema.

Schema file conversion script

If you have LDAP schema that are in the old OpenLDAP schema file format, you will need to use a different method to convert them to LDIF for the z/VM LDAP server.

We searched for a script that would work to convert the OpenLDAP schema files to the correct format for the z/VM LDAP server. We did not find a script that would do the job directly, so we modified the closest script we could find. The script is shown in Example 3-12.

Example 3-12 The schema-ol2ldif.pl script

```
#!/usr/bin/perl -w
#
# this is a quick perl script to convert OpenLDAP schema files
# to ldif (schema) files.  it is probably not anywhere near
# useful, but it did allow me to convert a few of my .schema
# files and have FDS successfully start with them.
#
# Original by Nathan Benson <tuxtattoo@gmail.com>
# Modified for z/VM LDAP by Vic Cross <viccross@au1.ibm.com>
#

use strict;

die "usage: $0 <openldap.schema>\n" unless my $file = $ARGV[0];
die "$! '$file'\n" unless -e $file;

my $start;

print "dn: cn=schema\n";
print "changetype: modify\n";
print "add: attributetypes\n";
print "-\n";
print "add: objectclasses\n";

open SCHEMA, $file;
while (<SCHEMA>)
{
        next if /^(#|$)/;


        if (/^(objectclass|attributetype)\s/i)
        {
                print "\n" if ($start);
                chomp;


                $_      =~ s/^objectclass/objectclasses:/i;
                $_      =~ s/^attributetype/attributetypes:/i;
                $_      =~ s/(\t|\s)/ /;


                $start = 1;
                print;
        }
        elsif ((/^\s*\w/) && ($start))
        {
                chomp;
                $_      =~ s/^(\s*)/ /;
                $_      =~ s/IA5Substring/Substring/i;
                print;
51              }
52      }
53      close SCHEMA;
```

We found that the following rules are not defined in the z/VM LDAP server, but they can be substituted for rules that do exist:

- `caseIgnoreIA5SubstringMatch`
- `caseExactIA5SubstringMatch`

The script is invoked as follows:

```
# schema-ol2ldif.pl <schema-file> > <ldif-output-file>
```

Note: For this script to work correctly, the "`add: objectclasses`" generated in the output file by line 23 of the script should be moved to the correct location in the output file. The correct location is between the last `attributeTypes` definition and the first `objectClasses` definition.

Schema prerequisites

The NIS schema was already converted by the team who wrote *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221. For our own experience, and to validate our conversion scripts and make sure we were using the most current version of the schema, we converted the `nis.schema` file that was included with our SLES 11 SP2 server.

Most of the dependencies of the object classes from the NIS schema were already satisfied by the schemas supplied by IBM. However, two attributes required by the posixAccount object class were not present: `uidNumber` and `gidNumber`. These attributes were actually present in the source OpenLDAP schema file for the NIS schema but were commented out because they were provided by other base schemas in OpenLDAP. We added the definition of these attributes back into the schema and reran the conversion script.

Note: We found that the latest version of the IBM schema files for the z/OS and z/VM LDAP servers contain the inetOrgPerson object. While inetOrgPerson is usually associated with user records in many LDAP user administration tools (including phpLDAPadmin), it is actually the posixAccount object that contains the attributes that nss_ldap and pam_ldap need. Adding the full set of objects and attributes from the NIS schema is necessary to use the z/VM or z/OS LDAP servers to authenticate Linux users.

We also found a couple of other definitions required by the NIS schema that were provided by other schema on OpenLDAP. Example 3-13 shows the LDIF file for the objects and attribute remaining from the “core” schema that prevented us from initially applying the NIS schema. Once we added these to z/VM LDAP we were able to successfully add the NIS schema.

Example 3-13 LDIF file containing missing definitions from the “core” schema

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributetypes: ( 2.5.4.65 NAME ‘pseudonym’
 DESC ‘X.520(4th): pseudonym for the object’
 SUP name )
-
add: objectClasses
objectClasses: ( 0.9.2342.19200300.100.4.19 NAME ‘simpleSecurityObject’
 DESC ‘RFC1274: simple security object’
 SUP top AUXILIARY
 MUST userPassword )
objectClasses: ( 1.3.6.1.1.3.1 NAME ‘uidObject’
 DESC ‘RFC2377: uid object’
 SUP top AUXILIARY MUST uid )
```

Using ldapmodify to update the z/VM LDAP server schema

We did all our schema updates from a Linux server using the **ldapmodify** utility (part of the OpenLDAP Suite). Example 3-14 shows how we extended the z/OS LDAP server schema for the missing “core” schema definitions.

Example 3-14 The ldapmodify command session

```
lnxslesa:~ # ldapmodify -v -x -h wtsc60.itso.ibm.com -D “cn=Admin,dc=itso,dc=ibm,
dc=com” -w secret < core-zvm.ldif
ldap_initialize( ldap://wtsc60.itso.ibm.com )
add attributetypes:
   ( 2.5.4.65 NAME ‘pseudonym’ DESC ‘X.520(4th): pseudonym for the object’ SUP
name )
add objectclasses:
   ( 0.9.2342.19200300.100.4.19 NAME ‘simpleSecurityObject’ DESC ‘RFC1274: simple
security object’ SUP top AUXILIARY MUST userPassword )
   ( 1.3.6.1.1.3.1 NAME ‘uidObject’ DESC ‘RFC2377: uid object’ SUP top AUXILIARY
MUST uid )
modifying entry “cn=schema”
modify complete
```

3.5 LDBM and native authentication

There are dependencies between the LDAP server and RACF when native authentication is enabled. We saw an example of this in the previous section when we tried to add an object containing the `userPassword` attribute.

3.5.1 LDBM record with the userPassword attribute

The LDAP server did not allow an object containing the `userPassword` attribute to be added to LDBM. On our LDAP server, native authentication was in effect for the entire DIT.

When an LDAP object exists in a portion of the DIT for which native authentication is in effect, RACF is responsible for the `userPassword` attribute and will not allow a potentially conflicting attribute to be added to the database.

Native authentication and moving objects

If you are using native authentication for only a portion of your DIT (that is, you have sections of your directory structure where native authentication is not used), be sure to take care when objects are moved between various sections of the DIT. LDAP does not natively support moving objects; most LDAP management tools implement a move by doing sequential add and delete operations.

Therefore, the following list indicates what will occur when objects are moved between native authentication-enabled DIT branches:

- From NA-enabled to non-NA-enabled: The operation will succeed but the object in its new location will have no `userPassword` attribute. Authentication attempts will fail until a password is set for the object.
- From non-NA-enabled to NA-enabled: The operation will fail because the original object will have a `userPassword` attribute and the LDAP server will reject the add. The `userPassword` attribute would have to be deleted before the move could be performed. Authentication attempts may fail after the move, because the RACF password is likely to be different from that which was stored in LDAP.
- From NA-enabled to NA-enabled: The operation will succeed and, assuming that the same RACF database is used for NA at both the source and destination DIT branches, authentication attempts will succeed.

Note: Part of the reason LDAP does not support the moving of objects is that the protocol supports redirection, which can be used to build a high-level DIT from a group of servers, each responsible for a portion of the DIT. Therefore, a move may actually involve an object being relocated from one LDAP server to another. This is why we do not assume that two NA-enabled DITs are actually served by the same RACF database.

In essence, moving user objects when native authentication is in use, especially when portions of the DIT are not configured in the same way, might require additional password management.

3.5.2 Creating a RACF account for an LDAP user

A RACF user account is not automatically created when a user is added to LDAP. Before the LDAP user object can be used for authorized access, a RACF account corresponding to the LDAP object must be created.

3.5.3 Identifying the RACF account corresponding to the LDAP object

By default, the LDAP server uses the `uid` attribute of the LDAP object to determine the RACF account to be used for verification of the password. If IDs used in RACF are different than in LDAP, this default can be overridden by adding the `ibm-nativeId` attribute to the LDAP object. To do this, add the object class ibm-nativeAuthentication to the account.

Note: We found that the comparison between LDAP *uid* and RACF user is case-sensitive. This means that, since RACF user IDs are uppercase and Linux/UNIX user IDs are usually lowercase, the `ibm-nativeId` attribute will be required in almost all cases when using RACF native authentication.

In “Linux authentication using the z/VM LDAP server” on page 71, we illustrate this by using a Linux system set up to authenticate to our z/VM LDAP server.

3.6 Access control lists

The z/VM LDAP server maintains an access control list (ACL) for every entry in the LDAP database. The ACL determines what level of access is granted to the database object, depending on criteria such as the DN accessing the object, the time of day, the source IP address of the connection, and other criteria.

Before you start to use either the z/VM or z/OS LDAP servers, you need to consider the ACLs you will need to implement your security policy. You will need to consider issues such as:

- Employees having access to change some of their own details, such as phone number and password, but not their position or title
- Managers requiring access to change details for their employees such as work location, but not their passwords, and also not for employees of other departments
- Administrators or support staff needing to be able to reset passwords for employees

Setting appropriate access controls is an important part of establishing any corporate directory, not just one on the z/VM or z/OS LDAP servers.

Note: Detailed discussion of LDAP ACLs is found in Chapter 9 of the manual *z/VM TCP/IP LDAP Administration Guide (version 6 release 2)*, SC24-6236-01.

3.6.1 ACL permissions

There are two types of ACL permission: The ability to modify attributes of an object, and the ability to add or delete the object itself. Table 3-2 lists the permissions that apply to objects in the directory, while Table 3-3 lists available permissions that apply to attributes of objects.

Table 3-2 Permissions that apply to a database object

Add	Add an entry below this entry in the DIT
Delete	Delete this entry

Table 3-3 Permissions that apply to attributes of an object

Read	Read attribute values
Write	Write attribute values
Search	Search filter can contain attribute type
Compare	Compare attribute values

3.6.2 ACL format

The basic syntax of the aclEntry attribute is:

```
[access-id:|group:|role:]subject_DN[granted_rights]
```

The *subject_DN* is any valid DN, to which the ACL entry grants privileges. The *subject_DN* field ends when the first text recognized as a *granted_rights* string is detected.

There are two special DNs:

cn=anybody	A group containing all valid DNs in the database
cn=this	The DN that is requesting access to the object

These DNs can be combined with the `access-id` and `group` keywords to simplify configuration. For example, `group:cn=anybody` represents all DNs in the database (or, all users in the system), and `access-id:cn=this` is used to create an ACL that applies when a

DN is accessing its own object in the database (for example a user updating their own directory entry).

An example ACL entry is the following:

```
access-id:cn=this:normal:rwsc:system:rsc:critical:rwsc
```

This ACL will allow a user to read all attributes in their LDAP definition, but only update "normal" attributes (they have no access to system attributes). In addition, the schema defines the `userPassword` attribute as "critical", so this ACL allows a user to update their password.

Note: More detailed information about ACL entries, including ACL filters, can be found in the LDAP manuals:

- For z/VM, *z/VM TCP/IP LDAP Administration Guide (version 6 release 2)*, SC24-6236-01
- For z/OS, *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191

3.6.3 Propagating ACLs

It is not necessary to set `aclEntry` attributes against every object in the LDAP database. The z/VM LDAP server supports the propagation of ACLs. This means that a high-level object in the DIT (for example, a departmental or geographical organizationalUnit object) can set an ACL for objects below it. This is referred to as a *propagating ACL*.

The ACL on an object becomes a propagating ACL when the attribute `aclPropogate` is set to `TRUE` on that object.

If an ACL is not specified for an entry, the following processing is done by the LDAP server:

- If an object above the requested object in the LDAP DIT has the `aclPropogate` attribute set to `TRUE`, the ACL of the nearest such object is applied to the requested object; or
- The default ACL of `cn=anybody:normal:rsc:system:rsc` applies to the requested object.

Note: The default ACL is generated at the time the LDBM database is initialized. If no `aclEntry` is specified when the suffix entry is created, the LDAP server uses the following ACL:

```
aclEntry: cn=anybody:normal:rsc:system:rsc
aclPropagate: TRUE
```

This becomes the default ACL for other objects under that suffix, because it is a propagating ACL.

Determining the origin of an ACL

A request for the `aclEntry` attribute of an object will always return a value. The value returned is either the actual `aclEntry` attributes of the object, or the generated `aclEntry` attributes inherited via the `aclPropogate` attribute of a higher object in the DIT.

To determine which of these is the case, every object has an `aclSource` attribute. This attribute gives the DN of the object that owns the `aclEntry` attributes that apply to the object. If the `aclSource` attribute is the same as the DN of the object, the `aclEntry` attributes are set locally on that object.

Example 3-15 shows the result of an LDAP search on a user object in our directory.

Example 3-15 LDAP output showing ACL source attribute

```
lnxslesb:~ # ldapsearch -x -v -h 9.12.4.236 -b “ou=cambridge_L,dc=itso,dc=ibm
,dc=com” -D “cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com” -W
“(&(objectClass=inetorgperson)(uid=viccross))” aclEntry aclSource
ldap_initialize( ldap://9.12.4.236 )
Enter LDAP Password:
filter: (&(objectClass=inetorgperson)(uid=viccross))
requesting: aclEntry aclSource
# extended LDIF
#
# LDAPv3
# base <ou=cambridge_L,dc=itso,dc=ibm,dc=com> with scope subtree
# filter: (&(objectClass=inetorgperson)(uid=viccross))
# requesting: aclEntry aclSource
#

# Vic Cross, People, cambridge_L, itso.ibm.com
dn: cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com
aclentry: group:cn=ldapadmin,ou=Groups,ou=cambridge_L,dc=itso,dc=ibm,dc=com:no
 rmal:rwsc
aclentry: access-id:cn=this:normal:rwsc:sensitive:rwsc:critical:rwsc
aclentry: group:cn=Anybody:normal:rsc:system:rsc
aclsource: ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

The `aclSource` attribute is read-only and cannot be modified. It is automatically generated for each object by the LDAP server. In this example, the `aclSource` attribute shows that the source of this ACL is the parent object of this user record. This would be a common scenario giving administrators appropriate access to records of multiple users at a time.

3.6.4 Updating ACLs

Since ACLs are managed by the LDAP server as additional attributes against objects in the LDAP DIT, normal LDAP utilities are used to update or change ACLs. The normal way is to use command line utilities such as **`ldapmodify`** with LDIF files however, since most graphical LDAP utilities (including phpLDAPadmin) do not expose the ACL attributes in their interface.

Note: You can use phpLDAPadmin to display ACL attributes on objects by selecting the “Show internal attributes” option when displaying an object. While this allows you to view the ACL attributes, you cannot change them using this interface.

If you are making a change to an existing `aclEntry` attribute, you must take care to provide the complete attribute content in your update. This is because the LDAP server replaces the attribute content with the update content. For example, the ACL in Example 3-15 gives full access (read, write, search and compare) for “sensitive” class attributes to the object represented by *access-id:cn=this*. To remove write access to this class of attributes, it is not sufficient to simply issue an update containing the following ACL entry:

```
aclEntry: access-id:cn=this:sensitive:rsc
```

Doing this would remove all access to normal and critical attributes. The correct ACL update is shown in the LDIF in Example 3-16.

Example 3-16 LDIF for ACL update

```
dn: ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com
changetype: modify
add: aclEntry
aclEntry: access-id:cn=this:normal:rwsc:sensitive:rsc:critical:rwsc
```

Note that even though an add operation was requested, the LDAP server will detect that a matching ACL already exists and use this update to replace the existing one. This makes it easy to replace existing ACLs without having to delete any possible existing entries first.

Also, note that the LDIF in Example 3-16 updates the ACL of the `ou=People` container object rather than the `cn=Vic Cross` object. We did this because `ou=People` is the source of the ACL that applied to the `cn=Vic Cross` object (see the `aclSource` attribute in the LDAP search output from Example 3-15). Because the ACL on the `ou=People` object is a propagating ACL, it will apply to all the objects below it that do not already have their own ACL. If we wanted the updated ACL to apply just to the `cn=Vic Cross` object, we would have specified the DN for that object in the LDIF update. That would have resulted in a specific ACL on the `cn=Vic Cross` object that would override the propagating ACL from the `ou=People` object.

Note: Full details of how to override propagating ACLs can be found in *z/VM TCP/IP LDAP Administration Guide (version 6 release 2)*, SC24-6236-01.

3.7 Linux authentication using the z/VM LDAP server

This topic has been covered in IBM Redbooks publications such as *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221 and *Security on z/VM*, SG24-7471. In this section, we illustrate what we saw in our configuration when we used LDAP according to the existing documents.

3.7.1 Using YaST to enable LDAP on SLES 11 SP2

We used Yet another Setup Tool (YaST) to activate the LDAP client in our SLES 11 SP2 server. We used the command **`yast2`** to invoke YaST, then selected **LDAP Client** from the Network Services menu shown in Figure 3-1 on page 72.

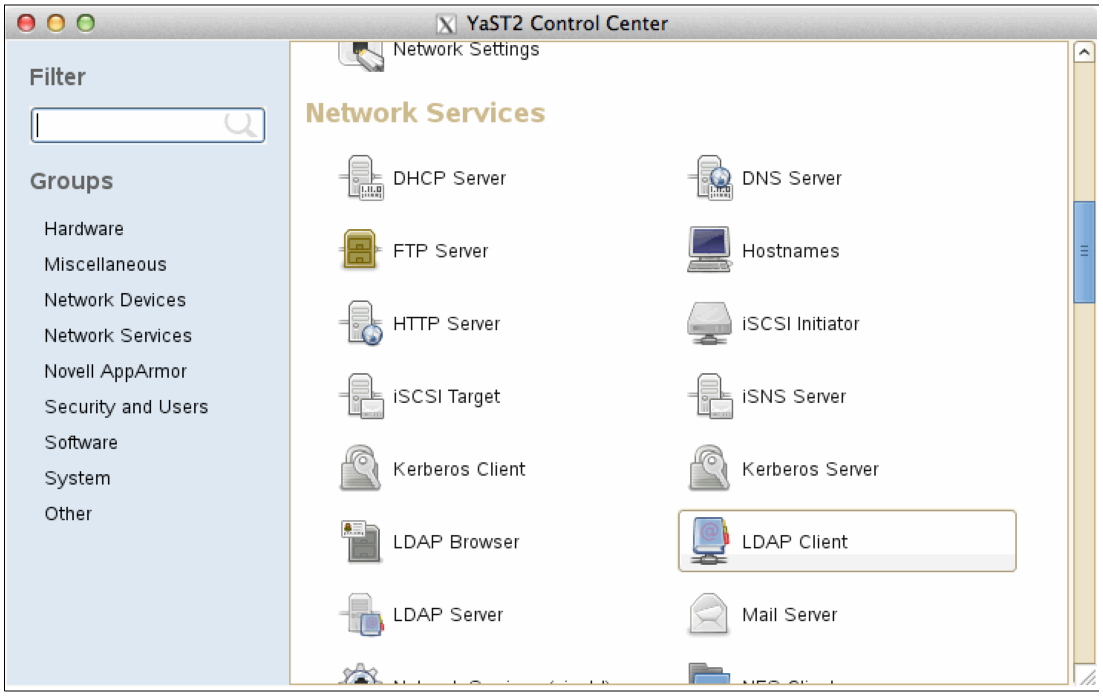
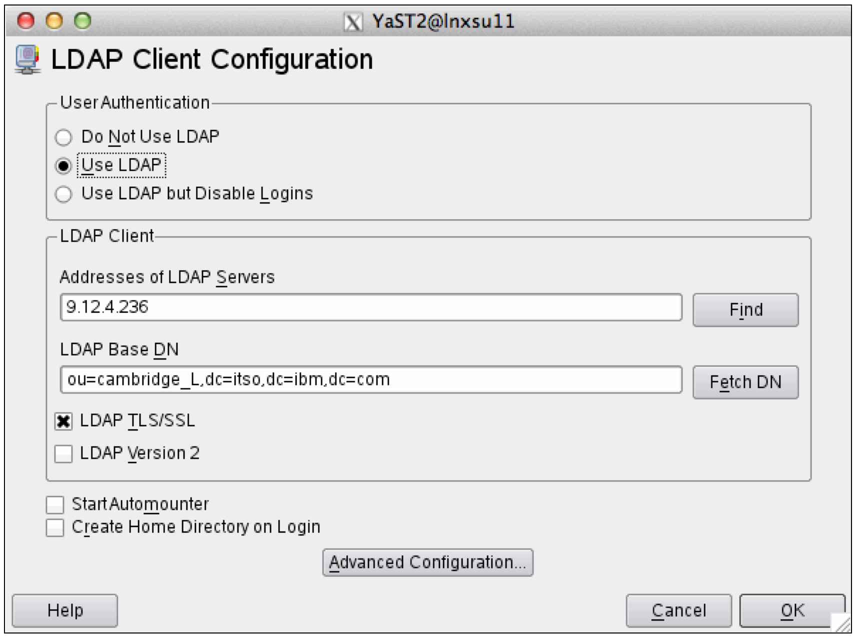


Figure 3-1 Selecting the LDAP Client configuration utility from the YaST2 Control Center

The LDAP Client configuration utility opens. We selected **Use LDAP** and filled in the details of our z/VM LDAP server, as shown in Figure 3-2.



YaST2@lnxsu11

LDAP Client Configuration

User Authentication

- ○ Do Not Use LDAP
- ● Use LDAP
- ○ Use LDAP but Disable Logins

LDAP Client

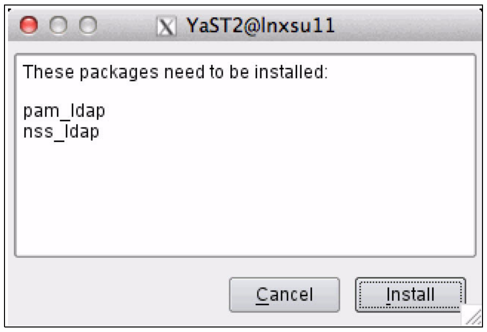
Addresses of LDAP Servers: 9.12.4.236

LDAP Base DN: ou=cambridge_L,dc=itso,dc=ibm,dc=com

- [x] LDAP TLS/SSL
- [ ] LDAP Version 2
- [ ] Start Automounter
- [ ] Create Home Directory on Login

Figure 3-2 Configuring the LDAP Client

We clicked **OK**. A message from YaST, shown in Figure 3-3 on page 73, indicated that additional packages had to be installed.


YaST2@lnxsu11

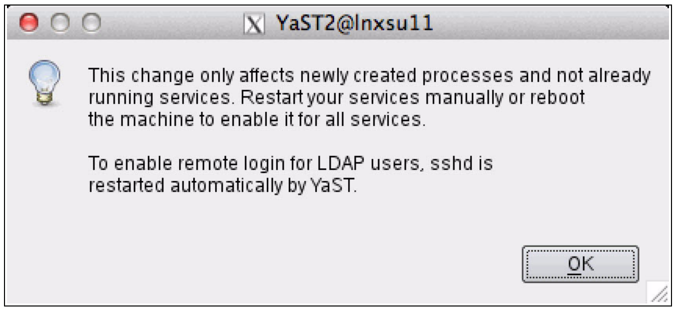
These packages need to be installed:

pam_ldap
nss_ldap

Figure 3-3 YaST2 information message about extra packages required for LDAP

Note: If you have not previously installed any LDAP capability in your server, you might see a longer list of packages when you perform this operation. This is normal, since YaST automatically determines all the prerequisites required.

We clicked **Install**. YaST installed the additional required packages. Another message, shown in Figure 3-4, informed us to restart the tasks so that they pick up the LDAP client support.

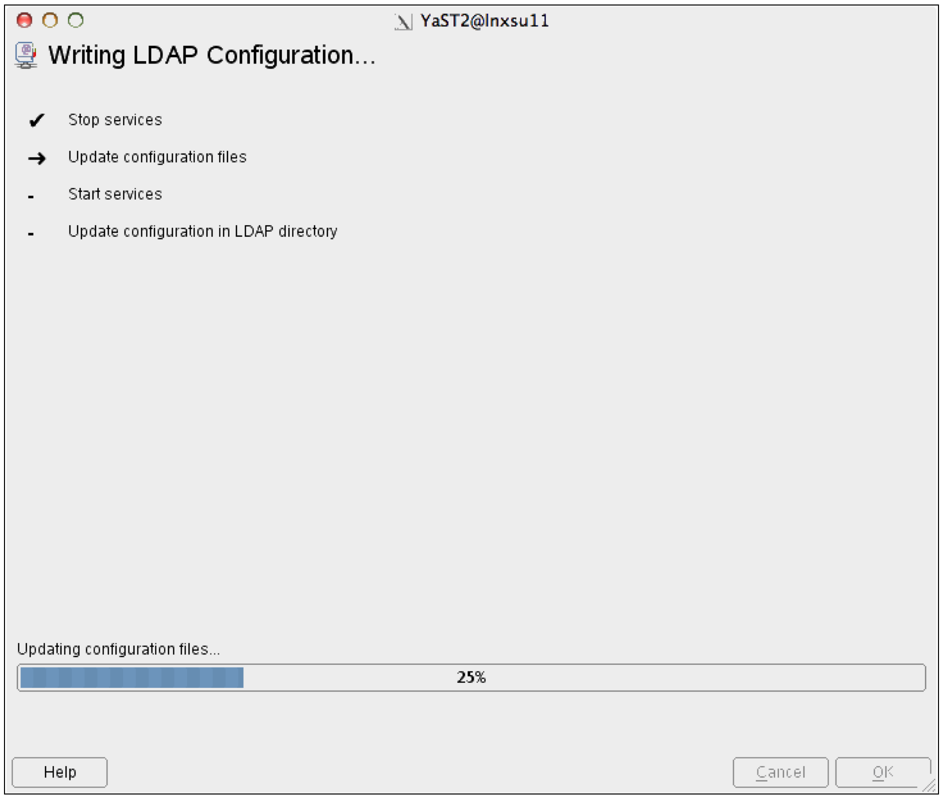

YaST2@lnxsu11

This change only affects newly created processes and not already running services. Restart your services manually or reboot the machine to enable it for all services.

To enable remote login for LDAP users, sshd is restarted automatically by YaST.

Figure 3-4 Task restart requirement warning from YaST2

We clicked **OK** at this message. The progress panel from YaST2, shown in Figure 3-5 on page 74, opened to indicate that the system configuration was being performed.



Writing LDAP Configuration...

✔ Stop services

→ Update configuration files

- Start services

- Update configuration in LDAP directory

Updating configuration files...

25%

Figure 3-5 YaST2 progress panel showing the activation of the altered configuration

After processing completes, the YaST2 Control Center is displayed again. We clicked **Quit** to exit YaST2.

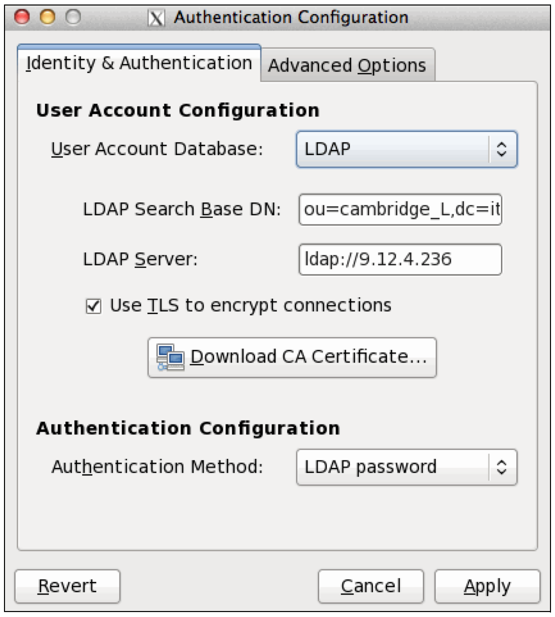
To verify that our system now had access to LDAP account information, we issued the **`id`** command, as shown in Example 3-17, to display the user ID we added earlier.

Example 3-17 Using the id command to test the LDAP client configuration

```
lnxsu11:~ # id viccross
uid=10001(viccross) gid=500(itsousers) groups=500(itsousers)
```

3.7.2 Enabling LDAP authentication on RHEL 6

RHEL includes a configuration program called system-config-authentication which performs configuration of SSSD. It is a graphical program, so you need to have X display support to run it (either on a VNC session or using X forwarding). The interface of system-config-authentication is shown in Figure 3-6 on page 75.


Authentication Configuration
Identity & Authentication
Advanced Options
Download CA Certificate...
Revert
Cancel
Apply

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: ou=cambridge_L,dc=it

LDAP Server: ldap://9.12.4.236

☑ Use TLS to encrypt connections

Authentication Configuration

Authentication Method: LDAP password

Figure 3-6 Red Hat Enterprise Linux 6 system-config-authentication utility

SSSD provides enhanced security for passwords because, when the “Authentication Method” is set to “LDAP Password”, it enforces that the LDAP connection be configured securely. If you try to configure non-secure LDAP, you get the warning message shown in Figure 3-7 and the tool will not save your configuration.

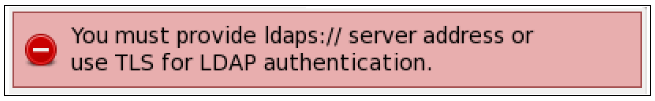

You must provide ldaps:// server address or use TLS for LDAP authentication.

Figure 3-7 system-config-authentication error message when non-secure LDAP is configured

We filled in the values as shown in Figure 3-6, but were unable to get SSSD to work against the z/VM LDAP server. By following the logs of SSSD we found that SSL/TLS negotiation was failing, but there was not enough information in the SSSD log to show why. Using network tracing we discovered that the secure connection fails at the initial handshake stage. One reason for this could be that SSSD and the z/VM LDAP server cannot agree on a suitable cipher, causing the establishment of a secure channel to fail.

To verify that the z/VM LDAP server was able to satisfy the LDAP requirements of SSSD, we were able to manually force SSSD to accept a non-secure LDAP channel by adding the following entry to `/etc/sssd/sssd.conf`:

```
ldap_auth_disable_tls_never_use_in_production = True
```

We also changed `ldap_id_use_start_tls` to be `False`. With this configuration in place, we were able to log on to our RHEL 6 server using our z/VM LDAP credentials. However, as the configuration setting suggests, this is not a workaround that should be used in a production environment. This may be a situation where the use of an OpenLDAP server with z/VM LDAP as an authentication backend, as discussed in “Using an OpenLDAP server with the z/VM LDAP server” on page 86, might be ideal.

3.7.3 Mapping the LDAP account to RACF

The RACF accounts corresponding to user objects in LDAP must be defined before logons can occur successfully. Example 3-18 shows the messages in the system log of our Linux server when we tried to log on when the RACF ID was not defined.

Example 3-18 Authentication error from pam_ldap

```
Aug 28 15:17:43 lnxsu1l sshd[44046]: pam_ldap: error trying to bind as user
"cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com" (Operations error)
Aug 28 15:17:43 lnxsu1 sshd[44044]: error: PAM: Authentication failure for
viccross from fearless.pok.ibm.com
```

The pam_ldap module was able to locate the correct LDAP object based on the `uid` attribute, but the LDAP bind failed because of the lack of the RACF account.

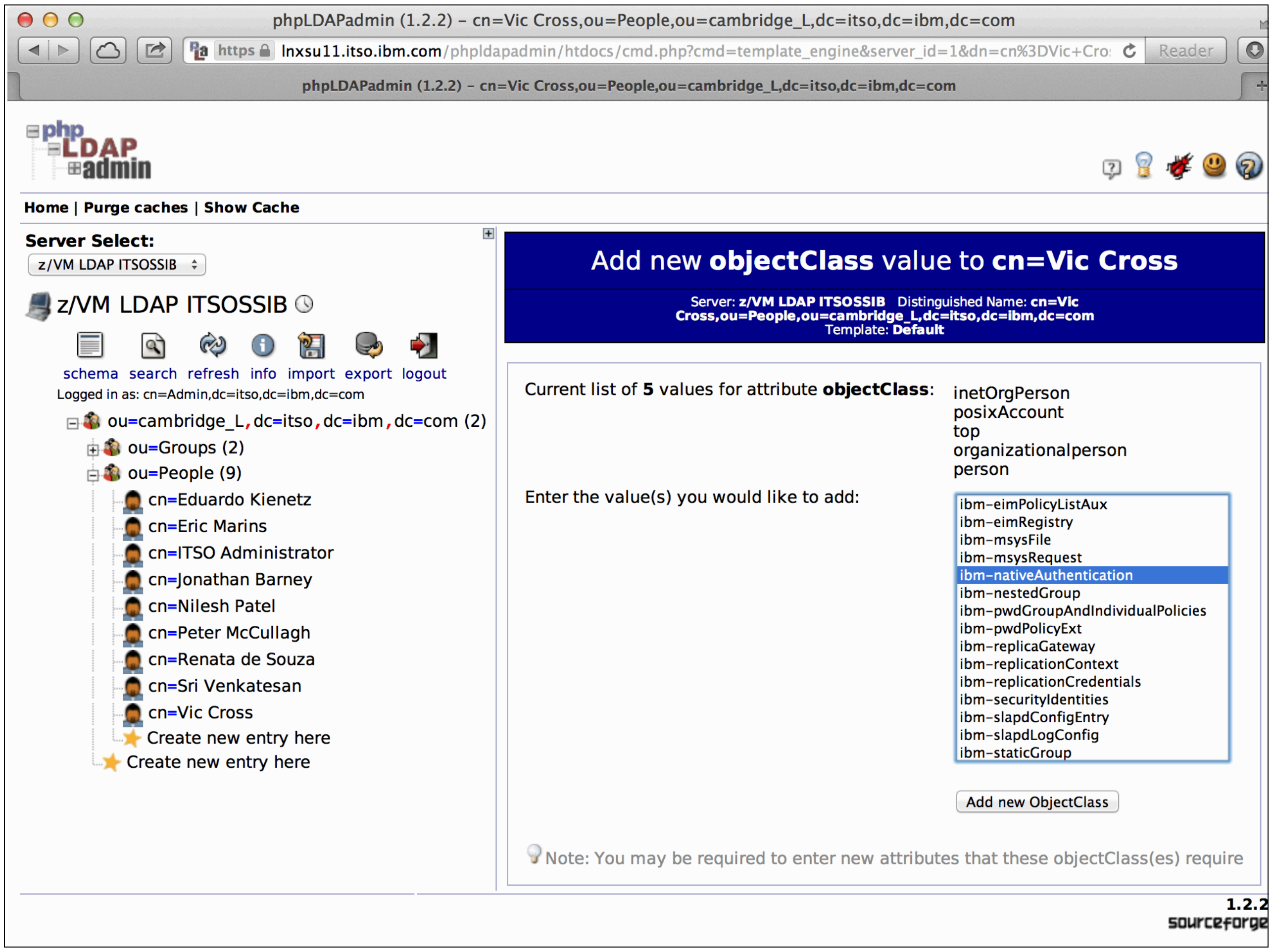
We have two ways to resolve this issue:

- Create a RACF account that matches the `uid` attribute of the LDAP object.
- If the user already has a z/VM user id, point the LDAP object to the existing RACF account using the `ibm-nativeId` attribute.

Note: A third way to solve the login problem does exist, of course. That is to remove native authentication and add the `userPassword` attribute to the LDAP object. Because we are talking about using RACF for authentication, however, that solution to the login problem is not further discussed.

Using phpLDAPadmin, we added the ibm-nativeAuthentication object class to the LDAP object. After logging on to phpLDAPadmin and navigating to the edit panel for the user object, we clicked the **(add value)** link under the objectClass heading. This displayed the panel shown in Figure 3-8 on page 77, where we selected **ibm-nativeAuthentication** from the list and clicked **Add new ObjectClass**.

Note: Setting up phpLDAPadmin to work with the z/VM LDAP server is covered in Appendix A, “Using phpLDAPadmin to manage the z/VM and z/OS LDAP servers” on page 299.


phpLDAPadmin (1.2.2) – cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?cmd=template_engine&server_id=1&dn=cn%3DVic+Cro

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

- ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
  - ou=Groups (2)
  - ou=People (9)
    - cn=Eduardo Kienetz
    - cn=Eric Marins
    - cn=ITSO Administrator
    - cn=Jonathan Barney
    - cn=Nilesh Patel
    - cn=Peter McCullagh
    - cn=Renata de Souza
    - cn=Sri Venkatesan
    - cn=Vic Cross
    - Create new entry here
  - Create new entry here

Add new objectClass value to cn=Vic Cross

Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Current list of **5** values for attribute **objectClass**:

inetOrgPerson
posixAccount
top
organizationalperson
person

Enter the value(s) you would like to add:

ibm-eimPolicyListAux
ibm-eimRegistry
ibm-msysFile
ibm-msysRequest
ibm-nativeAuthentication
ibm-nestedGroup
ibm-pwdGroupAndIndividualPolicies
ibm-pwdPolicyExt
ibm-replicaGateway
ibm-replicationContext
ibm-replicationCredentials
ibm-securityIdentities
ibm-slapdConfigEntry
ibm-slapdLogConfig
ibm-staticGroup

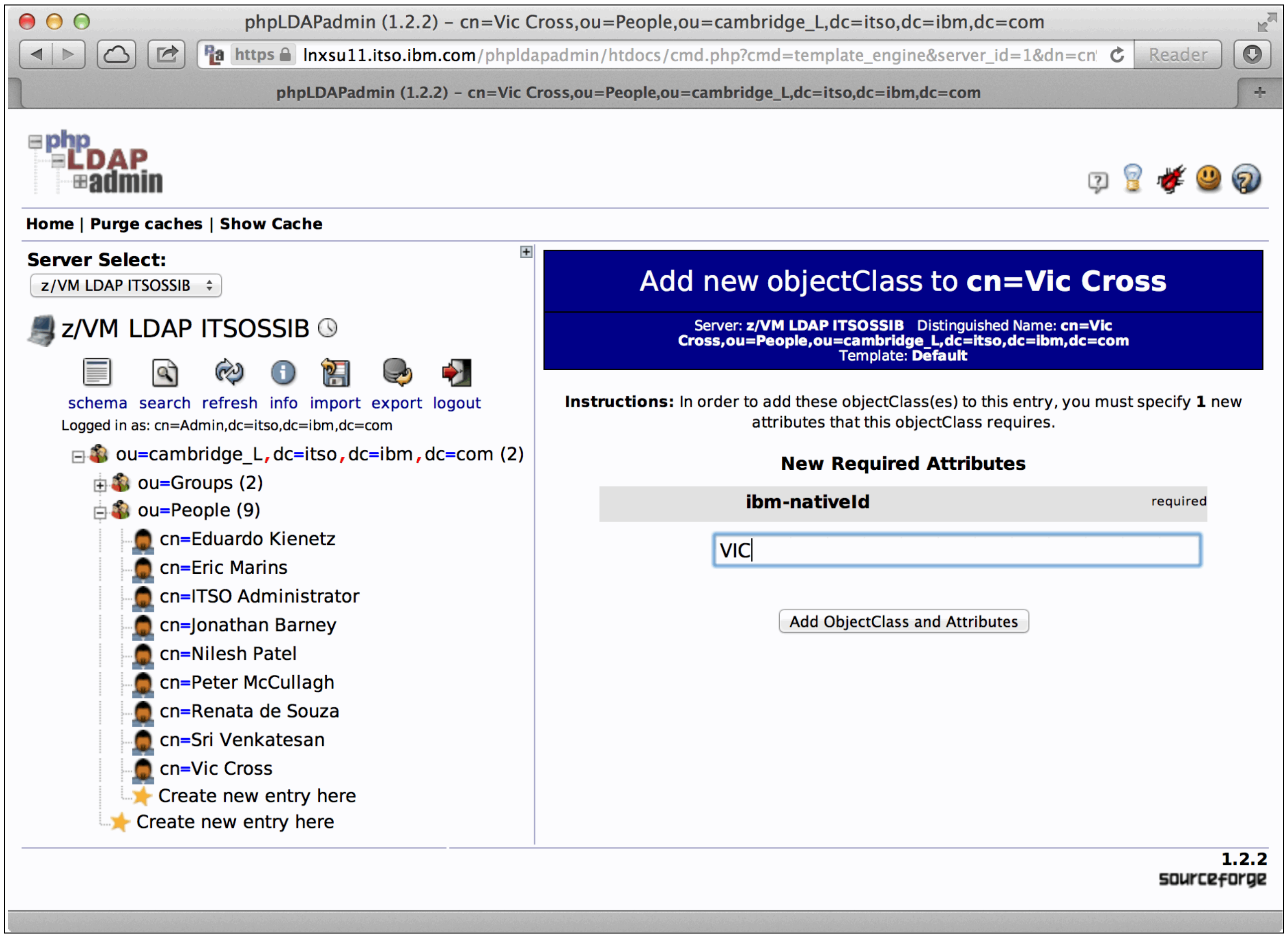
Add new ObjectClass

Note: You may be required to enter new attributes that these objectClass(es) require

1.2.2

Figure 3-8 Adding the ibm-nativeAuthentication object class to an LDAP object

As we saw previously, phpLDAPadmin checks the new object class for any mandatory attributes and prompts for these. This is shown in Figure 3-9 on page 78 (with the value we filled in for `ibm-nativeId` shown).


phpLDAPadmin (1.2.2) – cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?cmd=template_engine&server_id=1&dn=cn

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

- ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
  - ou=Groups (2)
  - ou=People (9)
    - cn=Eduardo Kienetz
    - cn=Eric Marins
    - cn=ITSO Administrator
    - cn=Jonathan Barney
    - cn=Nilesh Patel
    - cn=Peter McCullagh
    - cn=Renata de Souza
    - cn=Sri Venkatesan
    - cn=Vic Cross
    - Create new entry here
  - Create new entry here

Add new objectClass to cn=Vic Cross

Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Instructions: In order to add these objectClass(es) to this entry, you must specify **1** new attributes that this objectClass requires.

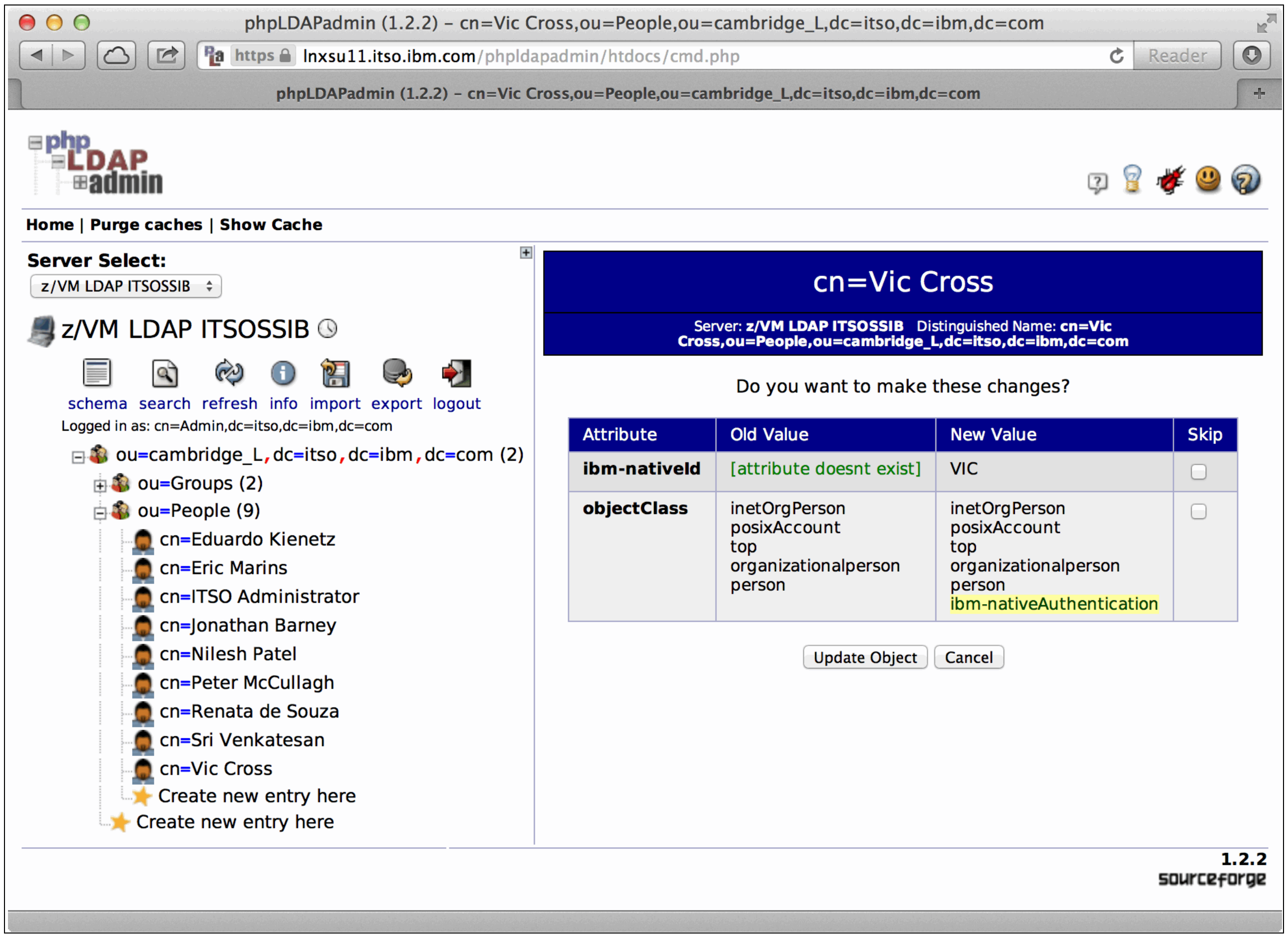
New Required Attributes

ibm-nativeId required

VIC

Figure 3-9 Filling in the value for the ibm-nativeId attribute

As shown in Figure 3-10 on page 79, phpLDAPadmin then asks us to confirm our LDAP changes.



phpLDAPadmin (1.2.2) – cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

```
ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
  ou=Groups (2)
  ou=People (9)
    cn=Eduardo Kienetz
    cn=Eric Marins
    cn=ITSO Administrator
    cn=Jonathan Barney
    cn=Nilesh Patel
    cn=Peter McCullagh
    cn=Renata de Souza
    cn=Sri Venkatesan
    cn=Vic Cross
    Create new entry here
  Create new entry here
```

cn=Vic Cross

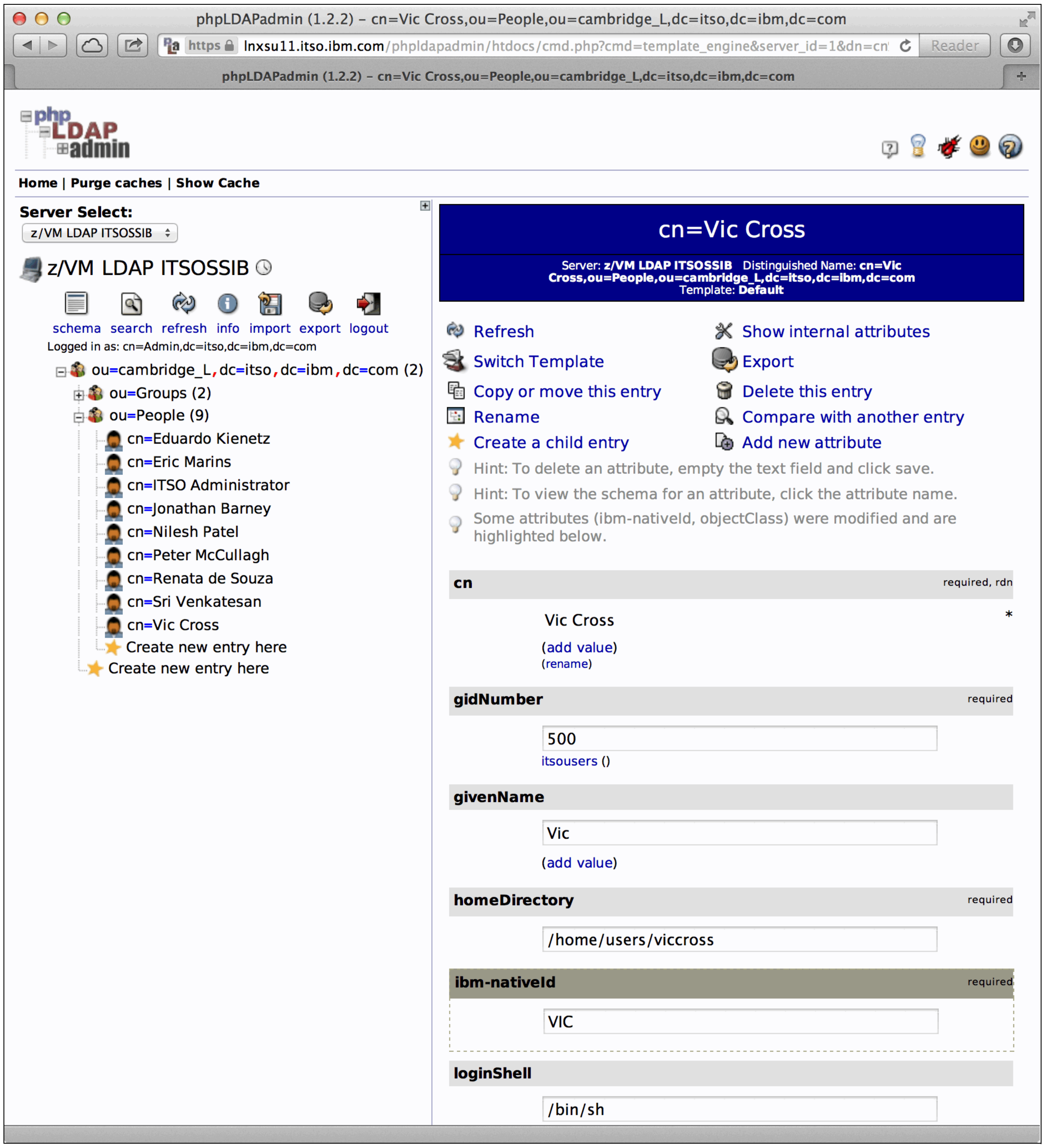
Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**

Do you want to make these changes?

Attribute	Old Value	New Value	Skip
ibm-nativeId	[attribute doesnt exist]	VIC	☐
objectClass	inetOrgPerson posixAccount top organizationalperson person	inetOrgPerson posixAccount top organizationalperson person ibm-nativeAuthentication	☐

Figure 3-10 Confirmation of the change to be made to our LDAP user object

In this example you can see the changes that will be made: the addition of the `ibm-nativeId` attribute, and the addition of `ibm-nativeAuthentication` to the `objectClass` attribute to allow the new attribute to be added. When we click **Commit**, the database is updated, as shown in Figure 3-11 on page 80.



phpLDAPadmin (1.2.2) – cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?cmd=template_engine&server_id=1&dn=cn

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

- ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
  - ou=Groups (2)
  - ou=People (9)
    - cn=Eduardo Kienetz
    - cn=Eric Marins
    - cn=ITSO Administrator
    - cn=Jonathan Barney
    - cn=Nilesh Patel
    - cn=Peter McCullagh
    - cn=Renata de Souza
    - cn=Sri Venkatesan
    - cn=Vic Cross
    - Create new entry here
  - Create new entry here

cn=Vic Cross

Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **cn=Vic Cross,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Refresh | Show internal attributes
Switch Template | Export
Copy or move this entry | Delete this entry
Rename | Compare with another entry
Create a child entry | Add new attribute

Hint: To delete an attribute, empty the text field and click save.

Hint: To view the schema for an attribute, click the attribute name.

Some attributes (ibm-nativeId, objectClass) were modified and are highlighted below.

cn (required, rdn)	Vic Cross (add value) (rename)
gidNumber (required)	500 itsousers ()
givenName	Vic (add value)
homeDirectory (required)	/home/users/viccross
ibm-nativeId (required)	VIC
loginShell	/bin/sh

Figure 3-11 The LDAP user object successfully updated

To verify that the change was successful, we tried to log in to the Linux server both before and after the LDAP update. The login attempts are shown in Example 3-19 (the LDAP update was performed between the two login attempts).

Example 3-19 Testing login by using LDAP credentials

```
[viccross@fearless ~]$ ssh 9.12.5.100
Password: <RACF password entered> <-- before LDAP update
Password: <RACF password entered> <-- after LDAP update
Last login: Mon Aug 27 16:08:34 2012 from fearless.pok.ibm.com
lnxsu11:~> id
```

```
uid=10001(viccross) gid=500(itsousers) groups=500(itsousers)
```

3.7.4 Password management with Linux and the z/VM LDAP server

When using native authentication in the z/VM LDAP server, it is important to remember that the `userPassword` attribute is managed by RACF and is not actually present in the LDBM backend. This is why the object add failed when we tried to create the user account with the password attribute—the ldap_add API was being used, which the z/VM LDAP server does not support for operations on the `userPassword` attribute.

The z/VM LDAP server does support changing the RACF password using the ldap_modify API, however. To make this work on a Linux LDAP client, a change is required in the client configuration. The parameter *pam_password* must be set to `racf`, as shown in Example 3-20.

Example 3-20 Password change configuration setting in /etc/ldap.conf

```
...
# Password setting for IBM RACF
pam_password racf
...
```

Example 3-21 shows the output of the Linux **`passwd`** command when used to change a RACF password via LDAP.

Example 3-21 Changing the RACF password using the Linux passwd command via pam_ldap

```
lnxsu11:~> passwd
Changing password for viccross.
Enter login(LDAP) password: <existing password>
New Password: <new password>
Reenter New Password: <new password>
LDAP password information changed for viccross
lnxsu11:~>
```

Changing an expired RACF password via LDAP

RACF supports an altered password change syntax which allows the password to be changed during the LDAP bind operation. When the RACF password is expired, this method has to be used because the normal method (through the LDAP modify API) does not work. To use this RACF support, instead of entering their normal password a user would provide a string in the following format:

`currentpassword/newpassword`

If the `currentpassword` matches the user's existing password, and `newpassword` is valid according to RACF password rules, the password is updated. If the password change string was entered at a login prompt, the user would be successfully logged on to the application.

Note: The full requirements for changing a RACF password during LDAP bind are explained in the section "Updating native passwords or password phrases during bind" in the "IBM Tivoli® Directory Server for z/OS Administration and Use" manual.

Unfortunately the pam_ldap module does not fully support this altered syntax, which means that updating an expired RACF password from Linux is different than a normal password change.

Note: z/VM LDAP does inform pam_ldap that the password is expired, but pam_ldap attempts to use the normal LDAP modify method to change the password instead of the bind password change, which fails because z/VM LDAP does not allow the bind to succeed using the expired password.

There are a number of options available for how to handle the change of an expired password:

- After a forced reset of a user password, inform the user that their next logon must use the altered RACF password change syntax for the password. This is a solution for any situation where the user knows their password is already expired, but if the password expires without their knowledge they will require a different solution;
- Make users aware that if they are prompted to change an expired password they must use the RACF bind password change syntax when prompted for their current password and they will receive errors (or they must terminate the login attempt and retry) if they continue the change password attempt;
- Use the `pam_password_prohibit_message` statement in the ldap.conf file to direct users to an external password change facility that can be used to update the password.

Use change password syntax at initial login

This method will work to change a password at any time, but is most useful when the password is expired. Example 3-22 shows a logon using the change password syntax.

Example 3-22 Changing password at login using SSH

```
[viccross@fearless ~]$ ssh 9.12.5.100
Password: <old password>/<new password>  <-- passwords not shown on screen
Last login: Mon Aug 27 15:59:23 2012 from fearless.pok.ibm.com
lnxsu11:~>
```

The password change is handled entirely between the z/VM LDAP server and RACF, which means that pam_ldap is simply informed that the password was correctly provided. This is why there is no confirmation text advising a password change has occurred.

Error messages when using pam_ldap for changing an expired password

When logging on to Linux using an account with an expired RACF password, pam_ldap receives the correct notification that the password is expired. However, after it prompts to change the password it attempts to use LDAP modify to change the password, which does not function correctly if the password is expired. Example 3-23 shows what happened when we tried to change an expired password at login using the pam_ldap prompts.

Example 3-23 Errors received at login when trying to change an expired password

```
[viccross@fearless ~]$ ssh 9.12.5.100
Password: <old password>
You are required to change your LDAP password immediately.
Enter login(LDAP) password: <old password>
New Password: <new password>
Reenter New Password: <new password>
LDAP password information update failed: Insufficient access
R003070 Access denied because user does not have ‘write’ permission for all
modified attributes (ldbm_modify_entry)

Password:
```

By enabling error logging in the LDAP server, we determined that the situation was caused by pam_ldap not being able to bind using the existing password. This caused pam_ldap to try to update the password using an anonymous bind, which failed because there is no access given to update the LDAP data store via an anonymous bind.

The best option in this situation is to use the LDAP bind method to update the password, and then cancel out of the pam_ldap password change attempt. A session illustrating this is shown in Example 3-24.

Example 3-24 Changing an expired password at login using LDAP bind password change syntax

```
[viccross@fearless ~]$ ssh 9.12.5.100
Password: <old password>
You are required to change your LDAP password immediately.
Enter login(LDAP) password: <old password>/<new password>
New Password: ^C

[viccross@fearless ~]$ ssh 9.12.5.100
Password: <new password>
Last login: Tue Aug 28 10:50:04 2012 from fearless.pok.ibm.com
lnxsu11:~>
```

While this is a functional way to change the password, it will be confusing to most users.

External password change mechanism

You can provide a password change tool external to the Linux system to manage password changes. While this may cause some inconvenience for users familiar with Linux or UNIX tools for password management, the use of a consistent interface has support and management benefits.

A simple password utility can be created using a web server with support for a web scripting language such as PHP or Perl. We found a simple password change tool written in PHP on the Internet and made some alterations to it to illustrate the concept. The key change required was to locate the LDAP modify call and change it to an LDAP bind call with the correct password syntax. The line of code is shown in Example 3-25.

Example 3-25 PHP code to perform a RACF password change using LDAP bind

```
...
if (ldap_bind($ldap_con,$records[0][“dn”],”$oldPassword/$newPassword”) === false) {
...
```

We also had to remove a couple of lines of code that did a bind using the existing password to verify that the password was correctly provided. Because the z/VM and z/OS LDAP servers fail a bind using an expired password, this test caused the tool to fail. Removing the check did not harm operation, since RACF verifies the current password before completing the password change.

Enforcing this external password change mechanism in pam_ldap is done using the `pam_password_prohibit_message` statement in `/etc/ldap.conf`. This is shown in Example 3-26.

Example 3-26 Setting up pam_ldap to support an external password change utility

```
...
pam_password_prohibit_message Please visit https://lnxsu11.itso.ibm.com/password to change your password.
...
```

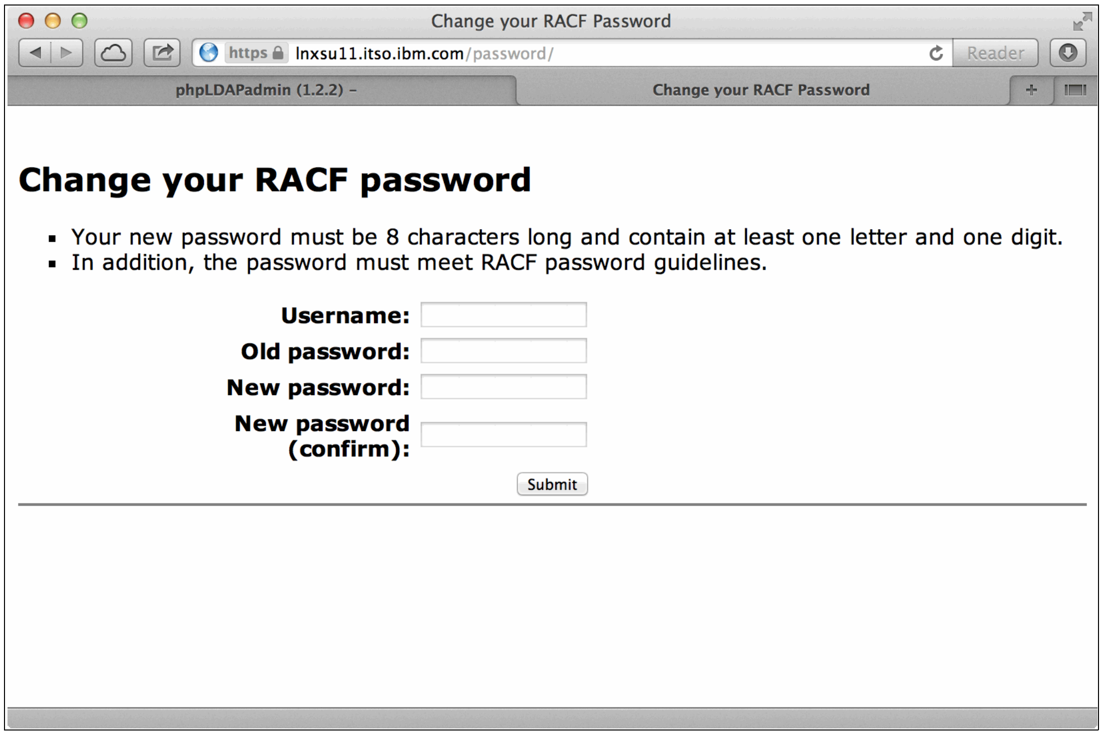
When users try to change their password using PAM, the attempt is rejected and the system displays the message given on the `pam_password_prohibit_message` statement. This is shown in Example 3-27, for both a required password change at logon and for an attempted password change using the Linux **`passwd`** command.

Example 3-27 Example of output when pam_password_prohibit_message is in effect

```
[viccross@fearless ~]$ ssh 9.12.5.100
Password: <current password>
You are required to change your LDAP password immediately.
Please visit https://lnxsu11.itso.ibm.com/password to change your password.
```

```
lnxsu11:~> passwd
Changing password for viccross.
Please visit https://lnxsu11.itso.ibm.com/password to change your password.
passwd: Permission denied
lnxsu11:~>
```

The next three images show the password change tool in use. Figure 3-12 shows the initial panel.


Change your RACF Password
https lnxsu11.itso.ibm.com/password/
Reader
phpLDAPadmin (1.2.2) –
Change your RACF Password

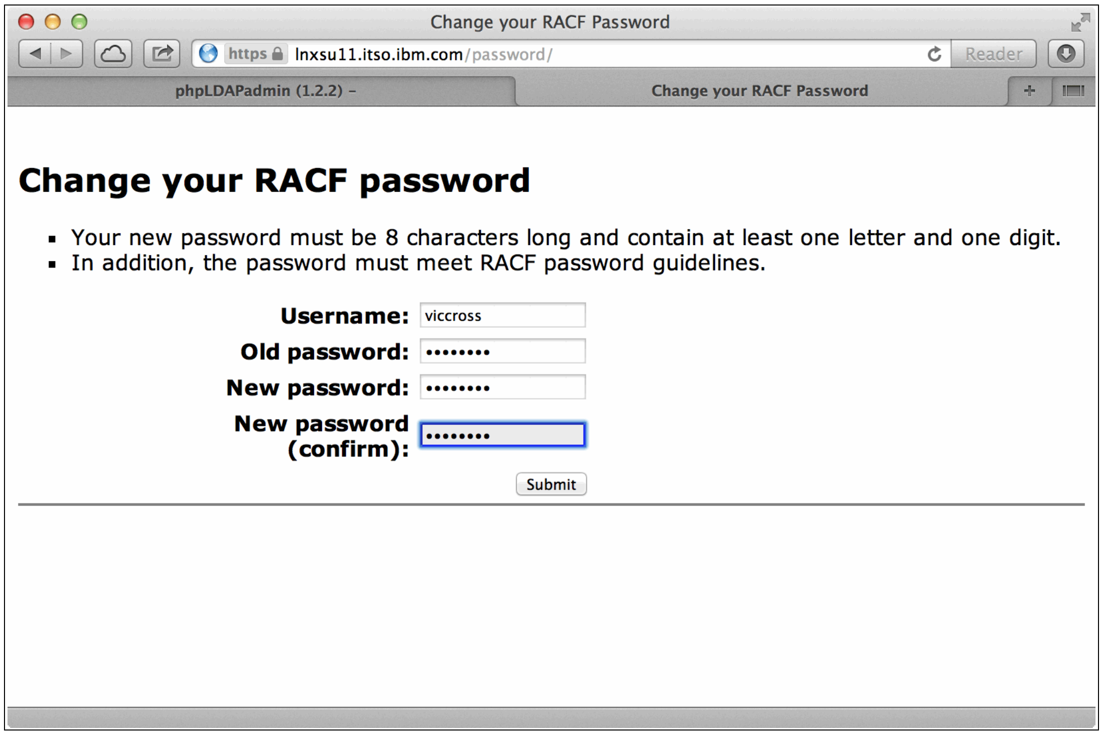
Change your RACF password

- Your new password must be 8 characters long and contain at least one letter and one digit.
- In addition, the password must meet RACF password guidelines.

Username:
Old password:
New password:
New password (confirm):
Submit

Figure 3-12 Password change utility initial panel

Figure 3-13 on page 85 shows the details of a password change entered on the panel. The user ID is the ID as known to LDAP, which might not be the same as the actual RACF ID.


Change your RACF Password
https lnxsu11.itso.ibm.com/password/
Reader
phpLDAPadmin (1.2.2) –
Change your RACF Password

Change your RACF password

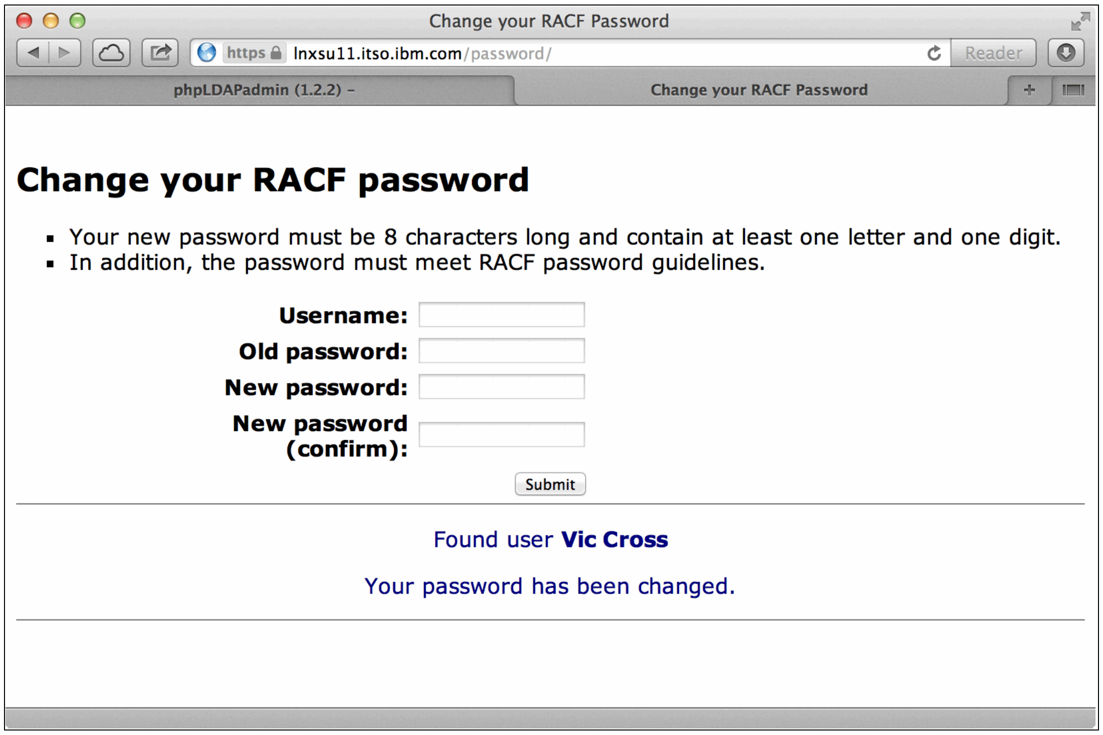
- Your new password must be 8 characters long and contain at least one letter and one digit.
- In addition, the password must meet RACF password guidelines.

Username:	viccross
Old password:	••••••••
New password:	••••••••
New password (confirm):	••••••••

Submit

Figure 3-13 Details entered on the password change utility panel

Figure 3-14 shows the result of a successful password change.


Change your RACF Password
https lnxsu11.itso.ibm.com/password/
Reader
phpLDAPadmin (1.2.2) –
Change your RACF Password

Change your RACF password

- Your new password must be 8 characters long and contain at least one letter and one digit.
- In addition, the password must meet RACF password guidelines.

Username:	
Old password:	
New password:	
New password (confirm):	

Submit

Found user **Vic Cross**

Your password has been changed.

Figure 3-14 Successful password change using the password change utility

3.8 Using an OpenLDAP server with the z/VM LDAP server

If you have to make extensive modifications to the schema of your LDAP server, or you are running applications on Linux that only work with OpenLDAP, you might not be able to use even the LDBM backend of the z/VM LDAP server.

We can think of only a few circumstances where this might be the case:

- You have an application vendor that is not providing support for its application on the z/VM LDAP server.
- The substitution of the International Alphabet Number 5 (IA5) versions of the matching rules (as discussed in "Converting OpenLDAP schema files for the z/VM LDAP server" on page 62) does not work properly for your application.
- You have complex requirements around LDAP data replication that cannot be accommodated in the z/VM LDAP server.
- You want to use some of the creative capabilities of OpenLDAP, such as scripted overlays and alternative backends, to build a highly customized LDAP capability.

Although other reasons might also exist, we believe that the LDBM backend of the z/VM LDAP server offers a high degree of function for supporting Linux applications and is a good approach to use when working with RACF on z/VM.

3.8.1 The OpenLDAP rewrite-remap overlay

If you find that you are unable to use the z/VM LDAP server and are implementing OpenLDAP, you might still be able to utilize RACF authentication. Using the z/VM LDAP server and a capability of the OpenLDAP server known as *overlays*, you can have the OpenLDAP server handle the majority of database references while authentication is redirected transparently to the z/VM LDAP server (and therefore to RACF).

In this configuration, the OpenLDAP database is set up as usual. After the basic setup is verified, the rewrite-remap (rwm) overlay is added and configured to rewrite bind requests to a DN hosted by the z/VM LDAP server. The back_ldap backend is used to give the OpenLDAP server access to the DIT managed by the z/VM LDAP server.

Note: The documentation for the slapo_rwm overlay suggests that it is still considered experimental. However, it has been in this state for several releases of OpenLDAP and is extensively used.

There is no requirement for any special configuration on the z/VM LDAP server to support the OpenLDAP *ldap* database or the slapo_rwm overlay.

3.8.2 Configuring OpenLDAP to authenticate using z/VM LDBM

We set up an OpenLDAP server on our Linux system, and added user objects to it. The DIT we used was similar to the one we used on the z/VM LDAP server.

Configuring the OpenLDAP server is no longer done using a single static configuration file. OpenLDAP supports a special DIT for configuration data which appears at *cn=config*. Making changes to the LDAP server configuration is then done by updating entries in this DIT. However, to configure the overlay and its dependencies correctly, it may have to be inserted into the database list ahead of existing databases, which is difficult to do dynamically. We found the easiest method was to shut down the OpenLDAP server and manipulate files in the /etc/openldap/slapd.d directory (which is the file store for the cn=config database). Upon restarting OpenLDAP, the modified configuration was brought online.

Configuring the OpenLDAP ldap database

Example 3-28 shows the database configuration for the LDAP database backend. This configuration had to be inserted into the list of configured databases in the correct place in our configuration because of the database suffix in use on one of the existing databases in our configuration. We had to move an existing database from order 2 to order 3 (by renaming its LDIF file and updating the dn and olcDatabase attributes) and adding this file as /etc/openldap/slapd.d/cn=config/olcDatabase{2}ldap.ldif.

Example 3-28 LDIF file to define the z/VM LDBM database as an LDAP backend to OpenLDAP

```
# LDIF for LDAP backend
# Added to our OpenLDAP server as:
# /etc/openldap/slapd.d/cn=config/olcDatabase={2}ldap.ldif
dn: olcDatabase={2}ldap
objectClass: olcDatabaseConfig
objectClass: olcLDAPConfig
olcDatabase: {2}ldap
olcSuffix: ou=cambridge_L,dc=itso,dc=ibm,dc=com
olcAddContentAcl: FALSE
olcLastMod: FALSE
olcMaxDerefDepth: 15
olcReadOnly: FALSE
olcRestrict: add
olcRestrict: modify
olcRestrict: rename
olcRestrict: delete
olcRestrict: search
olcRestrict: compare
olcRestrict: extended
olcSyncUseSubentry: FALSE
olcMonitoring: FALSE
olcDbURI: “ldap://9.12.4.236”
olcDbStartTLS: none  starttls=no
olcDbRebindAsUser: FALSE
olcDbChaseReferrals: TRUE
olcDbTFSupport: no
olcDbProxyWhoAmI: FALSE
olcDbProtocolVersion: 3
olcDbSingleConn: FALSE
olcDbCancel: abandon
olcDbUseTemporaryConn: FALSE
olcDbConnectionPoolMax: 16
olcDbNoRefs: FALSE
olcDbNoUndefFilter: FALSE
```

Having this backend in place allows LDAP users to query our OpenLDAP server for objects on the z/VM LDAP server if they know the DN of the object or DIT. Next, we had to configure the rewrite capability to change a request for one DN into a request for another (according to proper configuration).

Configuring the OpenLDAP slapo_rwm overlay

We configured the slapo_rwm overlay to perform a rewrite of bind requests to objects under the `ou=helsinki,dc=itso,dc=ibm,dc=com` tree to bind against the corresponding object under `ou=cambridge_L,dc=itso,dc=ibm,dc=com` instead. Restricting the rewrite only to bind requests meant that we did not need to narrow the rewrite expression to individual parts of the DIT, but this is certainly possible (that is, you could rewrite only requests for `ou=People` under the `ou=helsinki` DIT if required). Example 3-29 shows the LDIF file that defines the configuration of slapo_rwm that we needed.

Example 3-29 LDIF file to configure the slapo_rwm overlay to rewrite bind requests

```
# LDIF for slapo_rwm overlay
# Added to our OpenLDAP server as:
# /etc/openldap/slapd.d/cn=config/olcDatabase={-1}frontend/olcOverlay={0}rwm.ldif
dn: olcOverlay={0}rwm
objectClass: olcOverlayConfig
objectClass: olcRwmConfig
olcOverlay: {0}rwm
olcRwmRewrite: {0}rwm-rewriteEngine on
olcRwmRewrite: {1}rwm-rewriteContext “bindDN”
olcRwmRewrite: {2}rwm-rewriteRule “^(.+,)?ou=helsinki,dc=itso,dc=ibm,dc=com$”
 “$1ou=cambridge_L,dc=itso,dc=ibm,dc=com” “:@”
olcRwmTFSupport: false
olcRwmNormalizeMapped: FALSE
```

With this configuration in place in the OpenLDAP server, we were able to log on to systems using our RACF password instead of the password from the OpenLDAP database.

Note: At first we were not able to log on as we expected using our RACF password. Logons were still using the password from the `userPassword` attribute in the OpenLDAP DIT.

The rewrite capability of the slapo_rwm overlay can be enabled or disabled using an `olcRwmRewrite` attribute setting of `rwm-rewriteEngine`. Our original LDIF did not have this attribute value set. We used phpLDAPadmin to look at the configuration of our slapo_rwm overlay, and found that OpenLDAP had added an `olcRwmRewrite` attribute with the `rwm-rewriteEngine` value automatically, but it was set to `rwm-rewriteEngine off`. Once we issued an LDIF update to the attribute to change `off` to `on`, our RACF password logons worked as expected.

This process is very effective and functional, but it does require the LDBM backend to be configured on z/VM LDAP. In addition, any user in the OpenLDAP directory that has to authenticate to RACF must also be defined to the z/VM LDBM database. While this could be automated, it is an additional overhead.

3.8.3 Configuring OpenLDAP to authenticate using z/VM SDBM

Using the SDBM backend of the z/VM LDAP server eliminates the definition overhead of having to duplicate LDAP objects from the OpenLDAP server into the z/VM LDAP server. However, because the schema used in SDBM cannot be changed and is different from the normal RFC2307-based schemas used in most LDAP directories, the configuration of the rewrite rule is more complex.

We used another capability of the slapo_rwm overlay called a rewrite map to perform an LDAP query to find the RACF ID that corresponds to the LDAP user. That ID is used to produce the DN to bind against in the SDBM backend, and the bind DN is rewritten using a rewrite rule similar to the LDBM scenario.

Updating the OpenLDAP schema

In 3.5.3, "Identifying the RACF account corresponding to the LDAP object" on page 67, we describe how the RACF user ID corresponding to an LDAP user has to be specified to make native authentication work with the z/VM LDAP server LDBM backend. When using slapo_rwm to rewrite LDAP binds to SDBM, a similar process must be performed. The safest way to do this is to add an attribute to the LDAP user objects in OpenLDAP that points to the RACF user ID.

Note: This additional attribute would not be required if it were possible to guarantee that the LDAP `uid` and the RACF user ID would always be the same.

To achieve this, we had to add some details to the schema of the OpenLDAP server. The schema update was required for two reasons:

- The `ibm-nativeId` attribute is not present in the OpenLDAP schema, so we had to add this to the schema so that we could use this attribute.
- OpenLDAP does validation of DNs, especially for RDN® fields. To make our subsequent rewrite rule work, the attributes used in the DNs we will bind against in the SDBM backend must be present in the OpenLDAP schema so that RDN validation succeeds.

The LDIF we used to update the OpenLDAP schema is shown in Example 3-30.

Example 3-30 OpenLDAP schema update LDIF for password validation to SDBM

```
dn: cn=ibm-nativeAuthentication,cn=schema,cn=config
cn: ibm-nativeAuthentication
objectClass: olcSchemaConfig
olcAttributeTypes: ( 1.3.18.0.2.4.187 NAME ( 'racfid' ) DESC 'Identifies the n
 ame of a z/OS Security Server userid or groupid' SINGLE-VALUE SYNTAX 1.3.6.1.
 4.1.1466.115.121.1.26 USAGE userApplications )
olcAttributeTypes: ( 1.3.18.0.2.4.186 NAME ( 'profileType' ) DESC 'Identifies
 the name of a z/OS Security Server profile' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1
 466.115.121.1.15 USAGE userApplications )
olcAttributeTypes: ( 1.3.18.0.2.4.1158 NAME ( 'ibm-nativeId' ) DESC 'Userid in
  the native security manager' EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX
  1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
olcObjectClasses: ( 1.3.18.0.2.6.264  NAME ( 'ibm-nativeAuthentication' ) DESC
  'Use native security manager for authentication' AUXILIARY SUP ( top ) MUST
 ( ibm-nativeId ) )
```

Configuring the OpenLDAP slapo_rwm overlay

We used two capabilities of the slapo_rwm overlay to make SDBM binds work. A rewrite map allows us to perform an LDAP query to find the RACF ID that corresponds to the LDAP user. That ID is then used in a rewrite rule to produce the DN to bind against in the SDBM backend. The configuration LDIF is shown in Example 3-31.

Example 3-31 LDIF for rewriting binds to a z/VM LDAP server SDBM backend

```
# LDIF for slapo_rwm overlay
# Added to our OpenLDAP server as:
# /etc/openldap/slapd.d/cn=config/olcDatabase={-1}frontend/olcOverlay={0}rwm.ldif
dn: olcOverlay={0}rwm
objectClass: olcOverlayConfig
objectClass: olcRwmConfig
olcOverlay: {0}rwm
olcRwmRewrite: {0}rwm-rewriteEngine on
olcRwmRewrite: {1}rwm-rewriteMap ldap dn2RACF “ldap://localhost/ou=helsinki,dc
 =itso,dc=ibm,dc=com?ibm-nativeId?sub?”
olcRwmRewrite: {2}rwm-rewriteContext “bindDN”
olcRwmRewrite: {3}rwm-rewriteRule “^(.+)?,ou=People,ou=helsinki,dc=itso,dc=ibm
 ,dc=com$” “racfid=${dn2RACF($1)},profiletype=user,ou=cambridge_S,dc=itso,dc=i
 bm,dc=com” “:@”
olcRwmTFSupport: false
olcRwmNormalizeMapped: FALSE
```

Configuring the OpenLDAP ldap database

We also had to define the OpenLDAP ldap database for the SDBM backend on z/VM. Even though it is almost identical to the LDBM version, for completeness we show the file here in Example 3-32.

Example 3-32 LDAP database configuration file for the SDBM backend in z/VM LDAP

```
dn: olcDatabase={2}ldap
objectClass: olcDatabaseConfig
objectClass: olcLDAPConfig
olcDatabase: {2}ldap
olcSuffix: ou=cambridge_S,dc=itso,dc=ibm,dc=com
olcAddContentAcl: FALSE
olcLastMod: FALSE
olcMaxDerefDepth: 15
olcReadOnly: FALSE
olcRestrict: add
olcRestrict: modify
olcRestrict: rename
olcRestrict: delete
olcRestrict: search
olcRestrict: compare
olcRestrict: extended
olcSyncUseSubentry: FALSE
olcMonitoring: FALSE
olcDbURI: “ldaps://9.12.4.236”
olcDbStartTLS: none  starttls=no
olcDbRebindAsUser: FALSE
olcDbChaseReferrals: TRUE
olcDbTFSupport: no
olcDbProxyWhoAmI: FALSE
olcDbProtocolVersion: 3
olcDbSingleConn: FALSE
olcDbCancel: abandon
olcDbUseTemporaryConn: FALSE
olcDbConnectionPoolMax: 16
olcDbNoRefs: FALSE
olcDbNoUndefFilter: FALSE
```

With these updates in place on our OpenLDAP server, we started the **slapd** process and attempted to authenticate.

Note: You can edit files in the `/etc/openldap/slapd.d/cn=config` directory. If you do, you need to be aware that OpenLDAP calculates a CRC checksum for any file it generates or updates, and if the checksum is present it will verify the file by recalculating the checksum. If you edit the file directly, you must remember to remove the checksum from the comment on the second line of the file. If you do not, OpenLDAP will reject the file because the checksum does not match.

Testing OpenLDAP authentication to an SDBM backend

We found that logging on to Linux (in our case, phpLDAPadmin) was successful using our RACF password. To verify the operation of slapo-rwm in our configuration, we enabled full debug in our OpenLDAP instance. Example 3-33 shows the important output from our OpenLDAP server during our phpLDAPadmin logon.

Example 3-33 OpenLDAP slapd debug log of bind rewrite to SDBM

```
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1006 op=1 do_bind
Sep 12 23:16:45 lnxslesa slapd[22077]: >>> dnPrettyNormal: <cn=Vic Cross,ou=People,ou=helsinki,
dc=itso,dc=ibm,dc=com>
Sep 12 23:16:45 lnxslesa slapd[22077]: <<< dnPrettyNormal: <cn=Vic Cross,ou=People,ou=helsinki,
dc=itso,dc=ibm,dc=com>, <cn=vic cross,ou=people,ou=helsinki,dc=itso,dc=ibm,dc=com>
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1006 op=1 BIND dn=”cn=Vic Cross,ou=People,ou=helsin
ki,dc=itso,dc=ibm,dc=com” method=128
Sep 12 23:16:45 lnxslesa slapd[22077]: do_bind: version=3 dn=”cn=Vic Cross,ou=People,ou=helsink
i,dc=itso,dc=ibm,dc=com” method=128
Sep 12 23:16:45 lnxslesa slapd[22077]: ==> rewrite_context_apply [depth=1] string=’cn=Vic Cross
,ou=People,ou=helsinki,dc=itso,dc=ibm,dc=com’
Sep 12 23:16:45 lnxslesa slapd[22077]: ==> rewrite_rule_apply rule=’^(.+)?,ou=People,ou=helsink
i,dc=itso,dc=ibm,dc=com$’ string=’cn=Vic Cross,ou=People,ou=helsinki,dc=itso,dc=ibm,dc=com’ [1
pass(es)]
. . .
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1007 op=0 SRCH base=”ou=helsinki,dc=itso,dc=ibm,dc=
com” scope=2 deref=0 filter=”(cn=vic cross)”
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1007 op=0 SRCH attr=ibm-nativeId
. . .
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1007 op=0 ENTRY dn=”cn=vic cross,ou=people,ou=helsi
nki,dc=itso,dc=ibm,dc=com“
Sep 12 23:16:45 lnxslesa slapd[22077]: <= send_search_entry: conn 1007 exit.
Sep 12 23:16:45 lnxslesa slapd[22077]: send_ldap_result: conn=1007 op=0 p=3
Sep 12 23:16:45 lnxslesa slapd[22077]: send_ldap_result: err=0 matched=”” text=””
Sep 12 23:16:45 lnxslesa slapd[22077]: send_ldap_response: msgid=1 tag=101 err=0
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1007 op=0 SEARCH RESULT tag=101 err=0 nentries=1 te
xt=
. . .
Sep 12 23:16:45 lnxslesa slapd[22077]: ==> rewrite_context_apply [depth=1] res={0,’racfid=VIC,p
rofiletype=user,ou=cambridge_S,dc=itso,dc=ibm,dc=com’}
Sep 12 23:16:45 lnxslesa slapd[22077]: [rw] bindDN: “cn=Vic Cross,ou=People,ou=helsinki,dc=itso
,dc=ibm,dc=com” -> “racfid=VIC,profiletype=user,ou=cambridge_S,dc=itso,dc=ibm,dc=com”
Sep 12 23:16:45 lnxslesa slapd[22077]: >>> dnPrettyNormal: <racfid=VIC,profiletype=user,ou=camb
ridge_S,dc=itso,dc=ibm,dc=com>
Sep 12 23:16:45 lnxslesa slapd[22077]: <<< dnPrettyNormal: <racfid=VIC,profileType=user,ou=camb
ridge_S,dc=itso,dc=ibm,dc=com>, <racfid=VIC,profileType=user,ou=cambridge_s,dc=itso,dc=ibm,dc=c
om>
```

```
. . .
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1006 op=1 BIND dn="racfid=VIC,profileType=user,ou=c
ambridge_S,dc=itso,dc=ibm,dc=com" mech=SIMPLE ssf=0
Sep 12 23:16:45 lnxslesa slapd[22077]: do_bind: v3 bind: "racfid=VIC,profileType=user,ou=cambri
dge_S,dc=itso,dc=ibm,dc=com" to "racfid=VIC,profileType=user,ou=cambridge_S,dc=itso,dc=ibm,dc=c
om"
Sep 12 23:16:45 lnxslesa slapd[22077]: send_ldap_result: conn=1006 op=1 p=3
Sep 12 23:16:45 lnxslesa slapd[22077]: send_ldap_result: err=0 matched="" text=""
Sep 12 23:16:45 lnxslesa slapd[22077]: send_ldap_response: msgid=2 tag=97 err=0
Sep 12 23:16:45 lnxslesa slapd[22077]: conn=1006 op=1 RESULT tag=97 err=0 text=
```

The first set of messages of interest is when **slapd** receives the bind request from phpLDAPadmin to authenticate the user logging on. This is seen as “conn=1006” to **slapd**. In this first group of messages we see that slapo_rwm identifies that the rewrite rule applies and the rewrite processing begins. The rewrite rule includes a call to the rewrite map, which causes the second group of log messages as slapd searches for the required `ibm-nativeId` attribute (this is “conn=1007”).

The third group of messages shows the result of the rewrite rule having rewritten the bind DN. The rewrite map has obtained the value `VIC` as the contents of the `ibm-nativeId` attribute for our user object in OpenLDAP, and the rewrite rule has generated the appropriate bind DN to use for authentication against the SDBM backend. The last group of messages shows that **slapd** processed the rewritten bind and returned the success result back to phpLDAPadmin.

3.8.4 RACF password management with OpenLDAP slapo_rwm

Since the rewrite rules used in both the LDBM and SDBM backend configurations simply redirect bind requests to the z/VM LDAP server, the RACF password change syntax described in “Password management with Linux and the z/VM LDAP server” on page 81 still applies.

If the z/VM LDAP backend is LDBM with native authentication, managing the RACF passwords of users via LDAP will function in the same way whether the LDBM backend is accessed directly or whether the backend is accessed from the OpenLDAP server via slapo_rwm.

If the z/VM LDAP backend is SDBM, again, users can manage their passwords by accessing the OpenLDAP server or the SDBM backend directly. However, if they manage their password via OpenLDAP they do not have to know any details of their RACF ID (such as their RACF user ID, or the DN format to use to access their RACF account).

To verify this, we modified our PHP change password utility (described in “External password change mechanism” on page 83) to direct the password change request to the OpenLDAP server, and the base DN of the OpenLDAP database, instead of the z/VM LDAP server. The utility functioned exactly the same as it did when it directed password changes to the LDBM directly.

3.9 Centralizing Linux audit information with z/VM RACF

Similar to z/VM auditing, Linux has auditing capabilities that help an administrator to analyze what is going on in a system. These auditing capabilities are part of the Linux Auditing Framework, made up of an audit-enabled kernel (shipped as part of the standard distributions

SLES 11 and RHEL 5.4), auditing daemon, dispatchers, and user-space commands to run the audit. Figure 3-15 depicts this landscape.

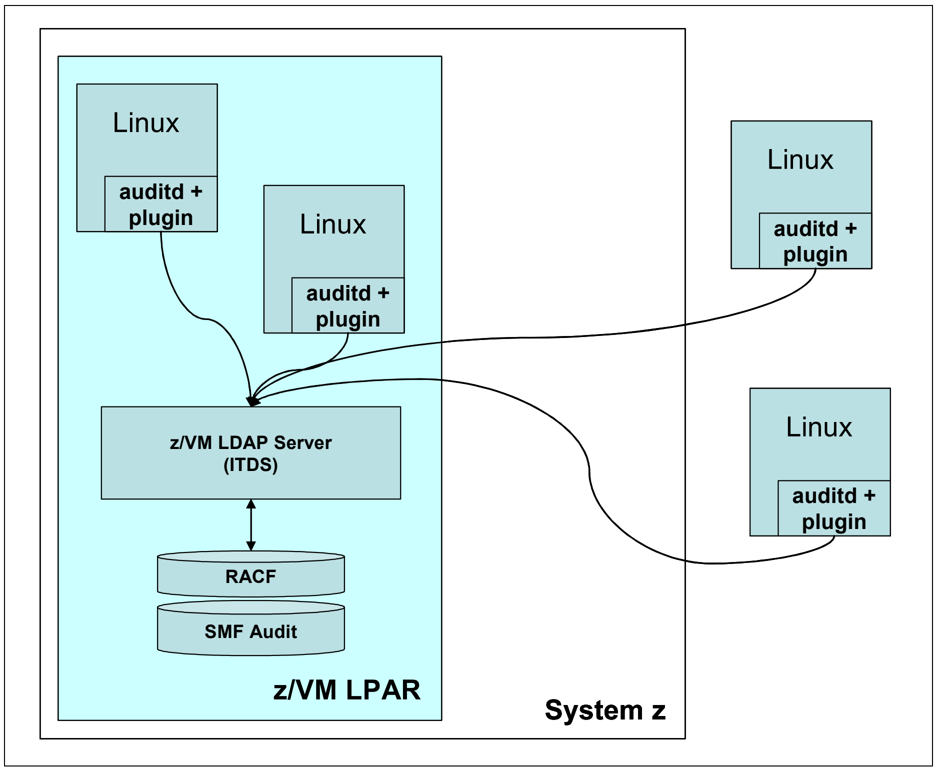


Figure 3-15 Centralized Linux auditing with z/VM LDAP server and RACF

The auditing rules are defined in the `/etc/audit/audit.rules` file. Several examples are provided in the `/usr/share/doc/packages/audit` location.

The auditing daemon is in charge of gathering the auditing data, writing it to log files (`/var/log/audit/audit.log`), and, if configured to do so, dispatching it to other subsystems. Dispatcher audisp-zos-remote is provided as part of auditd distribution. This dispatcher is used to write Linux audit data as z/VM RACF SMF records, using the LDAP server extended operations support.

The audispd-zos-remote dispatcher man page describes the setup for interacting with the z/OS LDAP server and z/OS RACF. For more information, refer to:

`http://linux.die.net/man/8/audispd-zos-remote`

The setup in the following sections is an attempt to adapt the configuration to work with z/VM LDAP server and z/VM RACF:

- 3.9.1, “Enabling extended operations support in z/VM LDAP server” on page 93
- 3.9.2, “RACF configuration” on page 94

3.9.1 Enabling extended operations support in z/VM LDAP server

The z/VM LDAP service provides two services that enable audit and authentication requests to be resolved using z/VM Security Server. These services are provided by using z/VM LDAP server extended operations (XOP). Remote auditing and authentication requests are handled by an LDAP component called ICTX, which must be activated in the configuration as shown in Example 3-34 on page 94.

Example 3-34 LDAP configuration for remote services

```
#ICTX extended operations support section
plugin clientOperation ITYBIC31 ICTX_INIT “CN=ICTX”
```

3.9.2 RACF configuration

Next, we created a RACF user, shown in Example 3-35, that will be used for binding to the LDAP server. This user does not have to exist in the directory. It must, however, be a valid RACF definition. We specify here, for the time of the test, that the password does not expire.

Example 3-35 Creating binduser in RACF

```
rac adduser binduser
rac alu binduser password(s0leil) noexpire
```

To recognize that a client is authorized to use remote service, it must have a specific RACF authorization. This is discussed in the “Remote authorization and auditing through LDAP” chapter in *z/VM: TCP/IP Programmer’s Reference* (for V5R4.0), SC24-6126.

For remote auditing, the binduser must have at least read access to the IRR.LDAP.REMOTE.AUTH profile of class FACILITY. This profile must be defined before any other operations. Example 3-36 details the commands.

Example 3-36 Create required profile and authorize binduser

```
rac rdefine facility irr.ldap.remote.audit uacc(none)
rac permit irr.ldap.remote.audit cl(facility) id(binduser) acc(read)
```

When the user for the LDAP binding is created and properly authorized to access the resources, an **ldapsearch** command can be issued from a Linux virtual machine to make sure the credentials are correct, as demonstrated in Example 3-37.

Example 3-37 Checking user binding

```
[root@lnxrh1 ~]# ldapsearch -H ldap://9.12.4.189 -x -v -D racfid=binduser,cn=ictx
-w s0leil -b "cn=ictx" objectclass=*
ldap_initialize( ldap://9.12.4.189 )
filter: objectclass=*
requesting: All userApplication attributes
# extended LDIF
#
# LDAPv3
# base <cn=ictx> with scope subtree
# filter: objectclass=*
# requesting: ALL
#

# search result
search: 2
result: 53 Server is unwilling to perform

# numResponses: 1
```

This **ldapsearch** command uses these flags:

- -H: Specifies the URI of our LDAP server.
- -x: Requests to use simple authentication.
- -v: Runs in verbose mode.
- -D: Specifies the Distinguished Name to use to bind to the LDAP server. According to the documentation, it must be racfid=binduser,cn=ictx. The user *binduser* might, however, be different in your configuration.
- -w: Specifies the password to use for the bind.
- -b: Specifies cn=ictx as the base for the search in the LDAP.
- objectclass=*: Returns all the objects found.

The **ldapsearch** command returns code 53, which states that the server will not perform the request. Anyway, the bind was successful, the server recognized our credentials.

Note: If the credentials are not recognized, **ldapsearch** would have failed this way:

```
lnxsu1:~ # ldapsearch -H ldap://9.12.4.189 -x -v -D racfid=binduser,cn=ictx -w
toto -b "cn=ictx" objectclass=*
ldap_initialize( ldap://9.12.4.189:389/??base )
ldap_bind: Invalid credentials (49)
        additional info: ITYXOR03 Password is incorrect, or userID is
undefined/revoked
```

3.9.3 Adding the @LINUX class to RACF

The dispatcher's man page specifies that the plug-in will use a dedicated RACF class, `@LINUX`, for all the events processed. Because RACF does not ship this class by default, its configuration has to be updated. Under z/OS, this is made thanks to a dynamic class description table (CDT). Because dynamic CDT does not exist in z/VM, we are going to update z/VM RACF CDT statically, adding our `@LINUX` class to the configuration and recompiling the updated LOADLIB.

Note: As a prerequisite to recompiling LOADLIBs, you must have the High Level Assembler (HLASM) product installed and configured.

The z/OS CDT class is defined with the command shown in Example 3-38.

Example 3-38 z/OS RACF @LINUX class definition

```
rac rdefine cdt @LINUX cdtinfo(posit(493) FIRST(alpha,national,numeric,special)
OTHER(alpha,national,numeric,special) RACLIST(REQUIRED) CASE(asis)
generic(allowed) defaultuacc(none) maxlength(246))
```

The procedure to add a class to RACF CDT is a two-step process:

1. Add an entry to the Class Descriptor table.
2. Add a corresponding entry to the RACF Router Table.

For more information about this, see *z/VM: RACF Security Server System Programmer's Guide* (for V5R4), SC24-6149 starting with the chapter entitled "Specifying Resource Class Options". Example 3-39 on page 96 is the assembler code we used to add the `@LINUX` class to RACF VM, based on the arguments of the z/OS command.

Example 3-39 ICHRRCDE ASSEMBLE file

```
ICHRRCDE ASSEMBLE E2  F 80  Trunc=71 Size=26 Line=13 Col=1 Alt=0
===== * * * Top of File * * *
===== */***************************************************************/
===== */* LICENSED MATERIALS - PROPERTY OF IBM                        */
===== */*                                                             */
===== */* 5694-A01                                                    */
===== */*                                                             */
===== */* (C) COPYRIGHT IBM CORP. 1991, 2007                          */
===== */* US GOVERNMENT USERS RESTRICTED RIGHTS - USE,                */
===== */* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP             */
===== */* SCHEDULE CONTRACT WITH IBM CORP.                            */
===== */*                                                             */
===== */* STATUS = HLE7740                                            */
===== */***************************************************************/
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7>.
===== ICHRRCDE CSECT
===== @LINUX   ICHERCDE CLASS=@LINUX,                                    X
=====                ID=130,                                             X
=====                POSIT=493,                                          X
=====                MAXLNTH=246,                                        X
=====                FIRST=ANY,                                          X
=====                OTHER=ANY,                                          X
=====                OPER=YES,                                           X
=====                RACLIST=ALLOWED,                                    X
=====                GENLIST=ALLOWED,                                    X
=====                DFTUACC=NONE
=====          ICHERCDE
=====          END
===== * * * End of File * * *
====>
                                                     X E D I T  1 File
```

Note: The z/OS `@LINUX` class definition sets a RACLIST=REQUIRED argument. With this value, HLASM exits with a non-zero return code. That is why we modified it to read RACLIST=ALLOWED. This file must have a file type of ASSEMBLE. When a line is longer than 72 characters, an X must be placed as continuation mark on the line at position 72. The position of the statements in the file is also important.

To get the proper formatting, copy an existing `*ASSEMBLE` file and update it accordingly.

Follow the steps in *z/VM: RACF Security Server System Programmer's Guide* (for V5R4), SC24-6149 to assemble the modified file and put it into production. With the addition of the new class, z/VM RACF is configured the same way as z/OS RACF. The class can be activated, and profiles can be added, as detailed in Example 3-40.

Example 3-40 Activating @LINUX class, and adding profiles to it

```
rac setropts classact(@LINUX)
Ready(00004); T=0.01/0.01 15:05:18
rac setropts list
ATTRIBUTES = INITSTATS NOWHEN(PROGRAM) SAUDIT CMDVIOL NOOPERAUDIT
STATISTICS = NONE
AUDIT CLASSES = NONE
ACTIVE CLASSES = DATASET USER GROUP VMMDISK VMRDR VMCMD VMBATCH VMLAN
```

```
                  FACILITY SURROGAT XFACILIT GXFACILI @LINUX
GENERIC PROFILE CLASSES =  NONE
[...]
rac rdefine @LINUX * uacc(none) audit(all(read))
```

The z/VM configuration steps are now completed.

3.9.4 Linux configuration

As described earlier, the Linux auditd daemon must be configured to send the audit data to the z/VM LDAP server.

Configure the audispd-zos-remote plug-in

The z/OS dispatcher plug-in must be activated prior to use. To do so, uncomment the `active` statement in the configuration file `/etc/audisp/plugins.d/audispd-zos-remote.conf` and set it to `yes`, as shown in Example 3-41. Also make sure the path to the dispatcher is correct.

Example 3-41 The /etc/audisp/plugins.d/audispd-zos-remote.conf file

```
# This is the configuration for the audispd-zos-remote
# audit dispatcher plugin - See audispd(8)
#
# Note that this specific plugin has a configuration file of
# its own. The complete path for this file must be entered as
# the argument for the plugin in the 'args' field below
# See audispd-zos-remote(8)

active = yes
direction = out
path = /usr/local/sbin/audispd-zos-remote
type = always
args = /etc/audisp/zos-remote.conf
format = string
```

Then, configure the dispatcher by editing the `/etc/audisp/zos-remote.conf` file and fill in the fields with the information relevant to your setup, as shown in Example 3-42.

Example 3-42 The /etc/audisp/zos-remote.conf file

```
## This is the configuration file for the audispd-zos-remote
## Audit dispatcher plugin.
## See zos-remote.conf(5) for more information

server = 9.12.5.94
port = 389
user = binduser
password = s0leil
timeout = 1
q_depth = 64
```

User credentials are the ones that were created in 3.9.2, “RACF configuration” on page 94.

The last step is to instruct auditd to use the z/OS dispatcher, by modifying the `dispatcher` statement in the `/etc/audit/auditd.conf` file. See Example 3-43 on page 98.

Example 3-43 The /etc/audit/auditd.conf file

```
#
# This file controls the configuration of the audit daemon
#

log_file = /var/log/audit/audit.log
log_format = RAW
log_group = root
priority_boost = 4
flush = INCREMENTAL
freq = 20
num_logs = 4
disp_qos = lossy
dispatcher = /usr/local/sbin/audispd-zos-remote
name_format = HOSTNAME
##name = mydomain
max_log_file = 5
max_log_file_action = ROTATE
space_left = 75
space_left_action = SYSLOG
action_mail_acct = root
admin_space_left = 50
admin_space_left_action = SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
##tcp_listen_port =
tcp_listen_queue = 5
##tcp_client_ports = 1024-65535
tcp_client_max_idle = 0
```

3.10 Using z/VM LDAP in an SSI cluster

When running in an SSI configuration, the LDAP server SVM is defined as a multiple configuration user ID. This means that each SSI member can run its own LDAP server. This is not required, but it does create a convenient way to run a distributed LDAP environment with high availability within your SSI cluster.

3.10.1 LDAP server high availability

LDAP servers are often critical to many company IT systems today. To provide the required quality of service for such a critical function, LDAP servers are often configured across multiple physical or virtual operating system instances. The content of the LDAP data stores is then replicated between the instances using clustering (either at the operating system or LDAP server level).

Most LDAP servers implement a method of replicating content between instances. Approaches can vary, from methods that work at the application layer down to disk and file system mirroring.

LDAP application-based replication

These methods utilize operations performed by the LDAP server application process using LDAP operations. In OpenLDAP versions up to 2.1, a process called **slurpd** ran on a backup server and performed an update to its managed replicated copy when the master database was altered. The **slurpd** process acted like an LDAP client to the master server.

Note: Microsoft Active Directory uses a similar pull-based replication technology for replication of site data.

In OpenLDAP 2.2, replication became a push-based mechanism driven by the master server. Replica servers are configured to accept replication data from particular master servers, and those master servers use LDAP update transactions to copy their updates to the replicas. One of the advantages of this approach is that a replica server can be a backup for several master servers.

When replicating LDAP databases, usually a single master server holds the repository of all data for the DIT. Replica servers can handle queries on behalf of the master, but if they are asked to perform an update they will issue a *referral*, which instructs the client to connect to the master server to perform its update. Some LDAP servers also support "multi-master" replication, where all the servers in a replicated LDAP collection can accept updates and send those updates to the other servers. Multi-master replication is available in recent versions of OpenLDAP, and also in other servers such as 389 Directory Server, IBM Tivoli Directory Server, Novell eDirectory, and others.

The z/VM LDAP server implements the push-based replication capability, but does not currently support multiple master databases.

Server system replication

For LDAP servers that do not implement application-level replication, server clustering technologies can be used to provide database replication. Technologies such as Distributed Remote Block Device (DRBD) or other physical disk replication technologies such as IBM Metro Mirror or Global Mirror, can be used to replicate physical disk updates to a remote location. These updates can be used to recover a copy of the LDAP datastore if the primary server is lost due to hardware or site failure.

3.10.2 z/VM LDAP server clustering for the LDBM backend

The z/VM LDAP server provides a replication capability which can be used to create a distributed LDAP service. The configuration can be used with or without an SSI cluster, but the user definitions that already exist in z/VM for the LDAP server SVMs are easily used to create a clustered configuration.

To enable replication of the z/VM LDAP server, two sets of configuration changes must be made:

- ► The servers receiving replicated content must be configured with a DN and password to use to contact the master server for replication.
- ► Replication objects are configured in the master server to represent the servers that LDAP content is to be replicated to.

Example 3-44 on page 100 shows the DS CONF changes used on the replica LDAP servers to specify the DN and password for replication.

Example 3-44 DS CONF changes for cluster operation on the second and subsequent SSI members

```
adminDn "cn=Admin,dc=itso,dc=ibm,dc=com"
allowAnonymousBinds on
audit on
audit all,add+delete
audit error,modify+search+bind
audit none,unbind
commThreads 10
listen ldap://:389
listen ldaps://:636
listen ldap://:pc
maxConnections 4096
sendV3StringsOverV2As UTF-8
sizeLimit 500
timeLimit 3600
validateIncomingV2strings on

sslAuth serverAuth
sslCertificate LDAP-6
sslKeyRingFile /etc/ldap/key.kdb
sslKeyRingPWStashFile /etc/ldap/key.sth

database LDBM GLDBLD31
suffix "ou=cambridge_L,dc=itso,dc=ibm,dc=com"
masterServer ldap://9.12.4.236:389
masterServerDN “cn=Admin,dc=itso,dc=ibm,dc=com”
masterServerPW secret
nativeAuthSubtree all
nativeUpdateAllowed on
useNativeAuth all

database SDBM GLDBSD31
suffix "ou=cambridge_S,dc=itso,dc=ibm,dc=com"
```

If the master LDAP server already has LDAP content in the database, you need to copy the database to the replica server before starting replication. If you know where the LDAP database is stored in the BFS (the default location is */var/lib/ldap*), you could copy the database files to the replica. It is much safer and more reliable, however, to generate an LDIF file of the database contents using **ldapsearch**, transfer the LDIF file to the replica, and use **ldapadd** or **ldapmodify** to add the contents to the replica. You should also ensure that the schema of the master and replica servers are synchronized; and schema updates you have applied to the master must be applied to any replicas.

Once the replica LDAP server SVM is started, the replication objects can be added to the master server to start the replication relationship.

Note: For more detail about configuring and managing replication in the z/VM LDAP server, refer to *z/VM TCP/IP LDAP Administration Guide (version 6 release 2)*, SC24-6236-01.

3.10.3 z/VM LDAP server clustering for the SDBM backend

When RACF is enabled in an SSI cluster, it is a requirement that all the cluster members share a RACF database. For the z/VM LDAP server, this means that the datastore of the SDBM backend is already shared and synchronized between all the members of the SSI cluster. In practical terms, this means that z/VM LDAP will automatically provide a replicated multi-master LDAP service across all LDAP servers in an SSI cluster. No additional configuration is needed to provide this capability.

By combining z/VM LDAP using SDBM, live guest relocation, and the OpenLDAP slapo_rwm technique shown in “Using an OpenLDAP server with the z/VM LDAP server” on page 86 we can build a high-performance LDAP capability with the flexibility of OpenLDAP and the security of RACF. The proposed implementation is shown in Figure 3-16 based on a two-member SSI cluster, although any size SSI cluster can be used.

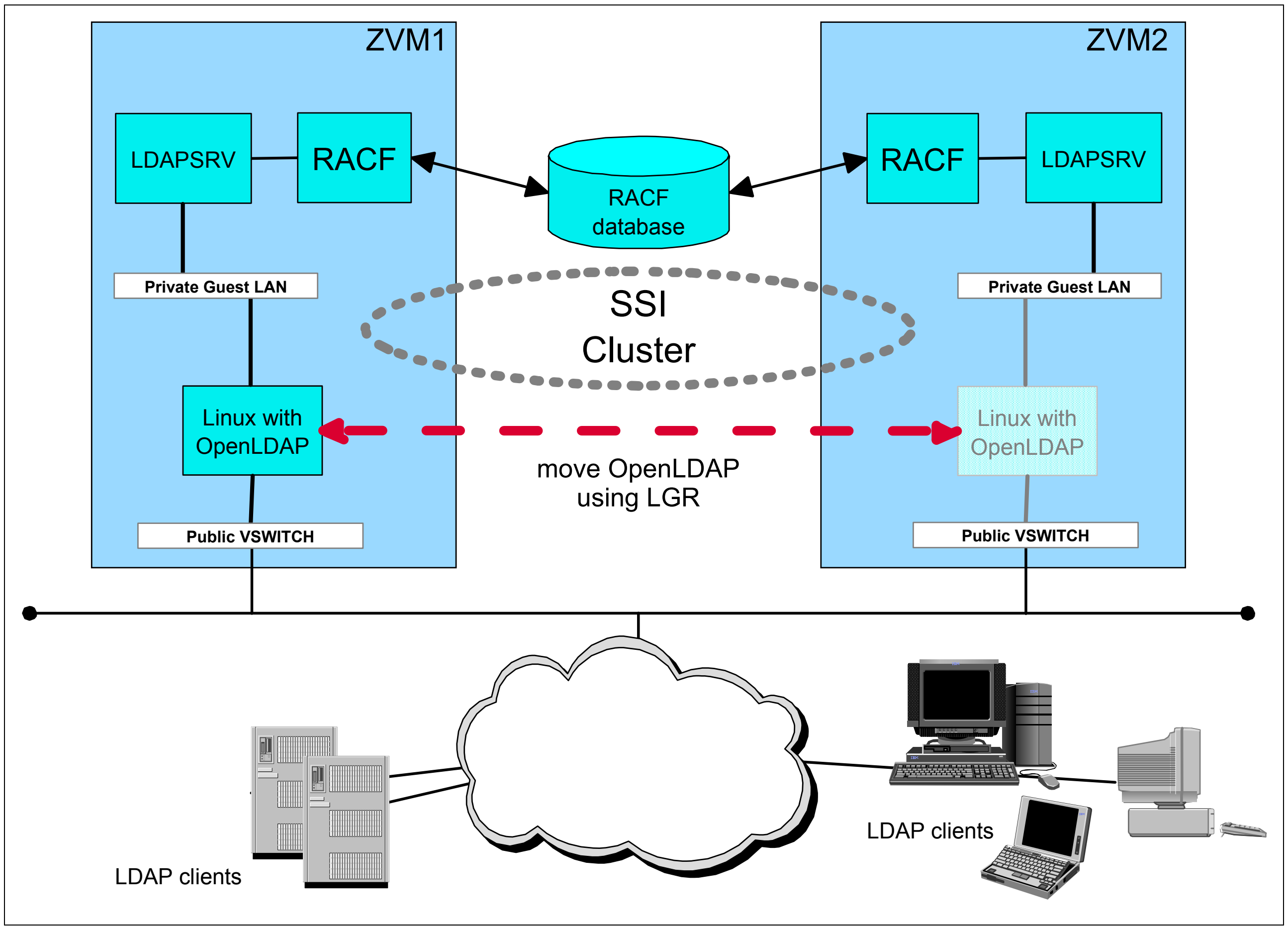


Figure 3-16 Combining z/VM LGR, z/VM LDAP with SDBM, and OpenLDAP

We started to implement this solution in our two-node SSI cluster, but encountered an issue. A connection to a z/VM Guest LAN will prevent a Linux guest from being relocatable, because a Guest LAN only has scope within a single z/VM system. If we were to implement our design as illustrated we would have to include a way to disconnect the Guest LAN NICs before relocation and reconnect them afterward, which is not a convenient way to provide a high availability LDAP service.

There are other ways to implement the connection from OpenLDAP to z/VM LDAP. Some of these include:

- A private VLAN on a VSWITCH connected to OSAs that span the z/VM systems. This could be the same VSWITCH that provides the primary network connection to z/VM and the Linux guests (the “Public VSWITCH” in the diagram) or another one.
- If the z/VM systems are in the same CPC, a HiperSockets™ CHPID accessible from all LPARs. If the z/VM systems are not in the same CPC, the HiperSockets Bridge function can be used to connect the HiperSockets networks in the different CPCs to each other through a shared LAN segment.

These methods have the drawback that they would not support both z/VM LDAP servers presenting with the same IP address. This means that the configuration of the OpenLDAP server will have to support connecting to the multiple z/VM LDAP servers using discrete IP addresses for each, which could result in delays when all of the z/VM systems are not active.

The ideal solution would be a high-availability capability built in to z/VM TCP/IP. For example, a Virtual Router Redundancy Protocol (VRRP) implementation in CMS would allow a single IP address to be managed across all the z/VM TCP/IP stacks in the SSI cluster. Alternatively, load balancing techniques such as those implemented in the Linux-HA Project for Linux could even provide distribution of LDAP requests between the z/VM LDAP servers.

Such an implementation would have to be based on more than network data in order to have an awareness of the “closest” z/VM server to direct the traffic to. If all the z/VM systems share the same flat Layer 2 network that the Linux guests are attached to, all the z/VM TCP/IP servers will appear to be the same network distance away from a Linux guest. However, the z/VM server that is on the same z/VM LPAR as the Linux guest will actually be closest since no network traversal would be needed to reach it. Some of the advanced packet-mangle and routing capabilities of the Linux kernel may offer techniques to solve this issue.

4

Authentication and access control

The process of access control is critical. It defines how users and systems can communicate, and their approach. Access control refers to controlling or limiting access to system resources, including data, and thus protects information from unauthorized access.

Authentication of Linux users with a central LDAP directory instead of using information stored locally on the system (in `/etc/passwd` and in `/etc/shadow`) is a usual practice to reduce the administration effort in an environment with multiple Linux systems.

In order to use LDAP for user authentication, you must install and configure pam_ldap, nss_ldap module, and the LDAP client, and configure each service that should participate in LDAP authentication (SSH, Telnet, sudo, FTP, and so forth). The user information must be available in the LDAP database to create a central user repository.

System users are assigned to groups, and they have the privileges of whichever group they are members.

In this chapter, we describe the authentication and access methods that are used for Linux on System z.

4.1 SELinux

Security-Enhanced Linux (SELinux) on Red Hat Enterprise Linux (RHEL) is an implementation of mandatory access control (MAC) using Linux Security Modules (LSM) in the 2.6 Linux kernel, based on the principle of least privilege. It is not a Linux distribution, but rather a set of modifications that can be applied to UNIX-like operating systems, such as Linux. Standard Linux security is a discretionary access control model (DAC).

DAC, being standard Linux security, provides no protection from broken software or malware running as a normal user or root. Users can grant risky levels of access to files they own.

MAC provides full control over all interactions of software. Administratively defined policy closely controls user and process interactions with the system, and can provide protection from broken software or malware running as any user. Figure 4-1 shows a schematic overview of SELinux.

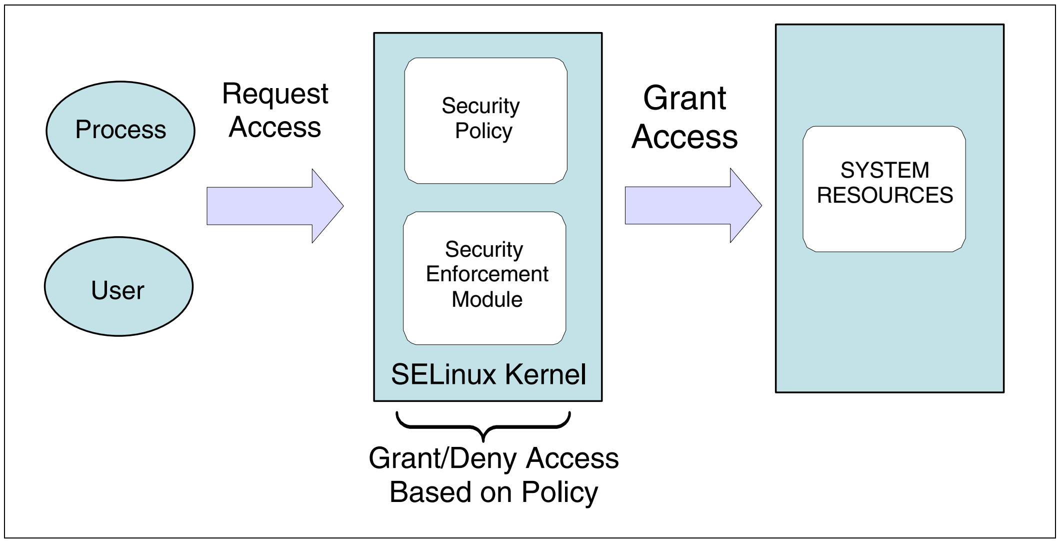


Figure 4-1 Schematic overview of SELinux

4.1.1 Important files and directories for SELinux

The essential files and directories used for SELinux are:

- `/etc/selinux/config`

  This is the configuration file for SELinux (see Example 4-1).

Example 4-1 Sample configuration file for SELinux

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=enforcing
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

- `/etc/selinux/<policy name>/rpolicy`

  This directory has runtime configuration files and binary policies. The `<policy name>` is the name of the policy according to the requirements of the administrator.

- `/etc/selinux/<policy name>/src/policy`

  This is a policy source file where `<policy name>` is the name of the policy according to the requirements of the administrator. More than one policy can exist on the system, but only one can be loaded at a time.

4.1.2 Enabling SELinux

SELinux is enabled in *permissive* mode on RHEL 4 systems. On RHEL 5 and RHEL 6 systems, it is enabled by default in *enforcing* mode.

However, enforcing such a policy-based security model comes at a price. Every access to a system resource by a user or process such as an I/O device must be controlled by SELinux. The process of checking permissions can cause overhead.

SELinux is of great value to any border server such as a firewall or a web server, but the added level of security on a backend database server might not justify the loss in performance. Alternatively, a backend database server might be *exactly* the kind of server that your organization's security policy dictates that SELinux should be enabled on, and any performance will have to be taken into account when sizing the server.

Special considerations for using SELinux on System z

To enable support of SELinux on System z, additional steps are required. Linux on System z supplies a prebuilt binary SELinux policy for each supported version of RHEL 4 and a set of binary policy modules for RHEL 5 and RHEL 6. To get the current mode of SELinux, the command **`getenforce`** can be used, as shown in Example 4-2.

Example 4-2 The getenforce command output in Red Hat Enterprise Linux 6.2

```
[root@lnxrh60a ~]# getenforce
Enforcing
```

The **`sestatus`** tool is used to check the status of a system running SELinux. This tool provides information about whether SELinux is enabled or disabled, its loaded policy and whether it is in enforcing mode or permissive mode, as shown in Example 4-3.

Example 4-3 The sestatus command output in Red Hat Enterprise Linux 6.2

```
[root@lnxrh60a ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
```

The **`sestatus`** command can be used to display the processes and security context of files in `/etc/sestatus.conf`, as shown in Example 4-4 on page 106.

Example 4-4 Sample sestatus configuration file /etc/sestatus.conf

```
[files]
/etc/passwd
/etc/shadow
/bin/bash
/bin/login
/bin/sh
/sbin/agetty
/sbin/init
/sbin/mingetty
/usr/sbin/sshd
/lib/libc.so.6
/lib/ld-linux.so.2
/lib/ld.so.1

[process]
/sbin/mingetty
/sbin/agetty
/usr/sbin/sshd
```

Additionally, policy source files are provided for users who want to use custom policies. For details of how custom policies are supported, see 4.1.4, "Policies" on page 110.

You must enable SELinux in the kernel so that Linux on System z can take advantage of SELinux features. The following briefly examines the steps here. See your RHEL documentation for more details about the steps.

To enable SELinux:

1. Confirm the current mode of SELinux by using the **getenforce** command, as shown in Example 4-2 on page 105.

   If SELinux is disabled, the RHEL configuration file shows the `SELINUX=disabled` setting in `/etc/selinux/config`, as shown in Example 4-5.

   Example 4-5 Sample configuration file with SELinux disabled

   ```
   # This file controls the state of SELinux on the system.
   # SELINUX= can take one of these three values:
   #       enforcing - SELinux security policy is enforced.
   #       permissive - SELinux prints warnings instead of enforcing.
   #       disabled - SELinux is fully disabled.
   SELINUX=disabled
   # SELINUXTYPE= type of policy in use. Possible values are:
   #       targeted - Only targeted network daemons are protected.
   #       strict - Full SELinux protection.
   SELINUXTYPE=targeted
   ```

   Before SELinux is enabled, each file on the system must be labeled with an SELinux context, otherwise restricted domains can be denied access, thus preventing the system from booting correctly. To avoid this issue, configure the setting `SELINUX=permissive` in `/etc/selinux/config` as shown in Example 4-6 on page 106.

   Example 4-6 Sample configuration file with SELinux set to print warnings

   ```
   # This file controls the state of SELinux on the system.
   # SELINUX= can take one of these three values:
   ```

```
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=permissive
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

After setting SELinux to `permissive`, run the **`reboot`** command to restart the system as the root user. When the system is online, before changing from permissive to enforcing mode, verify that no error messages are shown in the `/var/log/messages` file to ensure that SELinux did not deny actions during the last boot sequence. The SELinux policies are not enforced when the mode is set to `permissive`.

If no error messages are present in the `/var/log/messages` file, configure SELinux to `enforcing` in the `/etc/selinux/config` file as shown in Example 4-7.

Example 4-7 Sample configuration file showing SELinux is in enforcing mode

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=enforcing
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

2. After rebooting the system, confirm the current mode of SELinux using the **`getenforce`** command, as shown in Example 4-2 on page 105, and ensure that it returns `Enforcing`.
3. Install the appropriate binary policy. The supported binary policies are provided in the SELinux subdirectory of the Linux on System z installation. For each supported RHEL release, there is a directory containing the files specific to its version. Select the appropriate directory for the RHEL release of your machine, and copy the target policy into the `/etc/selinux/targeted/policy` directory.
4. Install the appropriate file contexts. Accompanying each binary policy is a set of file contexts listing how the file system should be labelled. For each RHEL release, there is a `file_contexts` file in the same directory as the policy. This file must be placed in the `/etc/selinux/targeted/contexts/files/` directory, for example:

   `/etc/selinux/targeted/contexts/files/file_contexts`
5. Load the new policy. Having installed the policy files, the new policy can be enabled by executing the command:

   For RHEL 5 and RHEL 6 (Policy file argument is no longer supported, installed policy is always loaded):

   ```
   [root@lnxrh60a /]# /sbin/load_policy
   ```
6. Ensure that Linux files are correctly labelled. To reset the labels for Linux files and directories based on the new policy, invoke a full file system relabel by using the **`e2label`** command.
7. Enable Linux domain transitions in the SELinux configuration file. To do this, add an entry to `/etc/selinux/config` as follows:

```
ENABLE_SELINUX_TRANSITIONS=y
```

For dynamic changes to the system, you can use **`setenforce 0`** or **`1`** for *Permissive* or *Enforcing* respectively, as shown in Example 4-8.

Example 4-8 Sample output of setenforce and getenforce in RHEL version 6.2

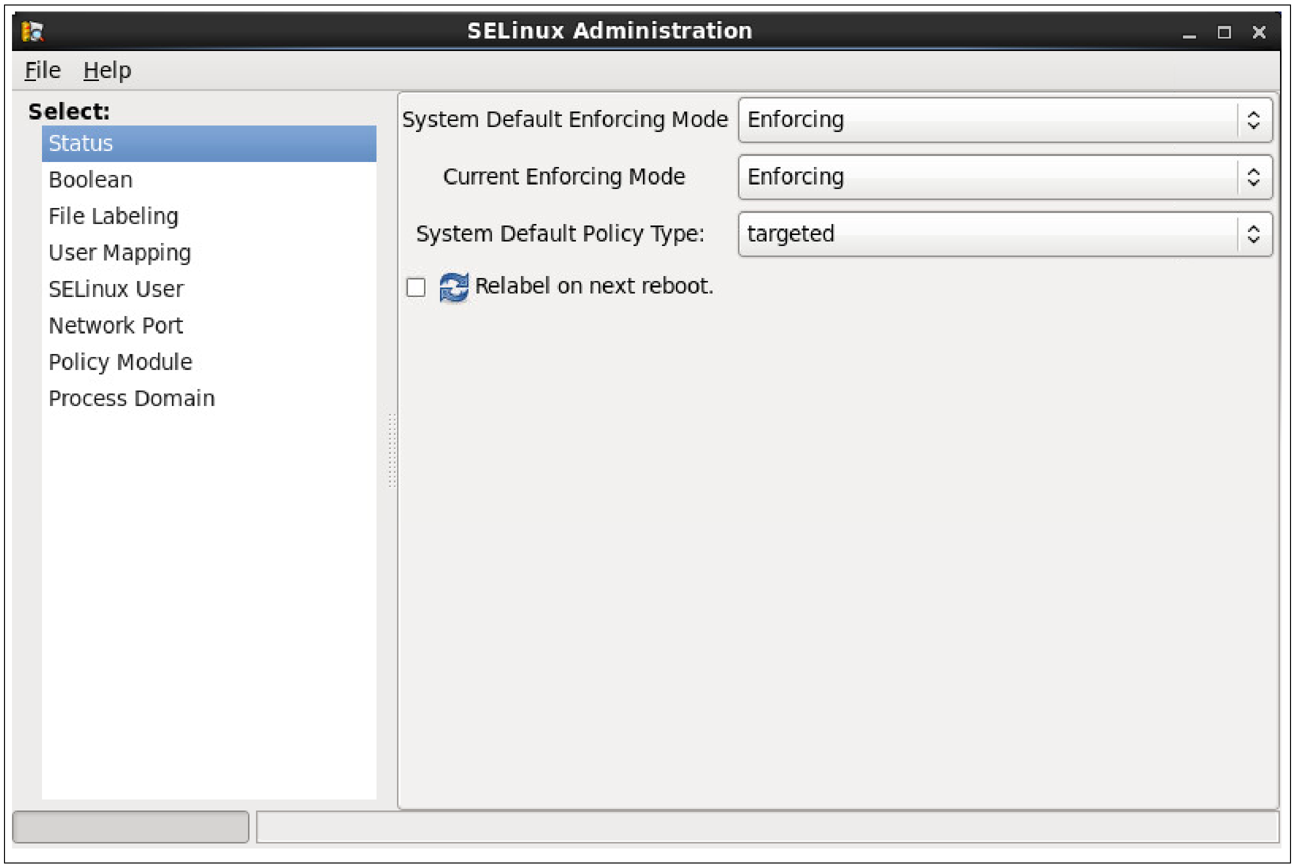
```
[root@lnxrh60a /]# setenforce 1
[root@lnxrh60a /]# getenforce
Enforcing
```

The **`system-config-securitylevel-tui`** command is deprecated in RHEL 6; if you need to verify and edit the security level that is presently being used on the machine, start the SELinux managing tool by issuing the **system-config-selinux** command via a VNC session or manually editing the configuration file (`/etc/sysconfig/selinux`).

```
[root@lnxrh60a ~]# system-config-selinux
```

Note: To have a graphical tool for managing SELinux, you need to install the policycoreutils-gui package.

See Figure 4-2 on page 108 for an example of running this command.



System Default Enforcing Mode: Enforcing
Current Enforcing Mode: Enforcing
System Default Policy Type: targeted
☐ Relabel on next reboot.

Figure 4-2 The system-config-selinux command in Red Hat Enterprise Linux 6.2

Securing FTP using SELinux

The Very Secure FTPD (VSFTPD) daemon is used for secure, fast, and stable data transfer. Its purpose is to transfer files between computer hosts on a network without requiring the user to log directly into the remote host or have knowledge of how to use the remote system. It allows users to access files on remote systems using a standard set of simple commands.

The SELinux policy defines how VSFTPD interacts with processes and files. SELinux prevents VSFTPD from accessing user home directories by default, so you must follow these steps to enable this access:

1. Enable authenticated users to log in by editing the `/etc/vsftpd/vsftpd.conf` file as the root user. Uncomment the `local_enable=YES` option.
2. Start the VSFTPD service, as shown in Example 4-9.

   Example 4-9 Restarting the VSFTPD service

   ```
   service vsftpd start
   Starting vsftpd for vsftpd:                                [  OK  ]
   ```

3. Log in with a valid user name and password, as shown in Example 4-10.

   Example 4-10 Sample FTP login

   ```
   [root@lnxrh60a ~]# ftp 0
   Connected to 0 (0.0.0.0).
   220 (vsFTPd 2.2.2)
   Name (0:root): test1
   331 Please specify the password.
   Password:
   500 OOPS: cannot change directory:/home/test1
   Login failed.
   ftp>
   ```

   An SELinux error is logged to the `/var/log/messages` file, as shown in Example 4-11. Note that he SELinux policy has prevented the FTP daemon from reading users' home directories.

   Example 4-11 Sample of error in /var/log/messages

   ```
   Aug  7 17:51:55 lnxrh60a setroubleshoot: SELinux is preventing /usr/sbin/vsftpd
   from search access on the directory /home. For complete SELinux messages. run
   sealert -l ddb7b307-09ae-4bc7-91de-0a53881e3464
   ```

4. For SELinux to allow VSFTPD access to read users' home directories, enable the `ftp_home_dir` boolean by running the **`setsebool`** command as the root user, as shown in Example 4-12.

   Example 4-12 Sample of enabling ftp_home_dir

   ```
   setsebool -P ftp_home_dir=1
   ```

   A SELinux message is logged to the `/var/log/messages` file, as shown in Example 4-13.

   Example 4-13 Sample of message in /var/log/messages

   ```
   Aug  7 17:54:08 lnxrh60a setsebool: The ftp_home_dir policy boolean was changed
   to 1 by root
   ```

4.1.3 Disabling SELinux

In certain circumstances, disabling SELinux on a server is necessary. Usually, this is the result of a change to a system: either a new application system has been installed, which is not functioning properly because of SELinux, or a change has been made to the SELinux configuration which affects the running system. In these cases, turning off SELinux enables administrators to correct the issue.

Important: This is *not* a suggestion to disable SELinux in general. SELinux provides protection from a wide variety of system attacks, so any plan to disable it must be done in conjunction with an exhaustive review of the applicable security and regulatory policies.

To disable SELinux after an installation, append the entry **`selinux=0`** to the line containing the parameters in the boot loader file (`/etc/zipl.conf`), as shown in Example 4-14.

Example 4-14 Sample /etc/zipl.conf file with disabled SELinux in Red Hat Enterprise Linux 6.2

```
[defaultboot]
timeout=5
default=linux-2.6.32-220.el6.s390x
target=/boot/
[linux-2.6.32-220.el6.s390x]
   image=/boot/vmlinuz-2.6.32-220.el6.s390x
   ramdisk=/boot/initramfs-2.6.32-220.el6.s390x.img
   parameters="root=/dev/disk/by-path/ccw-0.0.0202-part1 rd_NO_LUKS
LANG=en_US.UTF-8 rd_NO_MD  KEYTABLE=us cio_ignore=all,!0.0.0009
SYSFONT=latarcyrheb-sun16 crashkernel=auto rd_NO_DM rd_NO_LVM rd_DASD=0.0.0201
rd_DASD=0.0.0202 selinux=0"
```

Note: Parameter line changes take effect only after issuing the **`zipl`** command to write out the modified boot information.

On a test or development system, particularly one that regularly tests changes to SELinux policies, a worthwhile step might be to add a permanent boot option to the zipl menu to selectively boot without SELinux (such a boot option must *never* be added to a production server, however, to make sure that SELinux always runs as intended). Even on a test or development server, however, the normal operating mode should be with SELinux enabled.

4.1.4 Policies

Policies are sets of rules that allow the administrator to flexibly configure a system according to requirements. SELinux is a flexible access control system whose access decisions are guided by an administrator-definable policy. Policies depend on the layout and configuration of files on the system. Distributions that support SELinux include a predefined policy. Although editing the policy is possible, you generally edit a policy only to the extent of excluding whole chunks of policy relating to software that is not installed on the system. More detailed policy editing generally belongs to the realm of SELinux experts. In this section, we demonstrate how you can define your own policy.

SELinux policy configuration files end with a `.te` file extension and can be found in the policy directory and its subdirectories. The `policy.conf` file is found in the policy directory, and contains things such as the description of types, domains, rules for what each domain can do, roles, users, what roles users can access, and other information.

SELinux creates a `policy.<version>` file, where `<version>` is the version number. It is installed in the `/etc/security/selinux` directory by running the **`make install`** command in the policy directory. Policies will be loaded on the system by default in the next reboot. If, however, you want to make a dynamic change to your policy, run the **`make load`** command in the policy directory so that you can load the new policy into a running kernel.

4.1.5 RPMs required for SELinux

RPM originally was the acronym for Red Hat Package Manager. Over time, RPM has come to be known as the *RPM Package Manager*, a recursive acronym. To enable SELinux on your system, certain RPMs are required, which are:

- `policycoreutils`
- `selinux-policy`
- `selinux-policy-targeted`
- `libselinux`
- `libselinux-utils`
- `libselinux-python`-utils

To check whether the required packages are installed, you can run the command listed in Example 4-15 on page 111.

Example 4-15 Querying SELinux packages using rpm in Red Hat Enterprise Linux 6.2

```
[root@lnxrh60a ~]# rpm -q policycoreutils selinux-policy selinux-policy-targeted
libselinux libselinux-utils libselinux-python
policycoreutils-2.0.83-19.18.el6.s390x
selinux-policy-3.7.19-126.el6.noarch
selinux-policy-targeted-3.7.19-126.el6.noarch
libselinux-2.0.94-5.2.el6.s390x
libselinux-utils-2.0.94-5.2.el6.s390x
libselinux-python-2.0.94-5.2.el6.s390x
```

4.2 AppArmor

AppArmor is a Linux application security framework included with Novell SUSE Linux Enterprise. AppArmor is an open source project. The approach that AppArmor takes is different from SELinux.

AppArmor was created to provide easy-to-use Linux security software that protects Linux servers and applications from malicious threats, reducing the business risk while enforcing system and data integrity. Novell AppArmor features are:

- Yet another Setup Tool (YaST)-integrated administration tool for configuration, maintenance and automated development of a per-program security policy
- Predefined security policies for standard Linux programs and services
- Robust reporting and alerting capabilities to facilitate regulatory compliance
- Common Information Model (CIM)-enabled clients that integrate with industry-standard management consoles
- ZENworks Linux Management integration for profile distribution and report aggregation
- Path-name-based system that does not require labeling or relabeling of file systems

- Less reason to modify other profiles (when developing profiles incrementally), because all profiles simply refer to the path names they use

4.2.1 Important files and directories for AppArmor

Important directories for AppArmor are:

- `/etc/init.d/boot.apparmor`
- `/etc/apparmor.d/`
- `/lib/apparmor/`
- /etc/apparmor/ (for apparmor tools)

Example 4-16 lists the files for AppArmor.

Example 4-16 File list for AppArmor

```
lnxslesa:~ # ls -ltr /etc/apparmor.d
total 80
-rw-r--r-- 1 root root  255 Sep 26  2011 usr.lib.PolicyKit.polkit-grant-helper-pam
-rw-r--r-- 1 root root  231 Sep 26  2011
usr.lib.PolicyKit.polkit-explicit-grant-helper
-rw-r--r-- 1 root root  476 Sep 26  2011 usr.lib.PolicyKit.polkit-revoke-helper
-rw-r--r-- 1 root root  552 Sep 26  2011 usr.lib.PolicyKit.polkit-grant-helper
-rw-r--r-- 1 root root  177 Sep 26  2011
usr.lib.PolicyKit.polkit-resolve-exe-helper
-rw-r--r-- 1 root root  496 Sep 26  2011 usr.lib.PolicyKit.polkit-read-auth-helper
-rw-r--r-- 1 root root  424 Sep 26  2011 usr.lib.PolicyKit.polkitd
-rw-r--r-- 1 root root  745 Jan 23  2012 usr.sbin.traceroute
-rw-r--r-- 1 root root 1857 Jan 23  2012 usr.sbin.ntpd
-rw-r--r-- 1 root root 1246 Jan 23  2012 usr.sbin.nscd
-rw-r--r-- 1 root root  853 Jan 23  2012 usr.sbin.mdnsd
-rw-r--r-- 1 root root  842 Jan 23  2012 usr.sbin.identd
-rw-r--r-- 1 root root  696 Jan 23  2012 usr.sbin.avahi-daemon
-rw-r--r-- 1 root root 1130 Jan 23  2012 sbin.syslog-ng
-rw-r--r-- 1 root root 1087 Jan 23  2012 sbin.syslogd
-rw-r--r-- 1 root root  788 Jan 23  2012 sbin.klogd
-rw-r--r-- 1 root root  732 Jan 23  2012 bin.ping
drwxr-xr-x 2 root root 4096 Aug  7 10:13 tunables
drwxr-xr-x 2 root root 4096 Aug  7 10:13 program-chunks
drwxr-xr-x 2 root root 4096 Aug  7 10:13 abstractions

lnxslesa:~ # ls -ltr /etc/apparmor
total 40
-rw-r--r-- 1 root root   179 Aug 23  2011 reports.crontab
-rw-r--r-- 1 root root   955 Aug 23  2011 reports.conf
-rw-r--r-- 1 root root  1977 Feb  6  2012 subdomain.conf
-rw-r--r-- 1 root root 10342 Feb  6  2012 severity.db
-rw-r--r-- 1 root root   529 Feb  6  2012 notify.conf
-rw-r--r-- 1 root root  4291 Feb  6  2012 logprof.conf
drwxr-xr-x 3 root root  4096 Aug  7 10:13 profiles

lnxslesa:~ # ls -ltr /lib/apparmor
total 16
-rw-r--r-- 1 root root 13310 Feb  6  2012 rc.apparmor.functions
lnxslesa:~ #
```

The two ways in which AppArmor can be enabled or disabled by using YaST are:

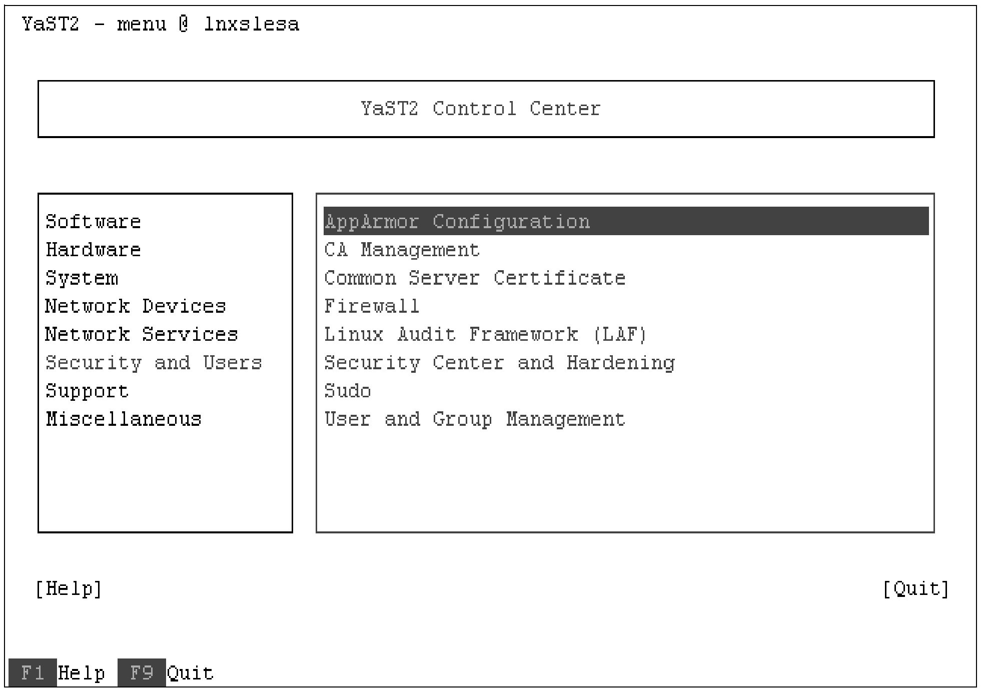
- Novell AppArmor Control Panel
- System Services

4.2.2 Enable or disable AppArmor by using YaST

Enable or disable AppArmor by using the Novell AppArmor Control Panel

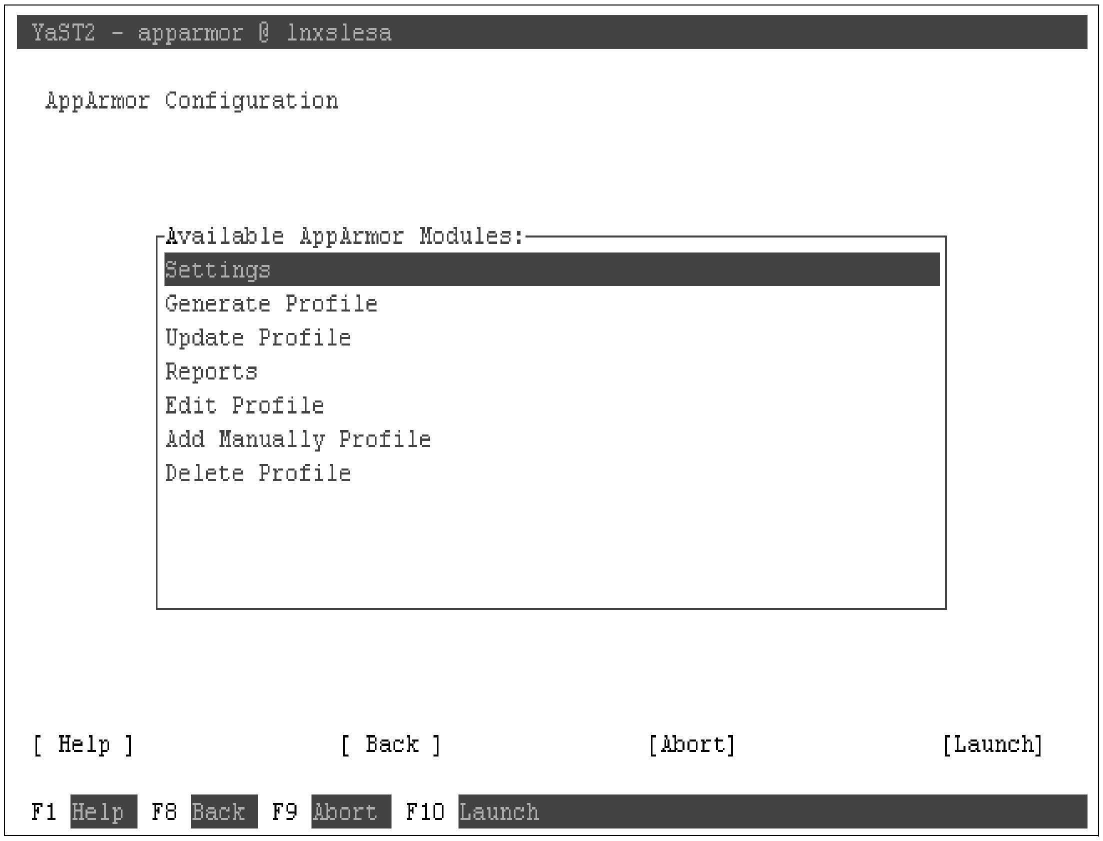
Perform the following steps to enable or disable AppArmor using the Control Panel:

1. Start YaST.
2. Select **Security and Users** → **AppArmor Configuration**, as shown in Figure 4-3 on page 113.
3. Select **Settings**, as shown in Figure 4-4 on page 114.
4. Select **Enable AppArmor**, shown in Figure 4-5 on page 114.
5. Exit the AppArmor Control Panel by selecting **Done**.



```
YaST2 - menu @ lnxslesa

                     YaST2 Control Center

 Software              AppArmor Configuration
 Hardware              CA Management
 System                Common Server Certificate
 Network Devices       Firewall
 Network Services      Linux Audit Framework (LAF)
 Security and Users    Security Center and Hardening
 Support               Sudo
 Miscellaneous         User and Group Management

[Help]                                              [Quit]

F1 Help  F9 Quit
```

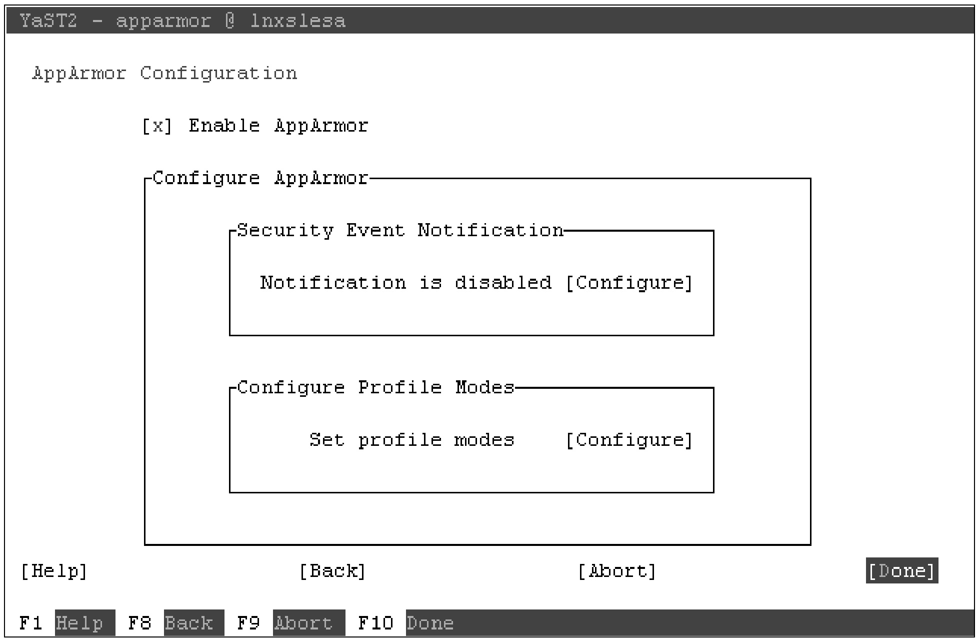
Figure 4-3 Access AppArmor Panel using YaST



```
YaST2 - apparmor @ lnxslesa

 AppArmor Configuration

              Available AppArmor Modules:
              Settings
              Generate Profile
              Update Profile
              Reports
              Edit Profile
              Add Manually Profile
              Delete Profile

[ Help ]              [ Back ]              [Abort]              [Launch]

F1 Help  F8 Back  F9 Abort  F10 Launch
```

Figure 4-4   Access AppArmor settings using YaST

Figure 4-5 shows the AppArmor Control Panel set to enable AppArmor.



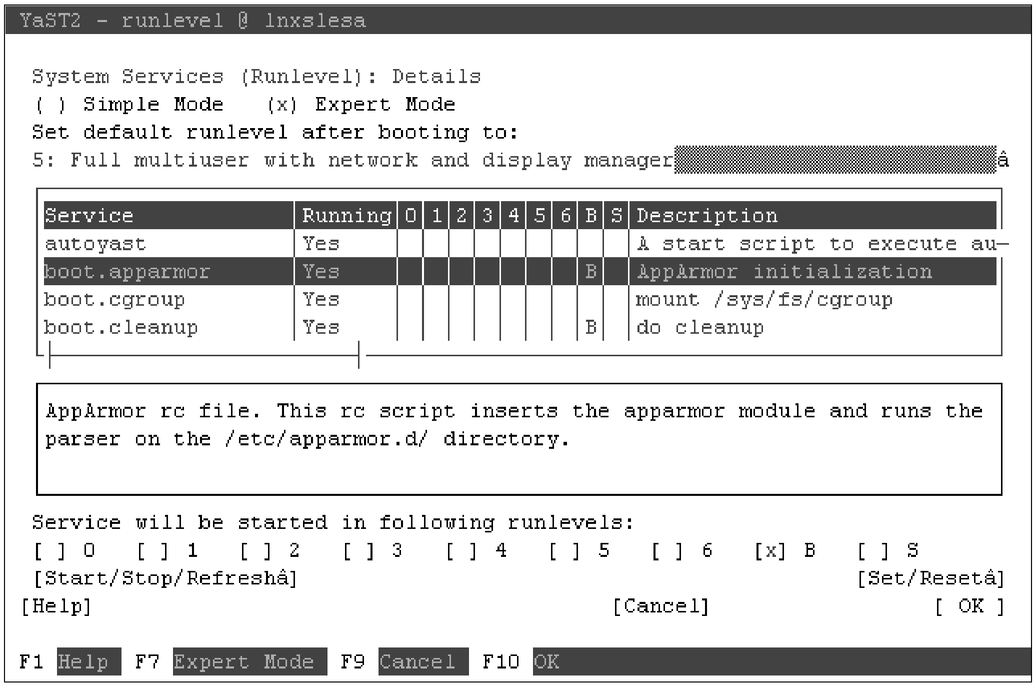
```
YaST2 - apparmor @ lnxslesa

 AppArmor Configuration

          [x] Enable AppArmor

          Configure AppArmor
                 Security Event Notification

                   Notification is disabled [Configure]

                 Configure Profile Modes

                       Set profile modes    [Configure]

[Help]                  [Back]                  [Abort]                  [Done]

F1 Help  F8 Back  F9 Abort  F10 Done
```

Figure 4-5   Enable AppArmor using YaST

Enable or disable by using System Services

Perform the following steps to enable or disable AppArmor by using System Services (see Figure 4-6 on page 115):

1. Start YaST.
2. Select **System** → **System Services (Runlevel)**.
3. Select **Expert Mode**.
4. Select **boot.apparmor** and select the run levels in which you want to run AppArmor.
5. Select **Set/Reset** to enable or disable the service.
6. Select **Ok** to exit the System Services.



```
YaST2 - runlevel @ lnxslesa

 System Services (Runlevel): Details
 ( ) Simple Mode    (x) Expert Mode
 Set default runlevel after booting to:
 5: Full multiuser with network and display manager░░░░░░░░░░░░░░░░░░░░░░░░░â

  Service                Running|0|1|2|3|4|5|6|B|S|Description
  autoyast               Yes    | | | | | | | | | |A start script to execute au-
  boot.apparmor          Yes    | | | | | | | |B| |AppArmor initialization
  boot.cgroup            Yes    | | | | | | | | | |mount /sys/fs/cgroup
  boot.cleanup           Yes    | | | | | | | |B| |do cleanup

  AppArmor rc file. This rc script inserts the apparmor module and runs the
  parser on the /etc/apparmor.d/ directory.

 Service will be started in following runlevels:
 [ ] 0   [ ] 1   [ ] 2   [ ] 3   [ ] 4   [ ] 5   [ ] 6   [x] B   [ ] S
 [Start/Stop/Refreshâ]                                         [Set/Resetâ]
[Help]                                       [Cancel]               [ OK ]

F1 Help  F7 Expert Mode  F9 Cancel  F10 OK
```

Figure 4-6 Enable AppArmor using System Service

Adding a profile using the wizard

The YaST utility for AppArmor includes a wizard that helps you to easily add a profile to the server AppArmor configuration. To add a profile using the wizard:

1. Start YaST.
2. Select **Security and Users** → **AppArmor Configuration**.
3. Select **Generate Profile** under the AppArmor Configuration panel.
4. Enter the name of the application, and select **Create**. This runs aa-autodep, a Novell AppArmor tool.
5. Select **Scan system log for AppArmor events**. This generates a chain of questions that must be answered to guide the wizard in generating the security profile.

   The Add Profile Wizard begins by suggesting directory path entries that have been accessed by the application you are profiling.

Manually adding a profile

Instead of using the wizard, a profile can be added to AppArmor manually, as follows.

1. Start YaST.
2. Select **Security and Users** → **AppArmor Configuration**.
3. In the Novell AppArmor panel, click **Manually Add Profile**.
4. Browse your system to find the application for which to create a profile. When you find the application, select it and click **OK**. A basic, empty profile appears in the Novell AppArmor Profile Dialog window.
5. From the AppArmor Profile Dialog window, you are able to edit Novell AppArmor profiles manually by adding, editing, or deleting entries. Select **Add Entry**.
6. When you are finished, select **Done**, and in the pop-up that appears, click **Yes** to confirm your changes to the profile and reload the AppArmor profile set.

4.2.3 RPMs required for AppArmor

Example 4-17 lists the RPMs required for AppArmor.

Example 4-17 List of RPMs for AppArmor on SLES 11 SP2

```
apparmor-docs
apparmor-parser
apparmor-profiles
apparmor-utils
audit
libapparmor1
perl-libapparmor
yast2-apparmor
```

You do not have to install AppArmor because it is included with all of the software that you need to implement for Linux application security. It can be configured by using a text-based console, or through the YaST Control Center. From YaST, you can create new AppArmor profiles, and manually edit or automatically update existing profiles. Administrators can enable or disable AppArmor, configure reporting and alerting, and run reports on AppArmor events.

4.3 Pluggable Authentication Modules

The Pluggable Authentication Module (PAM) framework provides system administrators with the ability to incorporate multiple authentication mechanisms into a system through the use of pluggable modules. PAM enables system administrators to select how applications authenticate users. Applications that are enabled to make use of PAM can be plugged into new technologies without modifying the existing applications. This flexibility allows administrators to:

- Select any authentication service on the system for an application.
- Use multiple authentication mechanisms for a given service.
- Add new authentication service modules without modifying existing applications.
- Use a previously entered password for authentication with multiple modules.

The PAM framework consists of a library, pluggable modules, and a configuration file. The PAM library implements the PAM application programming interface (API) and serves to manage PAM transactions and invoke the PAM service programming interface (SPI) defined in the pluggable modules.

Pluggable modules are dynamically loaded by the library based on the invoking service and its entry in the configuration file. Successful operation is determined by the pluggable module and also by the behavior defined for the service.

4.3.1 Important files and libraries for PAM

When a user logs in to Linux on System z, and PAM is enabled, the /bin/login program is run. When /bin/login is run, PAM reads the `/etc/pam.d/login` configuration file. In this configuration file are four main types of libraries (known as modules) that are used by PAM:

- The *auth* module checks the login device being used, authenticates the user, and checks for the presence of the `/etc/nologin` file. If the `/etc/nologin` file exists, no users, except root, are allowed to log in to the system. The following libraries for this module are:
  - `For SLES11 SP2:`

Table 4-1 Describing SLES11 SP2 modules

Module Name	Description
pam_nologin.so	Checks if `/etc/nologin` exists. If it does, no user other than root may log in.
pam_securetty.so	Disable or enable root logins from occurring on insecure terminals. It checks the `/etc/securetty` file.
pam_env.so	Loads the environmental variables specified in `/etc/security/pam_env.conf`.
pam_unix2.so	Checks the user's login and password against `/etc/passwd` and `/etc/shadow`.

  - For RHEL 6.2:

Table 4-2 Describing RHEL 6.2 modules

Module Name	Description
pam_env.so	Loads the environmental variables specified in `/etc/security/pam_env.conf`.
pam_unix.so	Checks the user's login and password against `/etc/passwd` and `/etc/shadow`.
pam_succeed_if.so	Used to validate against account characteristics.
pam_deny.so	Used to deny access (Add restrictions).

- An *account* module makes sure that a user has proper access. For example, it might determine whether the user is accessing the system from a remote host or whether the user is subjected to time-of-day restrictions. It also checks for password or account expiration. The required library for this is:
  - For SLES11 SP2: pam_unix2.so
  - For RHEL 6.2: pam_unix.so
- The *password* module sets the password and helps the transition from one authentication module to another. Required libraries for this module are:
  - For SLES11 SP2: `pam_pwcheck.so` (enforces password strength reading for settings in the `/etc/login.def` file) and `pam_unix2.so` (performs authentication token updates)

  - For RHEL 6.2: `pam_cracklib.so` (enforces password strength) and `pam_unix.so` (performs authentication token updates)

- The *session* module manages the user session (such as setting up and terminating login sessions). The required library is:
  - For: SLES11 SP2: `pam_limits.so` and `pam_unix2.so`

  And may be used by this module too:
  - For RHEL 6.2: `pam_selinux.so`, `pam_loginuid.so` and `pam_console.so` (optional library)

For more information about each of these libraries, see:

http://www.linux-pam.org/Linux-PAM-html/

For Novell SUSE Linux on System z, additional required modules can be found at:

http://www.suse.com/documentation/sles11/singlehtml/book_security/book_security.html#cha.pam

PAM applications (pam_login, pam_su, and pam_passwd; and pam_unix2, pam_devperm, pam_pwcheck, pam_homecheck, and pam_chroot to enable Novell SUSE Linux on System z) invoke the PAM API in the PAM library. The library determines the appropriate module to load based on the application entry in the configuration file and calls the PAM Service Provider Interface (SPI) in the module. Communication occurs between the PAM module and the library through the use of a conversation function implemented in the PAM module. Success or failure of the module and the behavior defined in the configuration file then determines whether another module must be loaded. If so, the process continues; otherwise, the data is passed back to the application.

Stacking

Through the concept of stacking a service can be configured to authenticate through multiple authentication methods. Modules can also be configured to use a previously submitted password rather than prompting for additional input.

Stacking is implemented in the configuration file by creating multiple entries with the same module_type field. The modules are invoked in the order in which they are listed in the file, with the final result determined by the control_flag field specified for each entry. Valid values for the control_flag field and the corresponding behavior in the stack are listed in Table 4-3.

Table 4-3 Sample Valid values for control_flag

Control flag	Description
required	For a successful result, all required modules in a stack must pass. If one or more required modules fail, all the required modules in the stack will be attempted, but the error from the first failed required module is returned.
sufficient	If a module flagged as sufficient succeeds and no previous required or sufficient modules have failed, all remaining modules in the stack are ignored and success is returned.
optional	If none of the modules in the stack are required and no sufficient modules have succeeded, then at least one optional module for the service must succeed. If another module in the stack is successful, a failure in an optional module is ignored.

4.3.2 Enabling PAM

The `/etc/pam.d` directory consists of files of service entries for each PAM module type and is used to route services through a defined module path. An overview of the organization of PAM on Linux is shown in Figure 4-7.

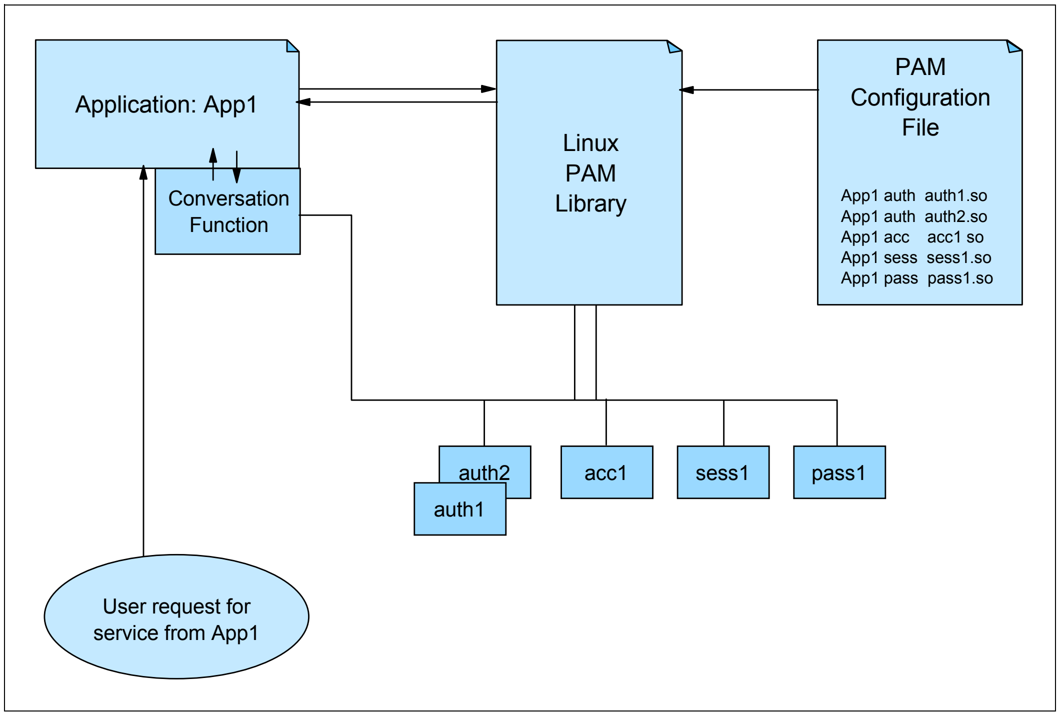


Figure 4-7 Overview of the PAM environment on Linux

The application has no knowledge of the authentication methods in use. It simply calls the PAM functions. PAM is responsible for loading the modules that are used for the application, and it refers to the PAM configuration file for the application to know which modules to use. Different modules can be used for each of the four management groups, shown in the lower-right of the diagram. If the modules have to output a message or require input, they can use the conversation() function that is built into the application (as part of its PAM support code).

See Example 4-18 and Example 4-19 on page 120 for a list of configuration files.

Example 4-18 The PAM configuration files in the /etc/pam.d directory in RedHat 6.2

```
[root@lnxrh60a pam.d]# ls
atd                 other              ssh-keycat
authconfig          passwd             su
authconfig-gtk      password-auth      subscription-manager
authconfig-tui      password-auth-ac   subscription-manager-gui
chfn                polkit-1           sudo
chsh                poweroff           sudo-i
config-util         reboot             su-l
crond               remote             system-auth
cups                rhn_register       system-auth-ac
cvs                 run_init           system-config-authentication
fingerprint-auth    runuser            system-config-date
fingerprint-auth-ac runuser-l          system-config-keyboard
```

```
gdm                    selinux-polgengui  system-config-network
gdm-autologin          setup              system-config-network-cmd
gdm-password           smartcard-auth     system-config-selinux
halt                   smartcard-auth-ac  system-config-users
ksu                    smtp               vsftpd
login                  smtp.postfix
newrole                sshd
[root@lnxrh60a pam.d]#
```

Example 4-19 The PAM configuration files in the /etc/pam.d directory in SLES11 SP2

```
lnxslesa:/etc/pam.d # ls
atd                                 common-password-pc                 shadow
chage                               common-session                     smtp
chfn                                common-session.pam-config-backup   sshd
chsh                                common-session-pc                  su
common-account                      crond                              sudo
common-account.pam-config-backup    gnomesu-pam                        su-l
common-account-pc                   login                              useradd
common-auth                         other                              vlock
common-auth.pam-config-backup       passwd                             xdm
common-auth-pc                      polkit                             xdm-np
common-password                     ppp                                xlock
common-password.pam-config-backup   rpasswd
lnxslesa:/etc/pam.d #
```

The configuration files in `/etc/pam.d` are named according to the name of the application that is using PAM.

PAM configuration file example

The login program manages a user login at the system console. If problems exist with the login process, check the `/etc/pam.d/login` file, because this file contains the entries that are used by the login program. Example 4-20 and Example 4-21 on page 121 show how that file looks in RHEL and SLES.

Note: For additional information, refer to PAM documentation at:

http://www.linux-pam.org/Linux-PAM-html/

Example 4-20 Sample PAM /etc/pam.d/login login file in Redhat 6.2

```
#%PAM-1.0
auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so
auth       include      system-auth
account    required     pam_nologin.so
account    include      system-auth
password   include      system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should only be followed by sessions to be executed in the
user context
session    required     pam_selinux.so open
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      system-auth
-session   optional     pam_ck_connector.so
```

Example 4-21 Sample PAM /etc/pam.d/login login file in SLES11 SP2

```
#%PAM-1.0
auth requisite pam_nologin.so
auth user_unknown=ignore success=ok ignore=ignore auth_err=die default=bad]
pam_securetty.so
auth include common-auth
account  include common-account
password include common-password
session  required pam_loginuid.so
session include common-session
session  optional pam_lastlog.sonowtmp
session  optional pam_mail.so standard
session optional pam_ck_connector.so
```

Example 4-22 and Example 4-23 are a sample `/etc/pam.d/passwd` file, which is used by the **`passwd`** program (to update a user password).

Example 4-22 Sample /etc/pam.d/passwd file in Redhat 6.2

```
#%PAM-1.0
auth       include system-auth
account    include system-auth
password   substack system-auth
password   optional pam_gnome_keyring.so
```

Example 4-23 Sample /etc/pam.d/passwd file in SLES11 SP2

```
#%PAM-1.0
auth includecommon-auth
account  includecommon-account
password includecommon-password
session includecommon-session
```

PAM INCLUDE syntax

On most systems, the same authentication methods are used for the majority of applications. This approach ensures operational consistency. Rather than putting the onus on the system administrator to update all the configuration files in the same way, PAM provides an INCLUDE statement that allows common definitions to be placed in a single file and allows the inclusion of any other configuration files. The PAM configuration file examples Example 4-22 and Example 4-23 show this syntax, by having INCLUDE statements for the `system-auth` file and common-`auth` file.

The `system-auth` configuration file provides common configuration statements for the basic authentication methods used on the system. See Example 4-24.

Example 4-24 Sample /etc/pam.d/system-auth file

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3 type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass 
use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet 
use_uid
session     required      pam_unix.so
```

Example changes to PAM configuration files

Note: Before making any changes to PAM files, make sure that you have a backup of files. Be careful when performing the configuration changes. A wrong configuration can lock down all login access including root access.

Another essential step when making PAM changes is to keep open a login session to a user ID with authority to edit PAM configuration. Test your changes by using another session; if your changes were unsuccessful, you can resolve the issue by returning to the session that you left open.

Warnings of unconfigured PAM programs

Every program that supports PAM usually has a specific PAM configuration file in the `/etc/pam.d` location (the file is named the same as the name of the program being executed). If no specific configuration file exists for a program, PAM uses the `/etc/pam.d/other` file.

The addition of the following lines to the `/etc/pam.d/other` file in the previous example would provide a suitable warning to the administrator that attempts are being made to use a program for which no specific PAM configuration exists:

```
other   auth     required      pam_warn.so
other   password required      pam_warn.so
```

See Example 4-25 after making changes to the `/etc/pam.d/other` file.

Example 4-25 Sample /etc/pam.d/other file after making changes to get warning

```
# default configuration: /etc/pam.d/other
#
auth     required      pam_warn.so
auth     required      pam_deny.so
account  required      pam_deny.so
password required      pam_warn.so
password required      pam_deny.so
session  required      pam_deny.so
```

Disabling logins for users with null passwords

Linux systems have a number of system accounts that are used to assign privileges to certain system services. Allowing these system accounts to have login privileges is a security risk, because they usually have blank (null) passwords. Therefore, the following lines in the `/etc/pam.d/login` file can be added to disable user login if using null passwords:

```
auth  required  pam_unix.so nullok
auth  required  pam_unix.so
```

Limiting SSH access by user ID

To allow selected users to log in with SSH, the following changes can be made:

1. Create the list of all allowed users to log in using **`sshd`**; see Example 4-26.

   Example 4-26 Sample file for list of allowed users to do ssh: /etc/ssh/allow-user

   ```
   root
   user1
   user2
   ```

2. Make sure the file is owned by the root user and permission is 600, as shown in Example 4-27.

   Example 4-27 Sample for checking file details

   ```
   [root@lnxrh60a ~]# ls -ltr /etc/ssh/allow-user
   -rw------- 1 root root 17 Aug  9 08:50 /etc/ssh/allow-user
   ```

3. Make changes to the `/etc/pam.d/sshd` file; add the following line to the `auth` session:

   ```
   auth required pam_listfile.so item=user sense=allow file=/etc/ssh/allow-user onerr=fail
   ```

   Example 4-28 shows the `/etc/pam.d/sshd` file after making this change.

   Example 4-28 Sample example for sshd file after making changes

   ```
   [root@lnxrh60a ~]# cat /etc/pam.d/sshd
   #%PAM-1.0
   auth required pam_listfile.so item=user sense=allow file=/etc/ssh/allow-user
   onerr=fail
   auth requiredpam_sepermit.so
   auth       include      password-auth
   account    required     pam_nologin.so
   account    include      password-auth
   password   include      password-auth
   # pam_selinux.so close should be the first session rule
   session    required     pam_selinux.so close
   session    required     pam_loginuid.so
   # pam_selinux.so open should only be followed by sessions to be executed in the
   user context
   session    required     pam_selinux.so open env_params
   session    optional     pam_keyinit.so force revoke
   session    include      password-auth
   ```

4.3.3 PAM and LDAP

In 3.7, "Linux authentication using the z/VM LDAP server" on page 71, the steps to configure a Linux system for authentication with the z/VM LDAP server are shown. The process is the same regardless of what type of LDAP server is used, however.

Some installation sites have discovered that when they enable LDAP authentication they have problems trying to log on to their systems when the LDAP server they use is not available. The problem is often traced to an incorrect PAM configuration where the `required` keyword is specified for pam_ldap in the PAM configuration, but sometimes the problem is more subtle and requires deeper extensive troubleshooting.

Note: This issue can be argued as a point in favor of using the z/VM LDAP server as the LDAP database for your Linux systems. By using a highly available LDAP server like this, you are decreasing the chances that downtime or unavailability will affect the operation of your Linux systems.

The following checks can ensure that you are able to at least log in to resolve a configuration problem if LDAP is not available:

- Have emergency administrator identification.

  An *emergency* administrator ID in the local user database of every system ensures that a valid ID is present on the system even when LDAP is not available. A good practice is to have a system administrator account in the `/etc/passwd` file.

  Note: For many installations, the `root account` (UID 0) can serve this purpose. However, you might determine that a different account be set up so that you can avoid the need for the root password to be known (or usable) by administrators.

- Never put root in LDAP.

  The root account (UID 0) has critical significance to Linux systems, and must always be stored in the local user database. Putting the root account in LDAP is a bad idea for at least two reasons:

  - Having an account with UID 0 in LDAP creates a possibility of privilege escalation across every server using a given LDAP database for authentication.
  - If an unsuspecting administrator mistakenly deletes the root account from the local database because it is present in LDAP, major problems with application operations are likely.

- Use automatic logon of a Linux *operator* ID at the console.

  With access to the z/VM console controlled as described in 2.4, "Securing console access to z/VM virtual machines" on page 9, almost no additional security is afforded by having administrators answer a Linux login prompt at the console. In the past, this approach was difficult to set up, but the version of the mingetty program in use on SLES 11 and RHEL 6 distributions now includes a switch to automatically log on a user at the console.

  Note: Again, this account normally is the root account, but a different account can be used to provide separation of duties or other control.

- Test your "LDAP unavailable" login process regularly.

  An essential step is to regularly test your ability to access your Linux systems (through the console at the very minimum) in various *impaired* configurations.

Note: Because of the way the Linux TCP/IP stack works, a difference exists between a remote server being inaccessible because the local network is down, and a remote server being inaccessible because *it* is down (or an intervening network problem has broken the path to it). In these situations, pam_ldap behaves differently, so each needs to be tested.

4.4 Sudo (superuser do)

Without a tool such as SELinux or AppArmor, it is a bit complicated to keep control and determine privileges of a user. The super user (root) is a privileged account that has unrestricted access and is often required by more than one user. Usually in most environments, the root password is shared amongst the administrators creating a security breach. Although it makes password management easier, it creates unacceptable risk and compliance violation, since you cannot meet the individual accountability requirements and segregation of duties.

Note: Individual accountability is the ability to directly attribute a specific action to a specific person.

Making administrators to use Sudo is a very good practice in terms of security. It helps restricting a set of commands and also the share of the root password. You should be cautious about reads and writes to the root password and protect it. Sudo is the way to protect it. With Sudo, users can execute commands as root, but without the need of sharing the root password.

Sudo reads the `/etc/sudoers` file which contains a list of rules for authorizing commands for a user ID. It improves security by restricting access commands that administrators can use. Additionally, Sudo provides logging for each privileged command, by writing it to the standard log (syslog) for future audits.

When adding or removing Sudo rules, you need to edit the main configuration file, `/etc/sudoers`. Although you can use vim or some other text editor, we suggest the use of visudo because:

- It locks the `/etc/sudoers` file.
- It saves edits to a temporary file before updating the main Sudo configuration file.
- It checks whether the syntax of the edited file is correct.

If invalid entries get inserted into the `/etc/sudoers` file, these could potentially break the Sudo file syntax enough for other commands to not work.

The use of Sudo has the following advantages:

- Commands are logged to a file, therefore you can figure out who did what and when, which improves accountability.
- The root password can be removed or only known for one person.
- Privileges can be added and revoked at any time.
- Easy transferring of user rights or activities to another user.

Caution in granting Sudo privileges

The person who has root privilege should safeguard updates of the `/etc/sudoers` file and not allow indiscriminate updates by other users. Therefore, this person will probably need to define which users can use Sudo and which commands they can execute to perform their daily support instead of granting excessive privileges to them.

Although providing full access to Sudo is simpler, do not do it. You need to try to limit Sudo access to only those commands specific users need and when possible, delegate authority by groups rather than individual users.

When all privileges are set up and tested, you can remove the password from root user ID or lock it by replacing the encrypted string with an asterisk ("*"). It means the root user ID can no longer be logged into from the network or have its password cracked. As stated before, all access to root via Sudo commands will be logged, which creates an audit trail of everything that is being executed as root.

How to check for unauthorized Sudo executions

Unauthorized executions are logged by default in the standard log, `/var/log/secure`, so when a user who does not have Sudo permission tries to execute a Sudo command, a message as shown in Example 4-29 is recorded.

Example 4-29 Output of /var/log/secure log file

```
Aug  9 14:40:07 lnxrh60a sudo:    user1 : user NOT in sudoers ; TTY=pts/2 ;
PWD=/home/user1 ; USER=root ; COMMAND=/sbin/ifconfig
```

4.5 OpenSSH

OpenSSH(OpenBSD Secure Shell) is a Linux Security application to provide end-to-end encrypted communication sessions over a computer network using the SSH protocol. OpenSSH is developed as part of the OpenBSD project.

4.5.1 Important files and directories for OpenSSH

OpenSSH works in client-server architecture, and configuration files for client and server are stored at different locations.

- SSH daemon configuration files.

  These are host-specific configuration files and are stored in the `/etc/ssh/` directory. In the ITSO test environment, we hosted specific configuration files, as shown in Example 4-30.

Example 4-30 Host-specific SSH configuration files

```
lnxsu11:~ #
lnxsu11:~ # ls -l /etc/ssh/
total 160
-rw------- 1 root root 125811 May  9  2010 moduli
-rw-r--r-- 1 root root   2705 May  9  2010 ssh_config
-rw------- 1 root root    668 Jun 20  2011 ssh_host_dsa_key
-rw-r--r-- 1 root root    602 Jun 20  2011 ssh_host_dsa_key.pub
-rw------- 1 root root    527 Jun 20  2011 ssh_host_key
-rw-r--r-- 1 root root    331 Jun 20  2011 ssh_host_key.pub
-rw------- 1 root root    887 Jun 20  2011 ssh_host_rsa_key
```

```
-rw-r--r-- 1 root root    222 Jun 20  2011 ssh_host_rsa_key.pub
-rw-r----- 1 root root   3902 May  9  2010 sshd_config
lnxsu11:~ #
```

Table 4-4 describes host-specific configuration files.

Table 4-4 Host-specific SSH configuration files

File Name	Description
/etc/ssh/moduli	OpenSSH uses the Diffie-Hellman key exchange algorithm to exchange keys at the beginning of the SSH session, and after successful exchange the same secret key is generated at client and server, and that will be used as shared secret key for encrypting communications. `/etc/ssh/moduli` contains Diffie-Hellman groups which are used for the key exchange.
/etc/ssh/ssh_config	This is the default host-specific SSH client configuration file; it is in key-value pair format. If users have their own specific configuration file (`~/.ssh/config`), it will override the values stored in `/etc/ssh/ssh_config`.
/etc/ssh/ssh_host_dsa_key	The SSH daemon can use Digital Signature Algorithm (DSA); this file stores the DSA private key used by the SSHD daemon.
/etc/ssh/ssh_host_dsa_key.pub	This file stores the DSA public key used by the SSHD daemon.
/etc/ssh/ssh_host_key	The SSH daemon can also use the RSA algorithm and this file stores an RSA private key used by the SSHD daemon for the SSH v1 protocol.
/etc/ssh/ssh_host_dsa_key.pub	This file stores the RSA public key used by the SSHD daemon for the SSH v1 protocol.
/etc/ssh/ssh_host_rsa_key	This file stores the RSA private key used by the SSHD daemon for the SSH v2 protocol
/etc/ssh/ssh_host_rsa_key.pub	This file stores the RSA public key used by the SSHD daemon for the SSH v2 protocol.
/etc/ssh/sshd_config	This file stores host-specific SSH server configuration.
/etc/ssh/shosts.equiv	This file stores trusted client hosts and server accounts.

- SSH client program configuration files

  SSH client program configuration files are user-specific configuration files stored in the `<home directory>/.ssh` directory on a UNIX platform. In the ITSO test environment, we had user-specific configuration files, as given in Example 4-31.

Example 4-31 User-specific SSH client configuration files

```
testuser@lnxsu11:/home/testuser> ls -l /home/testuser/.ssh/
total 4
-rw-r--r-- 1 testuser users 219 2012-08-08 10:58 known_hosts
testuser@lnxsu11:/home/testuser>
```

Table 4-5 describes user-specific configuration files.

Table 4-5 User specific SSH client configuration files

File Name	Description
~/.ssh/known_hosts	On a first logon request to the SSH server through the SSH client, the user is prompted to store host keys locally and if user agrees, the server host keys are added to the `~/.ssh/known_host` file. On subsequent logon requests the host key stored in the `known_hosts` file is compared to the one that is available on the server. If they do not match, the client program is notified before establishing a session between client and server. You should contact the system administrator to make sure the system has not been compromised if the key has changed and you are unsure why it has changed.
~/.ssh/authorized_keys	This file stores SSH client public keys and is used by the SSH server to authenticate a client. Note: This is an optional authentication method available with OpenSSH.
~/.ssh/id_dsa	This file stores the DSA private key of the user.
~/.ssh/id_dsa.pub	This file stores the DSA public key of the user.
~/.ssh/id_rsa	This file stores the RSA private key of the user; it is used by the SSH client program when using the SSH v2 protocol.
~/.ssh/id_rsa.pub	This file stores the RSA public key of the user; it is used by the SSH client program when using the SSH v2 protocol.
~/.ssh/identity	This file stores the RSA private key of the user; it is used by the SSH client program when using the SSH v1 protocol.
~/.ssh/identity.pub	This file stores the RSA public key of the user; it is used by the SSH client program when using the SSH v1 protocol.
~/.ssh/environment	If this file is available, it gets read into the environment during login.
~/.ssh/rc	This file stores initialization routines.

Note: You may see different host- and user-specific SSH configuration files in your environment; these configuration files vary from environment to environment.

4.5.2 OpenSSH commands

The essential OpenSSH commands are:

- ssh

  This is an OpenSSH client program for logging into remote hosts and executing commands through a secure encrypted channel. The OpenSSH client supports SSH protocols v1 and v2, with v2 being the default and v1 as fall back in case v2 is not supported.

- ssh-keygen

  This is used to generate, manage and convert public and private key pairs for RSA and DSA, and you can use this key pair for authentication. Also, **`ssh-keygen`** is used to generate groups for use in Diffie-Hellman group exchanges.

- ssh-agent

  This is a program to securely store private keys used for public key authentication (RSA, DSA).
- ssh-add

  This adds RSA or DSA private keys to the authentication agent. By default, the **ssh-add** command adds private keys to the files `~/.ssh/id_rsa`, `~/.ssh/id_dsa` and `~/.ssh/identity`; however, you can provide the file name as a command line parameter. The authentication agent must be running while executing the **ssh-add** command.
- ssh-copy-id

  This is a script that logs into a target machine and updates the `authorized_keys` file with the user's public key.
- ssh-keyconverter

  OpenSSH supports SSH v1 and v2 protocols so you can use the **ssh-keyconverter** command to convert RSA public and private keys used for public key authentication with protocol v1 to v2.
- ssh-keyscan

  This is a command for collecting the public SSH host keys of a host given in the file or standard input; it updates the `~/.ssh/known_hosts` file. **ssh-keyscan** connects to remote hosts in parallel through non-secured non-blocking I/O sockets.
- sshd

  This is the OpenSSH server daemon program for SSH. It provides secure encrypted communications between two hosts. By default, **sshd** listens on port 22; however, temporarily you can use the -p flag to change the default port, and for a permanent change you can update the `sshd_config` file. **sshd** can be configured using the configuration file `/etc/ssh/sshd_config` or command line options; command line options override configuration file values.
- scp

  This is a secure remote file copy program. It uses **ssh** for secure data transfer and authentication.
- sftp

  This is also a secure remote file transfer program. It uses **ssh** for secure data transfer and authentication.

4.5.3 Enable or disable OpenSSH

There are multiple ways to enable or disable OpenSSH. However, in the ITSO test environment we used SLES 11p1 and the **chkconfig** command for enabling and disabling OpenSSH. **chkconfig** is the commonly used command to enable or disable a system service on a Linux platform.

You can use the **chkconfig** command to check the status of OpenSSH, as given in Example 4-32.

Example 4-32 Status of the SSH daemon

```
lnxsu11:~ # chkconfig | grep sshd
sshd                 on
```

Note: In the ITSO test environment *sshd* daemon is enabled. If it is disabled in your environment, you will see the status as *off*.

You can use the **`chkconfig sshd on`** command to enable OpenSSH as given in Example 4-33 on page 130.

Example 4-33 Enabling the SSH daemon using the chkconfig command

```
lnxsu11:~ # chkconfig sshd on
```

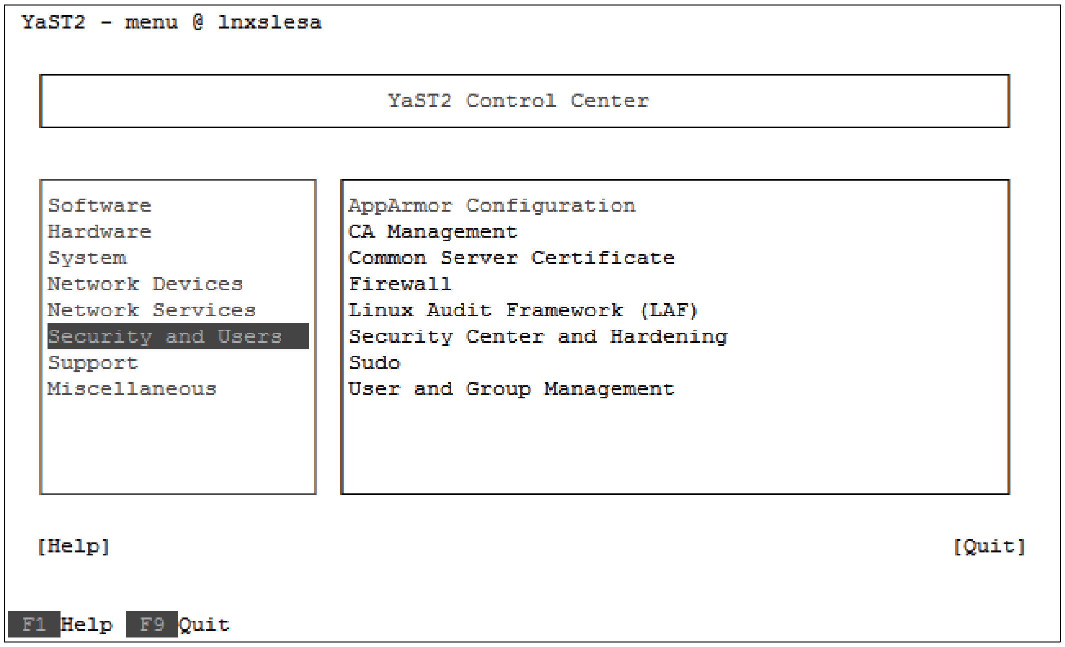
You can use the **`chkconfig sshd off`** command to disable OpenSSH, as shown in Example 4-34.

Example 4-34 Disabling the SSH daemon using the chkconfig command

```
lnxsu11:~ # chkconfig sshd off
```

You may need to restart iptables to remove SSH-specific rules after enabling or disabling OpenSSH. On SLES, **`SuSEfirewall2`** is the command to restart iptables. However, we want to demonstrate using yast to make these steps generalized across all Linux flavors.

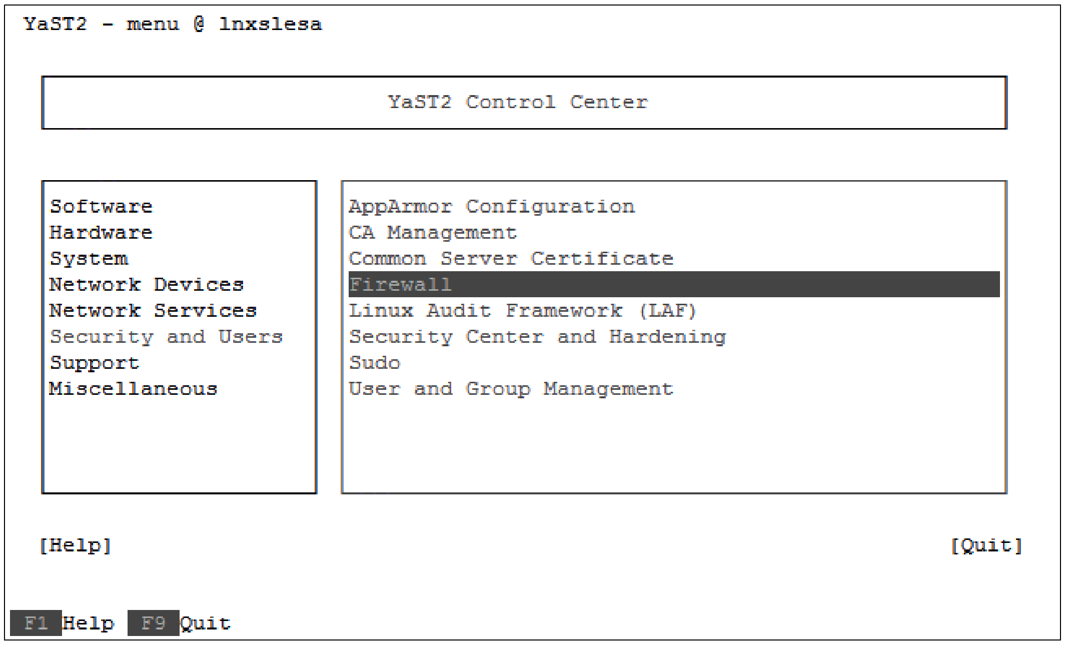
1. Start YaST.
2. Select **Security and Users**, as shown in Figure 4-8.



```
YaST2 - menu @ lnxslesa

                    YaST2 Control Center

Software               AppArmor Configuration
Hardware               CA Management
System                 Common Server Certificate
Network Devices        Firewall
Network Services       Linux Audit Framework (LAF)
Security and Users     Security Center and Hardening
Support                Sudo
Miscellaneous          User and Group Management

[Help]                                                [Quit]

F1 Help  F9 Quit
```

Figure 4-8 Security and Users in YaST

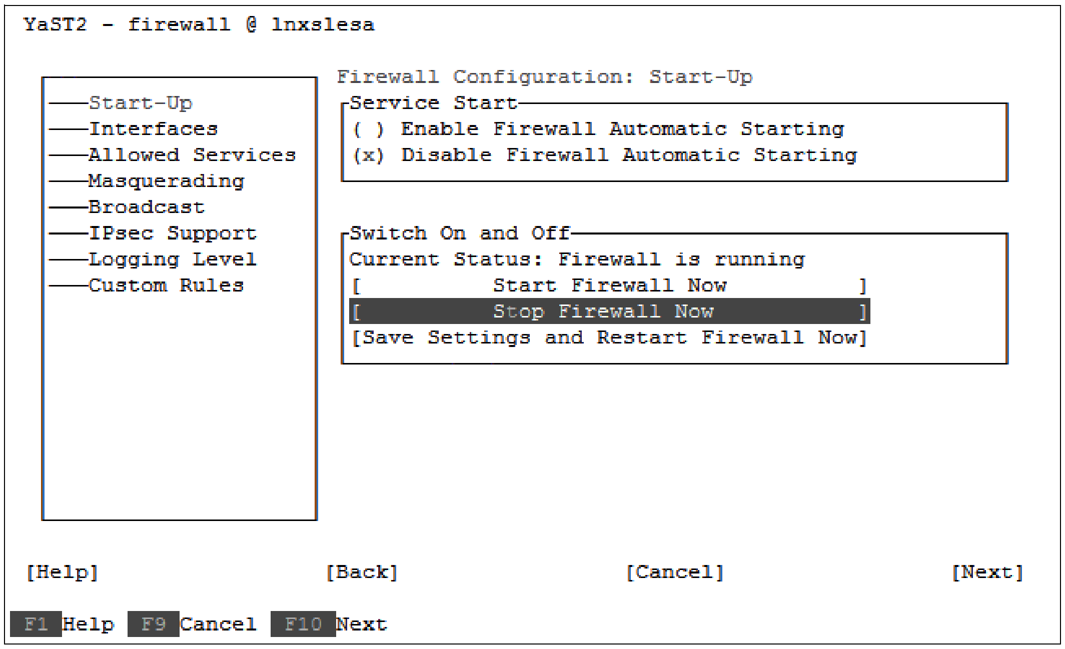
3. Select **Firewall**, as shown in Figure 4-9, and press the Enter key.



```
YaST2 - menu @ lnxslesa

                         YaST2 Control Center

 Software                 AppArmor Configuration
 Hardware                 CA Management
 System                   Common Server Certificate
 Network Devices          Firewall
 Network Services         Linux Audit Framework (LAF)
 Security and Users       Security Center and Hardening
 Support                  Sudo
 Miscellaneous            User and Group Management

[Help]                                                  [Quit]

F1 Help  F9 Quit
```

Figure 4-9 Firewall in the Security and Users menu in YaST.

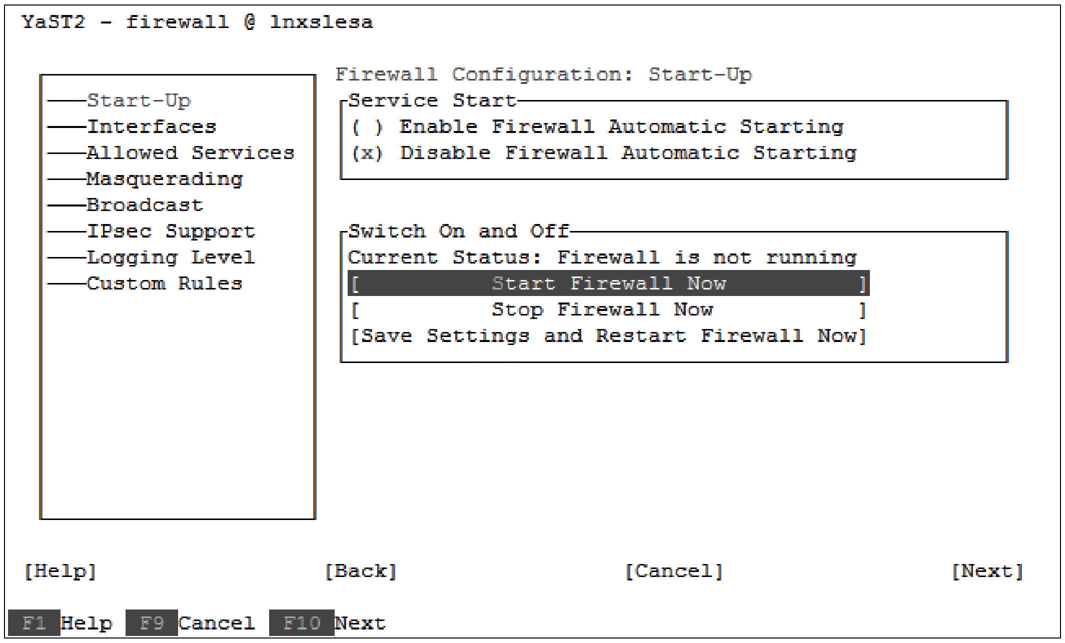
4. Select **Stop Firewall Now**, as shown in Figure 4-10, and press Enter.



```
YaST2 - firewall @ lnxslesa

                        Firewall Configuration: Start-Up
 ──Start-Up             Service Start
 ──Interfaces           ( ) Enable Firewall Automatic Starting
 ──Allowed Services     (x) Disable Firewall Automatic Starting
 ──Masquerading
 ──Broadcast
 ──IPsec Support        Switch On and Off
 ──Logging Level        Current Status: Firewall is running
 ──Custom Rules         [          Start Firewall Now          ]
                        [          Stop Firewall Now           ]
                        [Save Settings and Restart Firewall Now]

[Help]              [Back]              [Cancel]              [Next]

F1 Help  F9 Cancel  F10 Next
```

Figure 4-10 Stop Firewall Now in YaST

5. Once the firewall is stopped, select **Start Firewall Now**, as shown in Figure 4-11 on page 132, and press Enter.



```
YaST2 - firewall @ lnxslesa

                              Firewall Configuration: Start-Up
 ├─Start-Up                   ┌Service Start──────────────────────────────┐
 ├─Interfaces                 │( ) Enable Firewall Automatic Starting     │
 ├─Allowed Services           │(x) Disable Firewall Automatic Starting    │
 ├─Masquerading               └───────────────────────────────────────────┘
 ├─Broadcast
 ├─IPsec Support              ┌Switch On and Off──────────────────────────┐
 ├─Logging Level              │Current Status: Firewall is not running    │
 └─Custom Rules               │[          Start Firewall Now          ]   │
                              │[          Stop Firewall Now           ]   │
                              │[Save Settings and Restart Firewall Now]   │
                              └───────────────────────────────────────────┘

[Help]                 [Back]                  [Cancel]                [Next]

F1 Help  F9 Cancel  F10 Next
```

Figure 4-11   Start Firewall Now in YaST.

6. Exit the Firewall Control Panel with **Done**.

4.5.4 Authentication methods

OpenSSH supports a couple of different authentication methods. However, in this book we demonstrate the most commonly used and most difficult to configure authentication methods.

Host-based authentication

In host-based authentication the user does not have to provide a key, password or passphrase while authenticating to the host. So this authentication method is helpful where a user has first logged in to a trusted system, needs to access untrusted hosts over a network, and does not want to pass on the credentials to an untrusted host. Host-based authentication works with the assumption that the username is the same on trusted and untrusted systems.

To simplify the demonstration of host-based authentication, we assume that hostA is the trusted host where userA has already logged in and wants to log in to hostB (untrusted) without providing a password, key or passphrase. To configure host-based authentication, follow these steps:

1. Build a trust between hostA and hostB, update the `/etc/ssh/shosts.equiv` file of hostB and add hostA's hostname to build trust. You can use the command given in Example 4-35 to add the hostname of hostA in hostB's `/etc/ssh/shosts.equiv` file.

   Example 4-35   Building a trust between hostA and hostB

   ```
   hostB:/etc/ssh # echo "hostA" >> /etc/ssh/shosts.equiv
   hostB:/etc/ssh # cat /etc/ssh/shosts.equiv
   hostA
   ```

2. On hostA, update the `/etc/ssh/ssh_known_hosts` file with hostA, key type (rsa or dsa) and the public key of hostA. You can use the **ssh-keyscan** command to find the public key along with key type, as shown in Example 4-36 on page 133.

Example 4-36 Building the ssh_known_hosts file

```
hostB:/etc/ssh # ssh-keyscan hostA >> /etc/ssh/ssh_known_hosts
# hostA SSH-2.0-OpenSSH_5.1
hostB:/etc/ssh # cat /etc/ssh/ssh_known_hosts
hostA ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAx4BrhZgSABsREF18etJ1ckNZ13nYJp4RSpaIgh0isSNEoYwGru6
tB8hrZQCOgcXV+md4Nw/qq7+1tEm5KNMBhMUBUz4j3N6VrXqAnxePUd5TjhPBqXbKoN325EIKKVKKbj
s+QBElBIYJBBa/N1cBYvYNO8iD4CKwMklqpIcKvdE=
```

3. On hostB, update the `/etc/ssh/sshd_config` file and make sure that the parameters are set as given here:
   – hostBasedAuthentication yes
   – IgnoreUserKnownHosts yes
   – IgnoreRhosts yes
4. On hostB, restart the ssh daemon (sshd) using the **`service sshd restart`** command as shown in Example 4-37.

Example 4-37 Restarting the SSH daemon

```
hostB:/etc/ssh # service sshd restart
Shutting down SSH daemon                                    done
Starting SSH daemon                                         done
```

5. On hostA, update the `/etc/ssh/ssh_config` file and make sure that the parameters are set as given here:
   – EnableSSHKeysign yes
   – hostBasedAuthentication yes
6. On hostA, provide read access to all public keys. You can use the command shown in Example 4-38.

Example 4-38 Adding read access to public key of hostA

```
hostA:/etc/ssh # chmod 644 /etc/ssh/*.pub
hostA:/etc/ssh # ls -l /etc/ssh/
total 164
-rw-r--r-- 1 root root   2375 Jan 10  2012 ldap.conf
-rw------- 1 root root 125811 Jan 10  2012 moduli
-rw-r--r-- 1 root root   2722 Aug  9 14:58 ssh_config
-rw------- 1 root root    668 Aug  7 10:34 ssh_host_dsa_key
-rw-r--r-- 1 root root    603 Aug  7 10:34 ssh_host_dsa_key.pub
-rw------- 1 root root    528 Aug  7 10:34 ssh_host_key
-rw-r--r-- 1 root root    332 Aug  7 10:34 ssh_host_key.pub
-rw------- 1 root root    887 Aug  7 10:34 ssh_host_rsa_key
-rw-r--r-- 1 root root    223 Aug  7 10:34 ssh_host_rsa_key.pub
-rw-r----- 1 root root   3965 Jan 10  2012 sshd_config
hostA:/etc/ssh
#
```

7. On hostA, enable setuid on the keysign binary.

Example 4-39 Enabling setuid on keysign

```
hostA:/etc/ssh # ls -l /usr/lib64/ssh/ssh-keysign
-rwxr-xr-x 1 root root 213992 Jan 10  2012 /usr/lib64/ssh/ssh-keysign
hostA:/etc/ssh # chmod u+s /usr/lib64/ssh/ssh-keysign
hostA:/etc/ssh # ls -l /usr/lib64/ssh/ssh-keysign
-rwsr-xr-x 1 root root 213992 Jan 10  2012 /usr/lib64/ssh/ssh-keysign
```

8. **ssh** to hostA and log in as userA.
9. Now log in to hostB with the verbose flag.

Example 4-40 Host-based authentication

```
userA@hostA:~> ssh -v hostB
OpenSSH_5.1p1, OpenSSL 0.9.8j-fips 07 Jan 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to hostB [9.12.4.225] port 22.
debug1: Connection established.
debug1: identity file /home/userA/.ssh/id_rsa type -1
debug1: identity file /home/userA/.ssh/id_dsa type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.1
debug1: match: OpenSSH_5.1 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.1
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-cbc hmac-md5 none
debug1: kex: client->server aes128-cbc hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host 'hostB' is known and matches the RSA host key.
debug1: Found key in /home/userA/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue:
publickey,keyboard-interactive,hostAased
debug1: Next authentication method: hostAased
debug1: permanently_drop_suid: 1001
debug1: Remote: Accepted for hostA [9.12.5.104] by /etc/ssh/shosts.equiv.
debug1: Authentications that can continue:
publickey,keyboard-interactive,hostAased
debug1: permanently_drop_suid: 1001
debug1: Remote: Accepted for hostA [9.12.5.104] by /etc/ssh/shosts.equiv.
debug1: Authentication succeeded (hostAased).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
```

```
debug1: Sending env LANG = en_US.UTF-8
Last login: Thu Aug  9 15:18:23 2012 from hostA
userA@hostB:~>
```

Note: If you are trying to ssh to the untrusted host for the first time, ssh will prompt you to add a host key.

Public key authentication

By using public key authentication, users are allowed to log in to a remote host without providing a password. They just need to have public and private key pairs to log in to a remote host. Prior to the login attempt, a user's public key needs to be sent to the remote host and stored in the authorized key file; and the user needs to have a valid private key while authenticating to a remote host.

At present OpenSSH supports RSA and DSA asymmetric encryption algorithms.

To simplify the demonstration of public key authentication, we are assuming that hostA is the host where userA has already logged in and userA wants to log in to *hostB* with public key authentication. To configure public key authentication, follow these steps:

1. On hostA, log in as userA and generate public and private keys using the **ssh-keygen** command as shown in Example 4-41.

Example 4-41 Generating public and private keys using ssh-keygen

```
userA@lnxslesa:~/.ssh> ls -l
total 4
-rw-r--r-- 1 userA users 228 Aug  9 15:07 known_hosts
userA@lnxslesa:~/.ssh>
userA@lnxslesa:~/.ssh> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/userA/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/userA/.ssh/id_rsa.
Your public key has been saved in /home/userA/.ssh/id_rsa.pub.
The key fingerprint is:
fc:5c:59:1c:c0:96:6f:48:a4:94:41:20:d9:1f:08:00 userA@lnxslesa
The key's randomart image is:
+--[ RSA 2048]----+
|  E...o+.++=oo.  |
|      ..o.o.=. . |
|         ..+ oo  |
|       .  . .oo  |
|        S   o.   |
|         o .     |
|          o      |
|                 |
|                 |
+-----------------+
userA@lnxslesa:~/.ssh>
userA@lnxslesa:~/.ssh> ls -la
total 20
drwx------ 2 userA users 4096 Aug  9 16:29 .
drwxr-xr-x 6 userA users 4096 Aug  9 15:07 ..
```

```
-rw------- 1 userA users 1675 Aug  9 16:29 id_rsa
-rw-r--r-- 1 userA users  396 Aug  9 16:29 id_rsa.pub
-rw-r--r-- 1 userA users  228 Aug  9 15:07 known_hosts
userA@lnxslesa:~/.ssh>#
```

The command **ssh-keygen** generates id_rsa as a private key (it must be kept secret) and id_rsa.pub as a public key. You can use the `-t` flag to specify dsa if required, and the `-f` flag to generate keys at different locations.

2. On hostA, once you generate the public and private key pair, upload the public key to hostB using the **ssh-copy-id** command as shown in Example 4-42.

Example 4-42 Uploading hostA's public key to hostB

```
userA@lnxslesa:~/.ssh> ssh-copy-id -i /home/userA/.ssh/id_rsa.pub hostB
Password:
Now try logging into the machine, with "ssh 'lnxsu12'", and check in:
  .ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
userA@lnxslesa:~/.ssh>
```

3. On hostB, update the `/etc/ssh/sshd_config` file and make sure that these parameters are set as follows:
   - PubkeyAuthentication *yes*
   - AuthorizedKeysFile *%h/.ssh/authorized_keys*
   - hostBasedAuthentication *no*
   - IgnoreUserKnownHosts *no*
   - IgnoreRhosts *no*
   - PasswordAuthentication *no*
   - UsePAM *yes*
   - ChallengeResponseAuthentication *no*
4. On hostB, restart the ssh daemon (sshd) as shown in Example 4-43.

Example 4-43 Uploading hostA's public key to hostB

```
hostB:/etc/ssh # service sshd restart
Shutting down SSH daemon                                           done
Starting SSH daemon                                                done
```

5. **ssh** to hostA and log in as userA.
6. Now log in to hostB with the verbose flag.

Example 4-44 Public key authentication in OpenSSH

```
userA@lnxslesa:~/.ssh> ssh -v lnxsu12
OpenSSH_5.1p1, OpenSSL 0.9.8j-fips 07 Jan 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to lnxsu12 [9.12.4.225] port 22.
debug1: Connection established.
debug1: identity file /home/userA/.ssh/id_rsa type 1
debug1: identity file /home/userA/.ssh/id_dsa type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.1
```

```
debug1: match: OpenSSH_5.1 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.1
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-cbc hmac-md5 none
debug1: kex: client->server aes128-cbc hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host 'lnxsu12' is known and matches the RSA host key.
debug1: Found key in /home/userA/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Offering public key: /home/userA/.ssh/id_rsa
debug1: Server accepts key: pkalg ssh-rsa blen 277
debug1: read PEM private key done: type RSA
debug1: Authentication succeeded (publickey).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = en_US.UTF-8
Last login: Thu Aug  9 16:10:16 2012 from lnxslesa
userA@lnxsu12:~>
```

Note: If you are trying to ssh to a host for the first time, ssh will prompt you to add the host key.

Kerberos authentication

OpenSSH provides end-to-end encrypted communications in client-server architecture, and before starting communication, the OpenSSH server needs to authenticate the openSSH client. Kerberos is one of the most widely used authentication protocols; in this section we demonstrate detailed steps to integrate Kerberos with OpenSSH.

To simplify the demonstration of Kerberos authentication, we assume that hostA is the openSSH client and hostB is the server where the openSSH server is hosted. The hostKDC is the Key Distribution Center (KDC) which is the Kerberos server but also stores all user and host principal information in the Kerberos database. It also issues a Ticket Granting Ticket (TGT) to users after successful login to hosts. To configure Kerberos authentication, follow these steps:

1. Make sure that the Kerberos server RPM is installed on hostKDC. You can use the commands given in Example 4-45 on page 138 to check, install and reconfirm Kerberos server RPMs on the KDC server.

Example 4-45 Kerberos server RPMs on hostKDC

```
[root@hostKDC ~]# rpm -qa | grep -i krb
krb5-libs-1.9-22.el6.s390x
krb5-devel-1.9-22.el6.s390x
krb5-workstation-1.9-22.el6.s390x
pam_krb5-2.3.11-9.el6.s390x
[root@hostKDC ~]#
[root@hostKDC ~]#
[root@hostKDC ~]# yum install krb5-server
Loaded plugins: product-id, security, subscription-manager
Updating certificate-based repositories.
rhel                                               | 4.0 kB     00:00
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package krb5-server.s390x 0:1.9-22.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====================================================================
 Package           Arch       Version         Repository     Size
=====================================================================
Installing:
 krb5-server       s390x      1.9-22.el6      rhel           938 k

Transaction Summary
=====================================================================
Install       1 Package(s)

Total download size: 938 k
Installed size: 1.7 M
Is this ok [y/N]: y
Downloading Packages:
krb5-server-1.9-22.el6.s390x.rpm                   | 938 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : krb5-server-1.9-22.el6.s390x                         1/1
Installed products updated.

Installed:
  krb5-server.s390x 0:1.9-22.el6

Complete!
[root@hostKDC ~]#
[root@hostKDC ~]# rpm -qa | grep -i krb
krb5-libs-1.9-22.el6.s390x
krb5-devel-1.9-22.el6.s390x
krb5-workstation-1.9-22.el6.s390x
pam_krb5-2.3.11-9.el6.s390x
krb5-server-1.9-22.el6.s390x
[root@hostKDC ~]#
```

2. On hostKDC, create a Kerberos database using the **kdb5_util** command, as shown in Example 4-46.

Example 4-46 Creating a Kerberos database

```
[root@hostKDC ~]# kdb5_util create -r ITSO.IBM.COM -s
Loading random data
Initializing database '/var/kerberos/krb5kdc/principal' for realm 'ITSO.IBM.COM',
master key name 'K/M@ITSO.IBM.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
[root@hostKDC ~]#
```

Note: In the ITSO test environment, we used realm as ITSO.IBM.COM. You may need to change as per your environment.

3. On hostKDC, update the Kerberos access control list file and provide administrator access rights to admin to the Kerberos database. You can open the `/var/kerberos/krb5kdc/kadm5.acl` file in your favorite editor and make sure that the line shown in Example 4-47 is added (or updated) and saved successfully.

   Example 4-47 Updating Kerberos ACLs

   ```
   */admin@ITSO.IBM.COM    *
   ```

4. Configure a Kerberos client on all three hosts. You can open the `/etc/krb5.conf` file and make sure that all parameters given in Example 4-48 are set correctly in your environment.

   Example 4-48 Kerberos client configuration file

   ```
   [logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

   [libdefaults]
    default_realm = ITSO.IBM.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true

   [realms]
    ITSO.IBM.COM = {
     kdc = hostKDC.itso.ibm.com
     admin_server = hostKDC.itso.ibm.com
    }

   [domain_realm]
    .itso.ibm.com = ITSO.IBM.COM
    itso.ibm.com = ITSO.IBM.COM
   ```

```
[root@hostKDC ~]#
```

5. On hostKDC, start the Kerberos database administration daemon and KDC.

Example 4-49 Starting kadmin and krb5kdc daemons

```
[root@hostKDC ~]# service kadmin start
Starting Kerberos 5 Admin Server: [  OK  ]
[root@hostKDC ~]# service krb5kdc start
Starting Kerberos 5 KDC: [  OK  ]
```

6. On hostKDC, add an administrator to KDC using the **addprinc** command on **kadmin.local** prompt.

Example 4-50 Adding an administrator to KDC

```
[root@hostKDC ~]# kadmin.local -q "addprinc root/admin"
Authenticating as principal root/admin@ITSO.IBM.COM with password.
WARNING: no policy specified for root/admin@ITSO.IBM.COM; defaulting to no policy
Enter password for principal "root/admin@ITSO.IBM.COM":
Re-enter password for principal "root/admin@ITSO.IBM.COM":
Principal "root/admin@ITSO.IBM.COM" created.
[root@hostKDC ~]#
```

7. On hostKDC, add host principals for all three servers using **kadmin.local** prompt and the **addprinc** command.

Example 4-51 Adding host principals to KDC

```
[root@hostKDC ~]# kadmin.local -q "addprinc -randkey host/hostKDC.itso.ibm.com"
Authenticating as principal root/admin@ITSO.IBM.COM with password.
WARNING: no policy specified for host/hostKDC.itso.ibm.com@ITSO.IBM.COM;
defaulting to no policy
Principal "host/hostKDC.itso.ibm.com@ITSO.IBM.COM" created.
[root@hostKDC ~]#
[root@hostKDC ~]# kadmin.local -q "addprinc -randkey host/hostA.itso.ibm.com"
Authenticating as principal root/admin@ITSO.IBM.COM with password.
WARNING: no policy specified for host/hostA.itso.ibm.com@ITSO.IBM.COM; defaulting
to no policy
Principal "host/hostA.itso.ibm.com@ITSO.IBM.COM" created.
[root@hostKDC ~]#
[root@hostKDC ~]# kadmin.local -q "addprinc -randkey host/hostB.itso.ibm.com"
Authenticating as principal root/admin@ITSO.IBM.COM with password.
WARNING: no policy specified for host/hostB.itso.ibm.com@ITSO.IBM.COM; defaulting
to no policy
Principal "host/hostB.itso.ibm.com@ITSO.IBM.COM" created.
[root@hostKDC ~]#
```

8. On hostKDC, add host principals to the keytab file using **kadmin.local** prompt and the **ktadd** command.

Example 4-52 Adding host principals to the keytab file

```
[root@hostKDC ~]# kadmin.local
Authenticating as principal root/admin@ITSO.IBM.COM with password.
kadmin.local:  ktadd -k /var/kerberos/krb5kdc/kadm5.keytab
host/hostKDC.itso.ibm.com
```

```
Entry for principal host/hostKDC.itso.ibm.com with kvno 2, encryption type
aes256-cts-hmac-sha1-96 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostKDC.itso.ibm.com with kvno 2, encryption type
aes128-cts-hmac-sha1-96 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostKDC.itso.ibm.com with kvno 2, encryption type
des3-cbc-sha1 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostKDC.itso.ibm.com with kvno 2, encryption type
arcfour-hmac added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
kadmin.local:
kadmin.local:  ktadd -k /var/kerberos/krb5kdc/kadm5.keytab host/hostB.itso.ibm.com
Entry for principal host/hostB.itso.ibm.com with kvno 2, encryption type
aes256-cts-hmac-sha1-96 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostB.itso.ibm.com with kvno 2, encryption type
aes128-cts-hmac-sha1-96 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostB.itso.ibm.com with kvno 2, encryption type
des3-cbc-sha1 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostB.itso.ibm.com with kvno 2, encryption type
arcfour-hmac added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
kadmin.local:
kadmin.local:  ktadd -k /var/kerberos/krb5kdc/kadm5.keytab host/hostA.itso.ibm.com
Entry for principal host/hostA.itso.ibm.com with kvno 2, encryption type
aes256-cts-hmac-sha1-96 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostA.itso.ibm.com with kvno 2, encryption type
aes128-cts-hmac-sha1-96 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostA.itso.ibm.com with kvno 2, encryption type
des3-cbc-sha1 added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
Entry for principal host/hostA.itso.ibm.com with kvno 2, encryption type
arcfour-hmac added to keytab WRFILE:/var/kerberos/krb5kdc/kadm5.keytab.
kadmin.local:  quit
[root@hostKDC ~]#
```

9. On hostKDC, add user principal, that is, userA to the KDC server.

Example 4-53 Adding user principal

```
[root@hostKDC ~]# kadmin.local -q "addprinc userA"
Authenticating as principal root/admin@ITSO.IBM.COM with password.
WARNING: no policy specified for userA@ITSO.IBM.COM; defaulting to no policy
Enter password for principal "userA@ITSO.IBM.COM":
Re-enter password for principal "userA@ITSO.IBM.COM":
Principal "userA@ITSO.IBM.COM" created.
[root@hostKDC ~]#
```

10. Create userA on hostA and hostB.
11. On hostA, update the `/etc/ssh/ssh_config` file and make sure that parameters are set as follows:
    - Host **.itso.ibm.com*
    - GSSAPIAuthentication *yes*
    - GSSAPIDelegateCredentials *yes*
12. On hostB, update the `/etc/ssh/sshd_config` file and make sure that parameters are set as follows:
    - PubkeyAuthentication *no*
    - PasswordAuthentication *no*

– ChallengeResponseAuthentication *no*

– KerberosAuthentication *yes*

– GSSAPIAuthentication *yes*

– GSSAPICleanupCredentials *yes*

– UsePAM *no*

13. On hostB, restart the ssh daemon (*sshd*) using the **`server sshd restart`** command as shown in Example 4-54.

Example 4-54 Restarting sshd service

```
[root@hostB etc]# service sshd restart
Stopping sshd: [  OK  ]
Starting sshd: [  OK  ]
[root@hostB etc]#
```

14. Copy and rename `/var/kerberos/krb5kdc/kadm5.keytab` from hostKDC to hostB under `/etc/krb5.keytab`.

15. ssh to hostA and log in as userA.

16. On hostA, use the **`kinit`** command to cache Kerberos TGT. It will ask you for a password, provide it.

Example 4-55 Issuing Kerberos TGT

```
[userA@hostA ~]$ kinit
Password for userA@ITSO.IBM.COM:
[userA@hostA ~]$
```

17. On hostA, use the **`klist`** command to list cached Kerberos tickets.

Example 4-56 Listing cached Kerberos tickets

```
[userA@hostA ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_503
Default principal: userA@ITSO.IBM.COM

Valid starting     Expires            Service principal
08/17/12 11:19:34  08/18/12 11:19:34  krbtgt/ITSO.IBM.COM@ITSO.IBM.COM
        renew until 08/17/12 11:19:34
[userA@hostA ~]$
```

18. Now log in to hostB.

Example 4-57 Login to hostB

```
[userA@hostA ~]$ ssh hostB
Last login: Fri Aug 17 11:19:12 2012 from hostA.itso.ibm.com
[userA@hostB ~]$
```

19. On successful log in to *hostB*, log out and use the **`klist`** command to list cached Kerberos tickets; now you will see hostB's ticket along with the TGT ticket.

Example 4-58 Listing cached tickets after successful login and logout

```
[userA@hostB ~]$ logout
Connection to hostB closed.
[userA@hostA ~]$ klist
```

```
Ticket cache: FILE:/tmp/krb5cc_503
Default principal: userA@ITSO.IBM.COM

Valid starting     Expires            Service principal
08/17/12 11:19:34  08/18/12 11:19:34  krbtgt/ITSO.IBM.COM@ITSO.IBM.COM
        renew until 08/17/12 11:19:34
08/17/12 11:19:47  08/18/12 11:19:34  host/hostB.itso.ibm.com@ITSO.IBM.COM
        renew until 08/17/12 11:19:34
[userA@hostA ~]$
```

4.5.5 OpenSSH using the hardware cryptographic support of IBM System z

Common access methods for Linux servers such as telnet, or protocols for file transfer such as FTP, should be avoided not only in Internet environments but also in internal company networks. This is because sensitive information such as passwords is transferred over the network in the clear. The openSSH increases the security because the information sent across the network is encrypted. Not only passwords, but also data is protected by encryption technologies.

By its nature, encryption of data is expensive and can have a severe impact on performance, throughput, or processor load of a system. IBM System z provides hardware encryption support that can be used to reduce the impact of expensive encryption operations. Starting with OpenSSH version 4.4, OpenSSL dynamic engine loading is supported. This enables OpenSSH to benefit from IBM System z cryptographic hardware support, if a specific option (--with-ssl-engine) is used during the build of the OpenSSH package.

Here is a small overview of how OpenSSH accesses the hardware cryptographic support provided by IBM System z (see Figure 4-12 on page 144).

OpenSSH uses OpenSSL to perform the cryptographic requests. If the OpenSSH package is built using the option `--with-ssl-engine`, then the OpenSSL library can use available engines and load them dynamically. In a System z environment, you can install the ibmca engine and configure OpenSSL for dynamic engine loading. In this case, OpenSSL does not perform the encryption requests by itself, but passes them to the ibmca engine, which uses the library `libica` to handle the requests. The `libica` library is aware of which algorithms are supported by the underlying hardware (if installed and available) CPACF or Crypto Express feature. If an algorithm is supported by the underlying hardware, the `libica` library passes the request to the cryptographic hardware. If an algorithm is not supported by the underlying hardware, the `libica` library executes the algorithm in software as a fallback. The underlying virtualization layer of z/VM has no impact on the cryptographic architecture inside the Linux server.

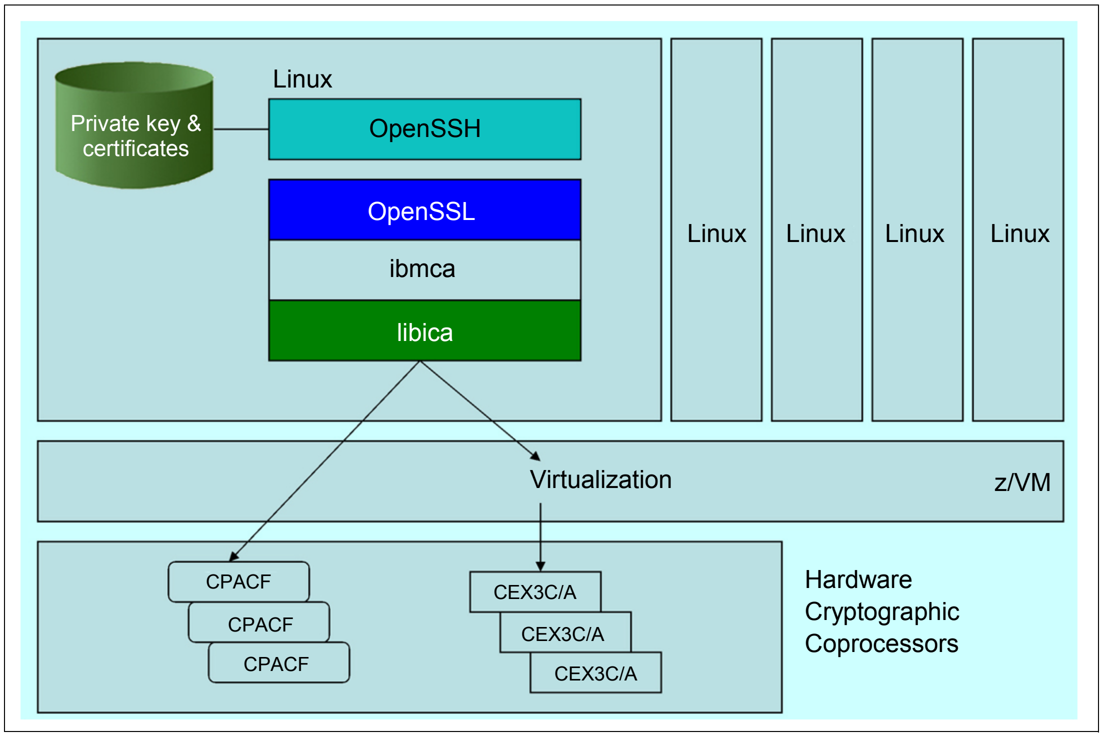


Figure 4-12 Linux for System z environment for hardware cryptographic support for OpenSSH

The following software and driver packages need to be installed on Linux for System z to enable OpenSSH to benefit from hardware cryptographic support on IBM System z.

- openssh
- openssl
- openssl-ibmca
- libica

These packages are part of the Linux for System z distribution. Some of them might be already installed with your initial setup. Probably, the packages openssl-ibmca and libica have to be installed separately.

We start with a RHEL 6.2 GA environment in which OpenSSL is installed, but dynamic engine loading support is not activated. The required `libibmca` libraries that contain the engine ibmca and the `libica` library are not yet installed (see Example 4-59).

Example 4-59 OpenSSL environment without dynamic engine loading

```
[root@lnxrh60b ~]# cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 1
bogomips per cpu: 18656.00
features        : esan3 zarch stfle msa ldisp eimm dfp etf3eh highgprs
processor 0: version = FF,  identification = 193BD5,  machine = 2817
[root@lnxrh60b ~]#
[root@lnxrh60b ~]# uname -a
Linux lnxrh60b 2.6.32-220.el6.s390x #1 SMP Wed Nov 9 08:20:08 EST 2011 s390x s390x
s390x GNU/Linux
[root@lnxrh60b ~]#
[root@lnxrh60b ~]# cat /etc/issue
Red Hat Enterprise Linux Server release 6.2 (Santiago)
```

```
Kernel \r on an \m

[root@lnxrh60b ~]#
[root@lnxrh60b ~]# rpm -qa | grep -i openssl
openssl-devel-1.0.0-20.el6.s390x
openssl-1.0.0-20.el6.s390x
pyOpenSSL-0.10-2.el6.s390x
[root@lnxrh60b ~]#
[root@lnxrh60b ~]# rpm -qa | grep -i libica
[root@lnxrh60b ~]#
[root@lnxrh60b ~]# rpm -qa | grep -i ibmca
[root@lnxrh60b ~]#
```

It can also be verified by an **`openssl engine`** command (see Example 4-60) that the ibmca engine for IBM System z is not yet available in your environment.

Example 4-60 OpenSSL speed test program: des-ede3-cdc in software on an IBM System z

```
[root@lnxrh60b ~]# openssl engine
(dynamic) Dynamic engine loading support
[root@lnxrh60b ~]#
[root@lnxrh60b ~]# openssl engine -c
(dynamic) Dynamic engine loading support
[root@lnxrh60b ~]#
```

To enable dynamic engine loading for Linux for System z, perform the following three steps:

1. Install and ensure that the ibmca engine is installed (see Example 4-61).

   Example 4-61 RPM command to ensure the ibmca engine is on the system

   ```
   [root@lnxrh60b ~]# rpm -qa | grep -i ibmca
   openssl-ibmca-1.2.0-2.el6.s390
   openssl-ibmca-1.2.0-2.el6.s390x
   [root@lnxrh60b ~]#
   ```

2. Install and ensure that the `libica` engine is installed (see Example 4-62).

   Example 4-62 RPM command to ensure that libica is on the system

   ```
   [root@lnxrh60b ~]# rpm -qa | grep -i libica
   libica1-0.43-5.1.el6.s390x
   libica-2.1.0-2.el6.s390x
   [root@lnxrh60b ~]#
   ```

3. Customize the OpenSSL configuration file to use dynamic engine loading.

   Note: You may need appropriate privileges in the system to perform these steps. Because the `libica` library and the engine ibmca support are shipped in separate RPM packages, you need to install them in addition to OpenSSL.

   As part of the openssl-ibmca package there is an `openssl.cnf.sample-xxx` file (see Example 4-63 on page 146). You can use this file to enable the use of the ibmca engine for applications that are aware of the dynamic engine support of OpenSSL.

Example 4-63 Sample openssl.cnf file

```
#
# OpenSSL example configuration file. This file will load the IBMCA engine
# for all operations that the IBMCA engine implements for all apps that
# have OpenSSL config support compiled into them.
#
# Adding OpenSSL config support is as simple as adding the following line to
# the app:
#
# #define OPENSSL_LOAD_CONF     1
#
openssl_conf = openssl_def

[openssl_def]
engines = engine_section

[engine_section]

foo = ibmca_section

[ibmca_section]
dynamic_path = /usr/lib64/openssl/engines/libibmca.so
engine_id = ibmca
init = 1
#
# The following ibmca algorithms will be enabled by these parameters
# to the default_algorithms line. Any combination of these is valid,
# with "ALL" denoting the same as all of them in a comma separated
# list.
#
# RSA
# - RSA encrypt, decrypt, sign and verify, key lengths 512-4096
#
# RAND
# - Hardware random number generation
#
# CIPHERS
# - DES-ECB, DES-CBC, DES-CFB, DES-OFB, DES-EDE3, DES-EDE3-CBC, DES-EDE3-CFB,
#   DES-EDE3-OFB, AES-128-ECB, AES-128-CBC, AES-128-CFB, AES-128-OFB,
#   AES-192-ECB, AES-192-CBC, AES-192-CFB, AES-192-OFB, AES-256-ECB,
#   AES-256-CBC, AES-256-CFB, AES-256-OFB symmetric crypto
#
# DIGESTS
# - SHA1, SHA256 digests
#
default_algorithms = ALL
#default_algorithms = RAND,RSA,CIPHERS,DIGESTS
```

To enable the ibmca engine, the OpenSSL configuration file has to be updated. To customize the OpenSSL configuration to enable dynamic engine loading for ibmca, perform the following four steps:

a. Do not forget to make a backup of the configuration file before you change it.

b. Append the content of the sample file `openssl.cnf.sample-s390x` (see Example 4-63) to the existing `openssl.cnf` file on the host.

c. Move the following line of the appended part to the top of the configuration file.

Example 4-64 OpenSSL configuration file update

```
openssl_conf = openssl_def
```

d. Check the value of the `dynamic path` variable and if necessary change it to the correct path, which is dependent on the Linux distribution in use.

The configuration file now looks as shown in Example 4-65. Note that this might not be mentioned in the README file of the current openssl-ibmca package.

Example 4-65 Updated openssl.cnf file

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd

openssl_conf = openssl_def

# Extra OBJECT IDENTIFIER info:
#oid_file               = $ENV::HOME/.oid
oid_section             = new_oids

--- some lines not displayed ---
--- some lines not displayed ---
--- some lines not displayed ---
--- some lines not displayed ---
#
# OpenSSL example configuration file. This file will load the IBMCA engine
# for all operations that the IBMCA engine implements for all apps that
# have OpenSSL config support compiled into them.
#
# Adding OpenSSL config support is as simple as adding the following line to
# the app:
#
# #define OPENSSL_LOAD_CONF     1
#
openssl_conf = openssl_def

[openssl_def]
engines = engine_section

[engine_section]

foo = ibmca_section

[ibmca_section]
dynamic_path = /usr/lib64/openssl/engines/libibmca.so
engine_id = ibmca
init = 1
```

```
#
# The following ibmca algorithms will be enabled by these parameters
# to the default_algorithms line. Any combination of these is valid,
# with "ALL" denoting the same as all of them in a comma separated
# list.
#
# RSA
# - RSA encrypt, decrypt, sign and verify, key lengths 512-4096
#
# RAND
# - Hardware random number generation
#
# CIPHERS
# - DES-ECB, DES-CBC, DES-CFB, DES-OFB, DES-EDE3, DES-EDE3-CBC, DES-EDE3-CFB,
#   DES-EDE3-OFB, AES-128-ECB, AES-128-CBC, AES-128-CFB, AES-128-OFB,
#   AES-192-ECB, AES-192-CBC, AES-192-CFB, AES-192-OFB, AES-256-ECB,
#   AES-256-CBC, AES-256-CFB, AES-256-OFB symmetric crypto
#
# DIGESTS
# - SHA1, SHA256 digests
#
default_algorithms = ALL
#default_algorithms = RAND,RSA,CIPHERS,DIGESTS
```

A first check of whether the dynamic engine support for ibmca is enabled can be done using the **openssl engine** command, as shown in Example 4-66.

Example 4-66 OpenSSL engine interface with dynamic engine ibmca

```
[root@lnxrh60b tls]# openssl engine
(dynamic) Dynamic engine loading support
(ibmca) Ibmca hardware engine support
[root@lnxrh60b tls]#
[root@lnxrh60b tls]# openssl engine -c
(dynamic) Dynamic engine loading support
(ibmca) Ibmca hardware engine support
 [RSA, DSA, DH, RAND, DES-ECB, DES-CBC, DES-OFB, DES-CFB, DES-EDE3, DES-EDE3-CBC,
DES-EDE3-OFB, DES-EDE3-CFB, AES-128-ECB, AES-128-CBC, AES-128-CFB, AES-128-OFB,
AES-192-ECB, AES-192-CBC, AES-192-CFB, AES-192-OFB, AES-256-ECB, AES-256-CBC,
AES-256-CFB, AES-256-OFB, SHA1, SHA256]
[root@lnxrh60b tls]#
```

To benefit from hardware support for OpenSSH, you need a minimum level of OpenSSH, and this package must be built using the option `--with-ssl-engine`. This option enables OpenSSH to request the dynamic engine loading support for the ibmca engine by OpenSSL.

After setting up and configuring OpenSSH for the use of the `libica` library and CPACF, you will want to check whether CPACF is really invoked for encryption when a cipher with an algorithm supported by CPACF is used. Besides the indirect proof by observing better performance, there are several options to verify whether the `libica` library and CPACF are used for encryption of data when OpenSSH is used.

The easiest way to check whether IBM System z hardware support is used during execution of the tests, is to check whether the library `libica` and ibmca are loaded into memory. We know that both `libica` and ibmca are present in memory only during execution of a program that needs services from them. Therefore, you can use a second SSH session to the Linux server and the **`lsof`** command to check whether a library is loaded or not.

Note: Starting with IBM System z10 EC, tracing of CPACF instructions is available. This function is for restricted use and only available in PEMODE, and therefore not for general client use.

Refer to Chapter 5, “Cryptographic hardware” on page 151 for more information about System z cryptographic functions.

5

Cryptographic hardware

In this chapter we describe how the System z cryptographic hardware and its functions can be used to provide value to workloads running on z/Linux.

For many years IBM has delivered products that implement cryptographic functions in specialty hardware. The functions implemented in hardware are typically a subset of those that can be utilized in software. The benefits of using hardware range to improved security, increased entropy, offloading cycles from the CP, and even acceleration. In recent years IBM has shipped the following cryptographic hardware products for mainframe servers:

1. Central Processor Assist for Cryptographic Functions (CPACF)
2. Cryptographic Express 3 Coprocessor (CEX3C)
3. Cryptographic Express 3Accelerator (CEX3A)
4. Cryptographic Express 3 Coprocessor (CEX3C)
5. Cryptographic Express 3Accelerator (CEX3A)

5.1 Clear key

In this chapter we introduce you to the clear key cryptography extensions that are available on System z10 through the Central Processor Assist for Cryptographic Function (CPACF) feature and the Crypto Express3 or Crypto Express4 features in accelerator mode (CEX3A and CEX3A). In the following sections, we show how these unique System z features can:

- Speed up handling of encrypted on disk data.
- Speed up encrypted communication.
- Improve security by providing advanced cryptographic functionality.

Note: Most of the examples in this chapter are based on Red Hat Enterprise Linux 5.4 but equally apply to SUSE Linux Enterprise Server 11, and in fact to most other Linux distributions as well. If there are, however, any significant differences between Red Hat Enterprise Linux 5.4 and SUSE Linux Enterprise Server 11, they are pointed out.

5.1.1 Accelerated Linux kernel functions

The standard Linux kernel includes modules that exploit the CPACF capabilities of the System z hardware. Through the `/proc` file system, you can query the kernel about the cryptographic modules that are currently in use. See Example 5-1.

Example 5-1 List of cryptographic modules currently in use by the Linux kernel

```
[nico@lnxrh2 ~]$ cat /proc/crypto
name         : crc32c
driver       : crc32c-generic
module       : kernel
priority     : 0
type         : digest
blocksize    : 32
digestsize   : 4

name         : sha1
driver       : sha1-generic
module       : kernel
priority     : 0
type         : digest
blocksize    : 64
digestsize   : 20
```

The hardware-dependent modules for the CPACF are typically located in the `/lib/modules` directory unless they have been compiled into the kernel itself. Example 5-2 and Example 5-3 on page 153 show modules that are available in a standard Red Hat Enterprise Linux 5.4 and SUSE Linux Enterprise Server 11.

Example 5-2 List of available s390 crypto modules in Red Hat Enterprise Linux 5.4

```
[nico@lnxrh2 ~]$ ls -alF /lib/modules/`uname -r`/kernel/arch/s390/crypto/
total 292
drwxr-xr-x 2 root root  4096 Sep 11 12:10 ./
drwxr-xr-x 6 root root  4096 Sep 11 12:10 ../
-rwxr--r-- 1 root root 36664 Aug 18 16:11 aes_s390.ko*
-rwxr--r-- 1 root root 27392 Aug 18 16:11 des_check_key.ko*
-rwxr--r-- 1 root root 41296 Aug 18 16:11 des_s390.ko*
```

```
-rwxr--r-- 1 root root 39552 Aug 18 16:11 prng.ko*
-rwxr--r-- 1 root root 36048 Aug 18 16:11 sha1_s390.ko*
-rwxr--r-- 1 root root 30696 Aug 18 16:11 sha256_s390.ko*
-rwxr--r-- 1 root root 32320 Aug 18 16:11 sha512_s390.ko*
```

As you can see in Example 5-3, SUSE Linux Enterprise Server 11 includes the same modules and an additional module called sha-common. As the name indicates, the module provides functions that are common to all SHA algorithms and have been split into a separate module.

Example 5-3 List of available s390 crypto modules in SUSE Linux Enterprise Server 11

```
nico@lnxsu2:~> ls -alF /lib/modules/`uname -r`/kernel/arch/s390/crypto/
total 96
drwxr-xr-x 2 root root  4096 2009-09-14 16:30 ./
drwxr-xr-x 7 root root  4096 2009-09-14 16:30 ../
-rw-r--r-- 1 root root 15976 2009-02-28 02:54 aes_s390.ko
-rw-r--r-- 1 root root  4840 2009-02-28 02:54 des_check_key.ko
-rw-r--r-- 1 root root 18416 2009-02-28 02:54 des_s390.ko
-rw-r--r-- 1 root root 11304 2009-02-28 02:54 prng.ko
-rw-r--r-- 1 root root  5664 2009-02-28 02:54 sha1_s390.ko
-rw-r--r-- 1 root root  5560 2009-02-28 02:54 sha256_s390.ko
-rw-r--r-- 1 root root  6800 2009-02-28 02:54 sha512_s390.ko
-rw-r--r-- 1 root root  6544 2009-02-28 02:54 sha_common.ko
```

You can load these modules manually, but usually the kernel takes care of that as soon as a cryptographic function is requested. Each module has a priority assigned to it, allowing the kernel to have two modules that are implementing the same functionality, and that is loaded and available at the same time. In this case, the kernel picks the module with the lowest priority to perform the cryptographic operation.

The following sections provide you with example Linux kernel functions that can be accelerated using these modules.

5.1.2 Random number generator

Another important aspect of cryptography is the availability of enough entropy to ensure secrecy because many cryptographic functions rely on randomly chosen values that are used, for example, to generate session keys or initialize the internal pseudorandom number generator (PRNG). The standard Linux devices from which random values are read are /dev/random and /dev/urandom. The difference is that /dev/random provides better quality random data and blocks if there are no values currently available, and /dev/urandom provides a constant stream of pseudorandom values.

CPACF provides a hardware PRNG that can deliver random numbers at a much higher rate than /dev/random or /dev/urandom and still provide statistically good random numbers.

In Example 5-4, you can see that, if you have udev (device manager) installed and running, loading the prng module is enough to create a new device, /dev/prandom, which is the source for the CPACF-generated random numbers. If you do not have udev installed or enabled you can still use **mknod** to create a character device with a major number of 10 and a minor number of 58.

Example 5-4 Loading the hardware PRNG module

```
[nico@lnxrh2 ~]$ sudo modprobe prng
```

```
[nico@lnxrh2 ~]$ ls -alF /dev/prandom
crw------- 1 root root 10, 58 Sep 14 17:45 /dev/prandom
```

If you have udev and you want to make the /dev/prandom area accessible for everyone, you have to add a new rule to udev that sets the correct permissions when the prng module is being loaded. Example 5-5 provides a sample udev rule to make /dev/prandom world-readable.

Example 5-5 The udev rules for prandom device permissions on Red Hat Enterprise Linux 5.4

```
[nico@lnxrh2 udev]$ echo 'KERNEL=="prandom", MODE="0444", OPTIONS="last_rule"' \
sudo tee -a /etc/udev/rules.d/99-user.rules
[nico@lnxrh2 udev]$ sudo rmmod prng
[nico@lnxrh2 udev]$ sudo modprobe prng
[nico@lnxrh2 udev]$ ll /dev/prandom
cr--r--r-- 1 root root 10, 59 Sep 14 18:13 /dev/prandom
```

You can test the speed of the new pseudorandom number generator by using the **dd** tool. Example 5-6 shows that the CPACF-assisted random number generator is in the authors' test system roughly twice as fast as the default generator that is being used for the /dev/urandom device. However, /dev/random drained after only 24 random bytes had been emitted in the 11 seconds that the test lasted.

Example 5-6 Speed comparison of Linux random number generators

```
[nico@lnxrh2 ~]$ dd if=/dev/random of=/dev/null bs=4k
0+3 records in
0+3 records out
24 bytes (24 B) copied, 11.0261 seconds, 0.0 kB/s
[nico@lnxrh2 ~]$ dd if=/dev/urandom of=/dev/null bs=4k
14975+0 records in
14974+0 records out
61333504 bytes (61 MB) copied, 11.0188 seconds, 5.6 MB/s
[nico@lnxrh2 ~]$ dd if=/dev/prandom of=/dev/null bs=4k
34746+0 records in
34745+0 records out
142315520 bytes (142 MB) copied, 11.0129 seconds, 12.9 MB/s
```

The hardware-supported PRNG can be used in any application that uses `/dev/random` or `/dev/urandom` to read random values. One example is the Apache httpd server program where you can use it for seeding the SSL or NSS pseudorandom number generator.

By default, the Apache server in SUSE uses the mod_ssl module for serving content over an encrypted communication channel. Example 5-7 shows the adjustments necessary for mod_ssl to access the PRNG device that utilizes CPACF.

Example 5-7 The /etc/apache2/ssl-global.conf configuration for using /dev/prandom with SLES 11

```
SSLRandomSeed startup file:/dev/prandom 1024
SSLRandomSeed connect file:/dev/prandom 512
```

Red Hat Enterprise Linux includes both mod_ssl and mod_nss for encrypted communication. The mod_ssl configuration is the same as in Example 5-7 for SUSE. Module mod_nss has its own configuration file and only supports seeding the internal random number generator at startup time, so only the one line presented in Example 5-8 on page 155 has to be modified or inserted into the mod_nss configuration file.

Example 5-8 The /etc/httpd/conf.d/nss.conf configuration for using /dev/prandom with RHEL 5.4

```
NSSRandomSeed startup file:/dev/prandom 512
```

Long random numbers

Long random numbers are also supported by System z. User-space applications can access large amounts of random data. The random data source is the built-in hardware random number generator on the CEX2C cards.

The requirements for generating and accessing long random numbers are:

- At least one CEX2C card must be installed on the system and be configured as a coprocessor (and the CCA library on this card must be at least version 3.30).
- Under z/VM, at least one CEX2C card must be configured as DEDICATED to the z/VM guest.
- Automatic creation of the random number character device requires udev.
- The cryptographic device driver z90crypt must be loaded.

If z90crypt detects at least one CEX2C card capable of generating long random numbers, a new miscellaneous character device is registered and can be found under `/proc/misc` as hw_random. The default rules provided with udev create a character device node called /dev/hwrng and a symbolic link /dev/hw_random pointing to /dev/hwrng.

For more information about long random number generation, see *Device Drivers, Features, and Commands available with SUSE Linux Enterprise Server 11,* SC34-2595.

5.2 File system encryption

The additional cryptographic capabilities of System z can also be used to speed up the process of protecting your file system data using encryption.

The four main ways to encrypt your data on disk with Linux are:

- Implementing data encryption in the application
- Using a file system that encrypts only the file contents on disk
- Using a file system with built-in encryption
- Encrypting a block device and using a regular file system on that block device

Clearly, the first approach is the least generic and most intrusive. If you choose this way, you can use the Crypto Express feature to speed up encryption and decryption. We are not going into detail because this is highly dependent on your application. For a sample of how you could access some of the Crypto Express features from a Java application, see “Java” on page 178.

The third option might give rise to side channel attacks by examining the file sizes, time stamps or other file attributes, which is why we cover only the third and fourth approaches in the next sections. They are also application-independent and currently available in both Red Hat Enterprise Linux 5.4 and SUSE Linux Enterprise Server 11.

eCryptfs

Native to the kernel, eCryptfs is a stacked cryptographic file system for Linux. A stacked file system is layered on top of an existing mounted file system, which is referred to as a lower file system. As the files are written to or read from the lower file system, eCryptfs encrypts and decrypts the files.[1]

To give applications and users a homogeneous view of the file system, Linux uses virtual file system (VFS) as an abstract interface to various underlying file system implementations. The VFS allows file systems to be layered on top of each other.

Figure 5-1 shows how eCryptfs works. It positions itself between the *userland* applications accessing the VFS and an arbitrary underlying file system. All file system operations form the userland pass through the VFS abstraction to eCryptfs. When eCryptfs receives a request to write data to a file, it encrypts the data and passes it to the underlying file system. If it receives a read request it asks the underlying file system for a chunk of encrypted data, decrypts it and passes the plain text back through the VFS to the userland.

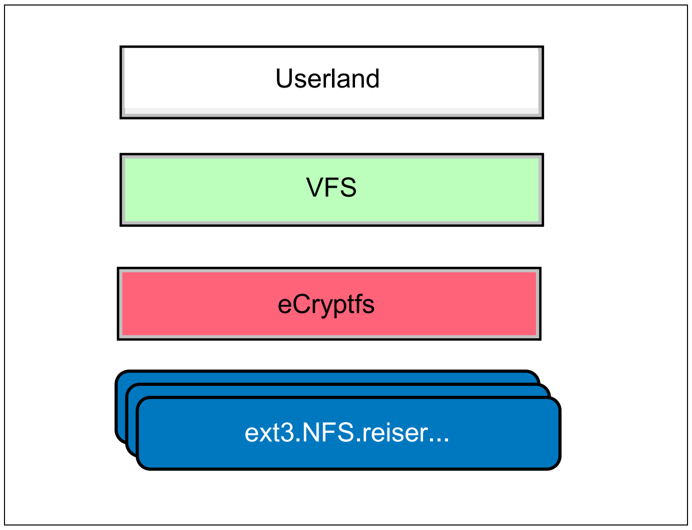


Figure 5-1 Diagram of where eCryptfs is logically positioned in the VFS stack

If the eCryptfs layer is removed, by unmounting it, the userland is able to access the data from the underlying file system. This data, however, will be encrypted and therefore not usable by the application any more.

Example 5-9 on page 157 shows you the necessary commands to create an encrypted file system with eCryptfs. The first step is to create the mount point and a directory where the encrypted data will be stored. Then, the directory that contains the encrypted files is mounted onto the mount point where the unencrypted files are written to, by using eCryptfs as the file system type.

eCryptfs automatically asks you a few questions about the encryption properties when you set up the file system. Although you can also pass all of these properties as mount options to automate the process, for this example we chose the interactive mode to illustrate how eCryptfs works. At the first prompt, you specify which key type to use. For this example, we used a simple passphrase but you can also supply an OpenSSL-readable key in a PEM file. The TSPI option is currently not supported on System z because there is no Trusted Platform Module (TPM) available to Linux.

After entering the password, you define which encryption algorithm should be used to protect your data. The CPACF supports the aes, des, and des3_ede128 modes. All other encryption is performed in software only. If the kernel modules for the selected encryption are not loaded yet, they will be loaded automatically. The key size that you specify is displayed in bytes, not bits so you have to multiply it by 8 to get the bit size. On System z10 hardware, the hardware supports AES key sizes up to 256 bits, which equals 32 bytes.

[1] `http://www.linuxjournal.com/article/9400`

The *plaintext passthrough* option specifies whether unencrypted files that are stored in the directory hierarchy where the encrypted files are stored should be accessible in the eCryptfs file system also. If you enable this option, an unencrypted file in `/mnt/encrypted` will be accessible in `/mnt/unencrypted`. If you do not enable this option, you will receive an error when trying to access the file through the `/mnt/encrypted` hierarchy. How you set this option depends on the requirements of your environments, but the authors advice would be to set this option to `NO` to avoid confusion and ensure that every file in the encrypted hierarchy really is encrypted.

Example 5-9 Creating and mounting an encrypted file system using eCryptfs

```
nico@lnxsu2:~> sudo mkdir /mnt/{un,}encrypted
nico@lnxsu2:~> sudo mount -t ecryptfs /mnt/encrypted /mnt/unencrypted
Select key type to use for newly created files:
 1) openssl
 2) tspi
 3) passphrase
Selection: 3
Passphrase: ****
Select cipher:
 1) aes: blocksize = 16; min keysize = 16; max keysize = 32 (loaded)
 2) blowfish: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 3) des3_ede: blocksize = 8; min keysize = 24; max keysize = 24 (not loaded)
 4) twofish: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 5) cast6: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 6) cast5: blocksize = 8; min keysize = 5; max keysize = 16 (not loaded)
 7) des: blocksize = 8; min keysize = 8; max keysize = 8 (loaded)
 8) des3_ede128: blocksize = 8; min keysize = 16; max keysize = 16 (loaded)
Selection [aes]: aes
Select key bytes:
 1) 16
 2) 32
 3) 24
Selection [16]: 24
Enable plaintext passthrough (y/n) [n]: n
Attempting to mount with the following options:
  ecryptfs_key_bytes=24
  ecryptfs_cipher=aes
  ecryptfs_sig=d395309aaad4de06
WARNING: Based on the contents of [/root/.ecryptfs/sig-cache.txt],
it looks like you have never mounted with this key
before. This could mean that you have typed your
passphrase wrong.

Would you like to proceed with the mount (yes/no)? yes
Would you like to append sig [d395309aaad4de06] to
[/root/.ecryptfs/sig-cache.txt]
in order to avoid this warning in the future (yes/no)? yes
Successfully appended new sig to user sig cache file
Mounted eCryptfs
```

After the encrypted file system has been created and mounted, you can access it like every other file system. See Example 5-10 on page 158.

Example 5-10 Creating files on an eCryptfs file system

```
nico@lnxsu2:~> find /mnt/{un,}encrypted
/mnt/unencrypted
/mnt/encrypted
nico@lnxsu2:~> echo hello world | sudo tee /mnt/unencrypted/test
hello world
nico@lnxsu2:~> file /mnt/{un,}encrypted/test
/mnt/unencrypted/test: ASCII text
/mnt/encrypted/test:   data
nico@lnxsu2:~> hexdump -C /mnt/unencrypted/test
00000000  68 65 6c 6c 6f 20 77 6f  72 6c 64 0a              |hello world.|
0000000c
nico@lnxsu2:~> hexdump -C /mnt/encrypted/test | head -n 5
00000000  00 00 00 00 00 00 00 0c  e2 3a 53 7f de bb e4 8a  |.........:S.....|
00000010  03 00 00 02 00 00 10 00  00 02 8c 2d 04 08 03 01  |...........-....|
00000020  00 11 22 33 44 55 66 77  60 4e 32 6c 34 60 5c 39  |.."3DUfw`N2l4`\9|
00000030  30 a2 d5 3c 19 31 3b 71  cc 38 42 a1 af 84 69 ab  |0..<.1;q.8B...i.|
00000040  77 bf 8b 17 a1 4f 4a 87  cd ed 16 62 08 5f 43 4f  |w....OJ....b._CO|
nico@lnxsu2:~> sudo umount /mnt/unencrypted
nico@lnxsu2:~> find /mnt/{un,}encrypted
/mnt/unencrypted
/mnt/encrypted
/mnt/encrypted/test
nico@lnxsu2:~> file /mnt/encrypted/test
/mnt/encrypted/test:   data
```

dm-crypt

One of the cryptography features of the Linux kernel is dm-crypt, which is a device mapper and a transparent disk encryption subsystem that allows you to encrypt whole block devices. It has recently gained popularity because many widely known Linux distributions such as SUSE, Red Hat, Debian, and Ubuntu support it in their installers.

The dm-crypt feature inserts a cryptographic layer between the block device and the accessing file systems or applications. You can use an encrypted block device like any other block device and install arbitrary file systems on it or use it as swap space. Figure 5-2 on page 159 shows where the dm-crypt encrypted device is logically positioned among real disks, the logical volume manager (LVM), and the file systems.

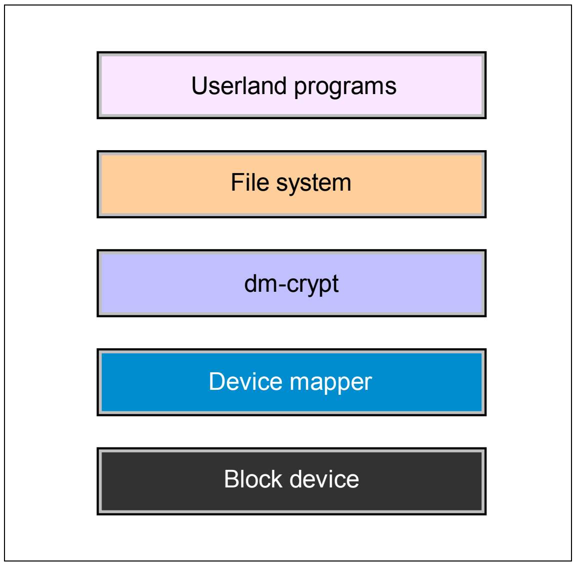


Figure 5-2 Logical position of dm-crypt in the block device and file system hierarchy

The administration of dm-crypt is done using **cryptsetup**, which features Linux Unified Key Setup (LUKS). LUKS standardizes the format of the encrypted disk which allows different implementations, even from other operating systems, to access and decrypt the disk. LUKS adds metadata to the underlying block device, which contains information about the ciphers used and a default of eight key slots that hold an encrypted version of the master key used to decrypt the device. You can unlock the key slots by either providing a password on the command line or using a key file, which could, for example, be encrypted with gpg and stored on an NFS share.

dm-crypt supports various cipher and hashing algorithms that you can choose from the ones that are available in the Kernel and listed in the `/proc/crypto procfs` file. This also means that dm-crypt will take advantage of the unique hardware acceleration features of System z which will increase encryption and decryption speed.

Example 5-11 shows how to set up an encrypted device on top of the LVM device vg0/test using a password to protect the encryption key. After formatting it with the LUKS metadata, the device is opened and a regular ext3 file system is created on the decrypted device.

Example 5-11 Setting up an encrypted device using dm-crypt and LUKS

```
nico@lnxsu2:~> sudo cryptsetup luksFormat /dev/mapper/vg0-test

WARNING!
========
This will overwrite data on /dev/mapper/vg0-test irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: ***
Verify passphrase: ***
Command successful.
nico@lnxsu2:~> sudo cryptsetup luksOpen /dev/mapper/vg0-test ctest
Enter LUKS passphrase: ***
key slot 0 unlocked.
Command successful.
nico@lnxsu2:~> sudo cryptsetup status ctest
/dev/mapper/ctest is active:
  cipher:  aes-cbc-essiv:sha256
  keysize: 128 bits
  device:  /dev/dm-1
  offset:  1032 sectors
  size:    2096120 sectors
  mode:    read/write
nico@lnxsu2:~> sudo mkfs.ext3 /dev/mapper/ctest
nico@lnxsu2:~> sudo mount /dev/mapper/ctest /mnt/test/
nico@lnxsu2:~> ls -alF /mnt/test/
total 24
drwxr-xr-x 3 root root  4096 2009-09-14 17:33 ./
drwxr-xr-x 3 root root  4096 2009-09-17 17:35 ../
drwx------ 2 root root 16384 2009-09-14 17:33 lost+found/
```

Subsequently, you can use the file system on the encrypted block device as you can with any other file system. You can also have the init process set up the encrypted file systems for you by creating the `/etc/crypttab` file. This file contains a description of all the encrypted block devices you want to set up before `/etc/fstab` is consulted. Example 5-12 shows the file for the device that was created in Example 5-11.

Example 5-12 The /etc/crypttab file for the device created in Example 5-11

```
#<target name> <source device> <key file> <options>
ctest    /dev/mapper/vg0-test    none    luks
```

Keep in mind that when you choose to protect the device with a password, you must enter it interactively during the boot process on the console; the boot process does not continue until the password has been entered. For more information about the possible crypttab options, see the crypttab in section 5 of the Linux man pages.

For a performance comparison see 5.5, "Statistics and performance" on page 181. Although only the performance results for IPSec are presented, you can expect the same kinds of improvements when using CPACF with dm-crypt or eCryptfs because the kernel modules that are used for all of these cryptographic operations are the same.

IPSec

"File system encryption" on page 155 showed how to secure your on-disk data using standard Linux features. In this chapter we focus on securing your data transfer using IPSec.

The IPSec protocol is used to encrypt on-wire IP traffic to counter eavesdropping and tampering by third parties. IPSec uses standard kernel cryptography and benefits from the System z hardware acceleration with CPACF.

IPSec has two major modes of operation:

- Tunnel mode
- Transport mode

In tunnel mode, IPSec can be compared to the Generic Routing Encapsulation (GRE) tunnel for IP packets or to OpenVPN in tunneling mode. The whole original IP packet, including the headers, is encrypted and signed and sent to the other remote host where it is being decrypted and possibly routed. You can use the IPSec tunnel mode to create virtual private networks consisting of many hosts communicating encrypted over an untrusted intermediary network.

In transport mode, only the payload of the IP packet is encrypted. The original IP headers remain and the packet is routed according to them. Because the headers are not encrypted, they could be tampered with while the packet is being routed through the network. To detect such tampering and discard modified packets, you may choose to sign the headers and add the signature as an authentication header. At the destination, the packet is decrypted and the payload is delivered to the receiving application. Transport mode can only be used to secure the communication between two hosts.

Figure 5-3 describes the example network setup that is used. IPSec is configured in transport mode with static keys. This setup encrypts all traffic flowing from host A to host B and backwards with a static encryption key and no key rotation algorithm. This is only an example setup; you might have different requirements for your environment and therefore are strongly encouraged to check with your network administrator and security officer.

For more information about running IPSec with Linux, see:

http://www.ipsec-howto.org

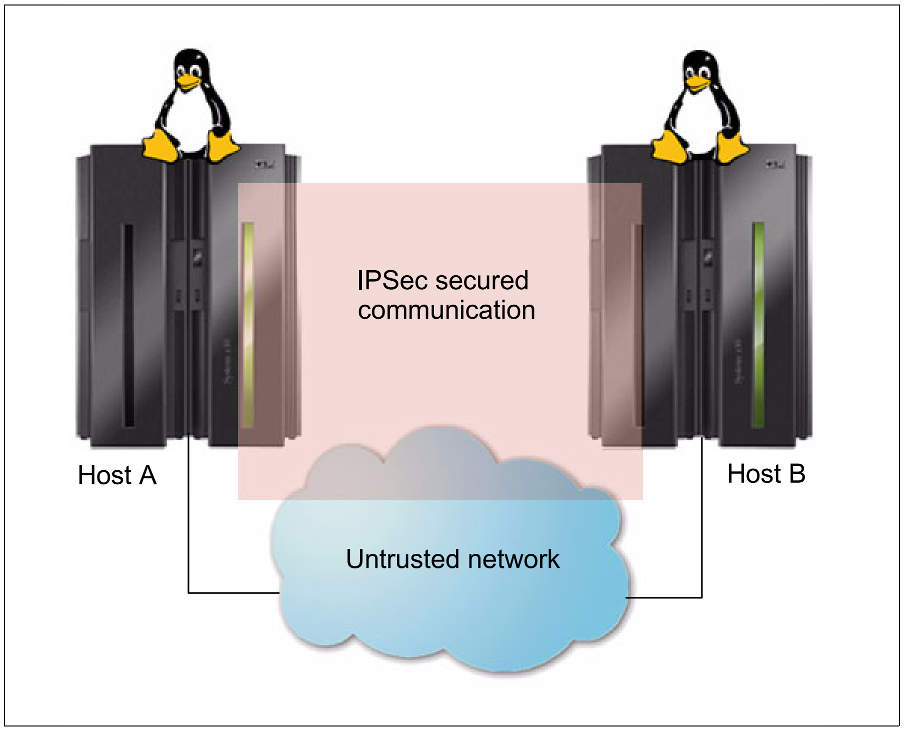


Figure 5-3 Sample network setup for IPSec

The encryption parameters are defined as a set of commands that are being fed to the setkey utility. First, the security policy and the security association database are flushed, and then new key authentication and encryption are added. Each key is associated with a direction and an algorithm. Make sure you choose one of the algorithms that are supported by CPACF. If in doubt, see the list that is presented when you run the **icainfo** command.

After adding the keys for both directions, A to B and B to A, the communication policies for the communication between both hosts are being defined. These policies dictate how the Linux kernel will communicate with the remote host and what kind of traffic it will accept. The configuration in Example 5-13 on page 162 requires all traffic to be encrypted and also mandates an authentication header to secure the IP headers.

Example 5-13 Configuration file for host A

```
# Flush the security policy database.
spdflush;
# Flush the security association database.
flush;

# Define authentication and encryption algorithms and keys.
add 192.168.33.129 192.168.33.1   ah   0xc001 -A hmac-sha256
0x1234567890123456123456789012345612345678901234561234567890123456;
add 192.168.33.129 192.168.33.1   esp  0xc002 -E aes-cbc
0x12345678901234561234567890123456123456789012345 6;

add 192.168.33.1   192.168.33.129 ah   0xc011 -A hmac-sha256
0x1234567890123456123456789012345612345678901234561234567890123456;
add 192.168.33.1   192.168.33.129 esp  0xc012 -E aes-cbc
0x123456789012345612345678901234561234567890123456;

# Define the security policies.
spdadd  192.168.33.129 192.168.33.1   any -P in ipsec
    esp/transport//require
     ah/transport//require;

spdadd  192.168.33.1   192.168.33.129 any -P out ipsec
    esp/transport//require
     ah/transport//require;
```

The configuration file for host B looks almost exactly like the one for host A. The only difference is in the policies, where you have to invert the defined directions in and out. See Example 5-14.

Example 5-14 Configuration for host B

```
# Flush the security policy database.
spdflush;
# Flush the security association database.
flush;

# Define authentication and encryption algorithms and keys.
add 192.168.33.129 192.168.33.1   ah   0xc001 -A hmac-sha256
0x1234567890123456123456789012345612345678901234561234567890123456;
add 192.168.33.129 192.168.33.1   esp  0xc002 -E aes-cbc
0x123456789012345612345678901234561234567890123456;

add 192.168.33.1   192.168.33.129 ah   0xc011 -A hmac-sha256
0x1234567890123456123456789012345612345678901234561234567890123456;
add 192.168.33.1   192.168.33.129 esp  0xc012 -E aes-cbc
0x123456789012345612345678901234561234567890123456;

# Define the security policies.
spdadd  192.168.33.129 192.168.33.1   any -P out ipsec
    esp/transport//require
     ah/transport//require;

spdadd  192.168.33.1   192.168.33.129 any -P in ipsec
    esp/transport//require
```

```
     ah/transport//require;
```

The new keys and policies are applied by using the **setkey** tool with the configuration files. Remember, these settings only last until the next reboot so you have to consider adding the command from Example 5-15 to /etc/rc.local or a similar script that is being called at boot time.

Example 5-15 Activating the manual ipsec configuration

```
sudo setkey -f /etc/ipsec.manual
```

The new settings take effect immediately. You can see the results clearly when you start a **ping** from host A to host B and use **tcpdump** to trace the network packets as in Example 5-16. Because we configured the transport mode to apply to any protocol, we do not have to perform any special configuration for the **ping** messages to be sent encrypted over the network.

Example 5-16 Packet trace for a ping with IPSec in transport mode

```
nico@lnxsu2:~> sudo tcpdump -tpni eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
IP 192.168.33.1 > 192.168.33.129: AH(spi=0x0000c011,seq=0xddf):
ESP(spi=0x0000c012,seq=0xddf), length 104
IP 192.168.33.129 > 192.168.33.1: AH(spi=0x0000c001,seq=0x9):
ESP(spi=0x0000c002,seq=0x9), length 104
IP 192.168.33.1 > 192.168.33.129: AH(spi=0x0000c011,seq=0xde0):
ESP(spi=0x0000c012,seq=0xde0), length 104
IP 192.168.33.129 > 192.168.33.1: AH(spi=0x0000c001,seq=0xa):
ESP(spi=0x0000c002,seq=0xa), length 104
```

For performance considerations with IPSec, see 5.5, “Statistics and performance” on page 181.

5.2.1 Key management

One important aspect of cryptography is managing the keys that give access to the unencrypted data. Commonly, these keys are stored in an encrypted file on the file system and are decrypted to access them. But this form of storing the keys allows anyone to make a copy of the encrypted key and try to decrypt it. It also allows insiders, who have access to the unencrypted key, to copy it without any trace.

To counter that risk, the keys can be stored in security modules. Security modules do not grant access to the contained key but have an API that allows users to request encryption and signature operations to be performed. These security modules are usually of two types: Hardware Security Module (HSM) or Software Security Module (SSM).

The Crypto Express3 and Crypto Express4 adapters in accelerator mode can be used as an HSM that protects your private keys from being read even by legitimate key users. The only difference to a full featured HSM, like the Crypto Express card in coprocessor mode, is that the encrypted keys are still stored in the file system of the user operating system. This situation implies that you take the necessary precautions to prevent the key from being deleted or overwritten, and implement a backup process according to your company’s policies.

If you need a full HSM, you can use the Crypto Express adapters in coprocessor mode. See 5.6, “Secure Key Crypto” on page 183 for more information about secure key processing.

5.3 Cryptographic APIs

PKCS#11

The most widespread standard for access tokens to security modules is PKCS#11 from the RSA Laboratories, nicknamed *Cryptoki*. PKCS#11 allows applications to access cryptographic objects stored in a token in a unified way. Every PKCS#11 device has a set of slots in which you can place tokens that in turn can hold cryptographic objects or perform cryptographic operations.

Both Red Hat Enterprise Linux 5.4 and SUSE Linux Enterprise Server 11 include the necessary tools to initialize and access the security tokens. To access the CEX3A adapter, all of the tools work with the openCryptoki library, which is available in the correspondent package in both distributions. Additionally, IBM provides the GSKit toolset to manage tokens with a PKCS#11 interface.

Before you start working with the PKCS#11 API, be sure that the z90crypt module is loaded and the `/dev/z90crypt` device is present. This module provides access to the Crypto Express adapter that will be used by the ibmca PKCS#11 implementation. You also have to enable and start the slot daemon that coordinates access to the security tokens. To allow users other than root to communicate with the slot daemon, you have to add them to the pkcs11 group.

The next step is to set a new PIN on the slot that you want to use, initialize the token, and set a user PIN on the token. For security reasons, the user PIN should be changed by the user as soon as the user is granted access. This step ensures that the slot operator has no access to the token itself. Example 5-17 details the commands to set up the slot and token. Slot #0 is used here to access the clear key cryptographic functions on the CEX3A adapter.

Example 5-17 Change the PIN for slot 0 and initialize the token

```
[nico@lnxrh2 ~]$ pkcsconf -c 0 -P
Enter the SO PIN: ****
Enter the new SO PIN: ****
Re-enter the new SO PIN: ****
[nico@lnxrh2 ~]$ pkcsconf -c 0 -I
Enter the SO PIN: ****
Enter a unique token label: TOK01
[nico@lnxrh2 ~]$ pkcsconf -c 0 -u
Enter the SO PIN: ****
Enter the new user PIN: ****
Re-enter the new user PIN: ****
[nico@lnxrh2 ~]$ pkcsconf -c 0 -p
Enter user PIN: ****
Enter the new user PIN: ****
Re-enter the new user PIN: ****
[nico@lnxrh2 ~]$ pkcsconf -i -s -t
PKCS#11 Info
        Version 2.11
        Manufacturer: IBM
        Flags: 0x0
        Library Description: Meta PKCS11 LIBRARY
        Library Version 2.2
```

```
Token #0 Info:
        Label: TOK01
        Manufacturer: IBM Corp.
        Model: IBM ICA
        Serial Number: 123
        Flags: 0x44D
(RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|CLOCK_ON_TOKEN|TOKEN_INITIALIZED)
        Sessions: -1/-1
        R/W Sessions: -1/-1
        PIN Length: 4-8
        Public Memory: 0xFFFFFFFF/0xFFFFFFFF
        Private Memory: 0xFFFFFFFF/0xFFFFFFFF
        Hardware Version: 1.0
        Firmware Version: 1.0
        Time: 09:09:14 AM
Slot #0 Info
        Description: Linux 2.6.18-164.el5 Linux (ICA)
        Manufacturer: Linux 2.6.18-164.el5
        Flags: 0x5 (TOKEN_PRESENT|HW_SLOT)
        Hardware Version: 0.0
        Firmware Version: 1.1
```

The token that we just created is a hardware token but is not stored inside the Crypto Express3 or Crypto Express4 feature; it is on the local file system in `/var/lib/opencryptoki/lite`. Be sure that access to this directory is properly restricted.

s

Note: Only for our example purposes is the slot operator and user of the token the same person. In real use, you should assign two separate roles for the technical task of setting up slots and the security task of managing tokens. Neither of the two roles involved should know the other's PIN. This approach prevents the security operator from gaining access to the tokens' content, and the token user from manipulating the slot and other tokens that might be on that slot.

For more information about using openCryptoki on Linux, go to:

http://www.ibm.com/developerworks/linux/library/s-pkcs/

NSS library

Starting with version 5.4, Red Hat Enterprise Linux supports Network Security Services (NSS), a library for managing cryptographic objects and performing cryptographic functions in a platform-independent way. NSS started out as a Netscape project and is now being developed by the Mozilla Foundation. It is also used for key management by other vendors, for example in the Sun Solaris operating system.

The first step when working with NSS is to create and initialize a new database. This is done with the **`certutil`** utility, as described in Example 5-18. The password you have to enter is used to protect the default internal NSS key database. It is not the PIN of your token but you should remember it anyway because some applications might prompt you for it.

Example 5-18 Initializing a NSS key database with certutil

```
[nico@lnxrh2 ~]$ sudo mkdir /etc/httpd/nss/
[nico@lnxrh2 ~]$ sudo certutil -N -d /etc/httpd/nss
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password: ****
Re-enter password: ****
```

To use the openCryptoki library together with NSS you have to define it in the NSS database as an available PKCS#11 implementation module. Modifications to the list of available PKCS#11 implementations are done using the **modutil** tool. Example 5-19 shows the information about the currently defined implementation, which is only the default PKCS#11 module that is provided with NSS in Red Hat Enterprise Linux 5.4.

Example 5-19 List of installed default PKCS#11 NSS modules

```
[nico@lnxrh2 ~]$ sudo modutil -dbdir /etc/httpd/nss/ -list

Listing of PKCS #11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS #11 Module
         slots: 2 slots attached
        status: loaded

         slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services

         slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB
-----------------------------------------------------------
```

To make the openCryptoki implementation available, use the **modutil -add** command on the key database that you created in a previous step. Example 5-20 shows how to add openCryptoki for a defined set of encryption, hash, and random operations and how to make it the default implementation for those operations.

Example 5-20 Adding the openCryptoki implementation to the list of PKCS#11 implementations

```
[nico@lnxrh2 ~]$ sudo modutil -dbdir /etc/httpd/nss/ -add opencryptoki \
-libfile /usr/lib64/opencryptoki/PKCS11_API.so -mechanisms \
RSA:DH:DES:AES:SHA1:MD5:SHA256:SHA512:RANDOM:SSL:TLS

WARNING: Performing this operation while the browser is running could cause
corruption of your security databases. If the browser is currently running,
you should exit browser before continuing this operation. Type
'q <enter>' to abort, or <enter> to continue:

Module "opencryptoki" added to database.
[nico@lnxrh2 ~]$ sudo modutil -dbdir /etc/httpd/nss/ -default opencryptoki
-mechanisms RSA:DH:DES:AES:SHA1:MD5:SHA256:SHA512:RANDOM:SSL:TLS

WARNING: Performing this operation while the browser is running could cause
corruption of your security databases. If the browser is currently running,
you should exit browser before continuing this operation. Type
'q <enter>' to abort, or <enter> to continue:

Successfully changed defaults.
[nico@lnxrh2 ~]$ sudo modutil -dbdir /etc/httpd/nss/ -list
```

```
Listing of PKCS #11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS #11 Module
         slots: 2 slots attached
        status: loaded

         slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services

         slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB

  2. opencryptoki
        library name: /usr/lib64/opencryptoki/PKCS11_API.so
         slots: 1 slot attached
        status: loaded

         slot: Linux 2.6.18-164.el5 Linux (ICA)
        token: TOK01
-----------------------------------------------------------
```

The token is now available to the NSS library and we can start adding keys and generating certificate requests. Example 5-21 outlines the steps involved to create a certificate request for a new 2048-bit RSA key that is stored in the previously created token. The hardware PRNG is being used to seed the key generation. The **`certutil -K`** command finally lists the generated key in the token.

Example 5-21 Generating a new key and certificate request using certutil

```
[nico@lnxrh2 ~]$ dd if=/dev/prandom bs=1M count=1 | \
sudo certutil -R -d /etc/httpd/nss -h TOK01 \
-s 'C=US,ST=New York,L=Poughkeepsie,O=IBM,OU=ITSO,CN=lnxrh2.itso.ibm.co' \
-o lnxrh2.req -a -k rsa -g 2048 -n key01
Enter Password or Pin for "TOK01":

A random seed must be generated that will be used in the
creation of your key.  One of the easiest ways to create a
random seed is to use the timing of keystrokes on a keyboard.

To begin, type keys on the keyboard until this progress meter
is full.  DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!


Continue typing until the progress meter is full:

|************************************************************|

Finished.  Press enter to continue:


Generating key.  This may take a few moments...

[nico@lnxrh2~]$ sudo certutil -K -d /etc/httpd/nss -h TOK01
certutil: Checking token "TOK01" in slot "Linux 2.6.18-164.el5 Linux (ICA)"
Enter Password or Pin for "TOK01":
```

```
< 0> rsa      0b82248c5e011d1ca32821e76d7c2f6590f39a39   (orphan)
```

The certificate request in `lnxrh2.req` can now be transferred to a certificate authority (CA) for signing. After it has been signed, we have to import the certificate into the NSS database to use it in conjunction with the key. Importing the certificate is also done using the **`certutil`** command as shown in Example 5-22.

Example 5-22 Importing a signed certificate into the NSS database

```
[nico@lnxrh2 ~]$ sudo certutil -A -d /etc/httpd/nss -n key01 -h TOK01 -a \
-i lnxrh2.crt -t u,u,u
[nico@lnxrh2 ~]$ sudo certutil -L -d /etc/httpd/nss -h TOK01

Certificate Nickname                                         Trust Attributes
                                                             SSL,S/MIME,JAR/XPI

Enter Password or Pin for "TOK01":
TOK01:test-ca                                                c,c,c
TOK01:key01                                                  u,u,u
[nico@lnxrh2 ~]$ sudo certutil -K -d /etc/httpd/nss -h TOK01
certutil: Checking token "TOK01" in slot "Linux 2.6.18-164.el5 Linux (ICA)"
Enter Password or Pin for "TOK01":
< 0> rsa      0b82248c5e011d1ca32821e76d7c2f6590f39a39   TOK01:key01
```

The token now contains a key and an accompanying certificate and can be used for cryptographic functions by programs that are enabled to work with the NSS PKCS#11 library.

The pkcs11-helper package

Configuring the token in SUSE Linux Enterprise Server 11 is performed by using **`pkcs11-tool`**, which is provided by the pkcs11-helper software package. Start by installing additional packages that are required to work with the openCryptoki PKCS#11 implementation and the IBM ICA module. See Example 5-23.

Example 5-23 Packages required for PKCS#11 access on SUSE Linux Enterprise Server 11

```
sudo zypper install engine_pkcs11 pkcs11-helper
```

Make sure the `pkcsslotd` daemon is started and you have added your user to the pkcs11 group. Initialize the token by using the **`pkcsconf`** tool as shown in Example 5-17 on page 164. If everything is configured correctly, you should be able to list the available slots similar to Example 5-24.

Example 5-24 List of uninitialized slots after starting pkcsslotd for the first time

```
nico@lnxsu2:/var/lib/opencryptoki/lite> pkcs11-tool --module \
/usr/lib64/opencryptoki/libopencryptoki.so  -L
Available slots:
Slot 0           Linux 2.6.27.19-5-default Linux (ICA)
  token label:   TOK01
  token manuf:   IBM Corp.
  token model:   IBM ICA
  token flags:   rng, login required, PIN initialized, token initialized, other
flags=0x800040
  serial num  :  123
```

Interfacing with the cryptographic token on SUSE Linux Enterprise Server 11 also requires OpenSSL support to generate certificate requests and convert between file formats. Access to the token is provided by the pkcs11 engine, which has to be configured in the `openssl.cnf` file. The openssl-ibmca package includes an example for the configuration statements in the following file:

`/usr/share/doc/packages/openssl-ibmca/openssl.cnf.sample`

Copy the contents of that file to the top of `/etc/openssl.cnf` and adapt them as in Example 5-25 to make the pkcs11 engine globally available.

Example 5-25 OpenSSL configuration for dynamic pkcs11 engine support

```
openssl_conf    = openssl_def

[openssl_def]
engines = engine_section

[engine_section]
pkcs11 = pkcs11_section

[pkcs11_section]
engine_id = pkcs11
dynamic_path = /usr/lib64/engines/engine_pkcs11.so
MODULE_PATH = /usr/lib64/opencryptoki/libopencryptoki.so
default_algorithms = ALL
init = 1
```

If the engine has been configured correctly, you should see output similar to Example 5-26 when you run the **`openssl engine -c`** command.

Example 5-26 OpenSSH engine list after pkcs11 engine has been defined

```
nico@lnxsu2:~> openssl engine -c
(dynamic) Dynamic engine loading support
(pkcs11) pkcs11 engine
 [RSA, DSA, DH, RAND]
```

We proceed by creating a new RSA key in the token with the **`pkcs11-tool`** command and generating a certificate request for it. Example 5-27 shows how to use the pkcs11 OpenSSL engine to access the new private key on the token. The engine uses the openCryptoki as the backend, which in turn accesses the ICA library to manipulate the token.

Example 5-27 Generating a new key on the token using the pkcs11-tool command

```
nico@lnxsu2:~> pkcs11-tool --module /usr/lib64/opencryptoki/libopencryptoki.so -k
--key-type rsa:2048 --private -a key01 -l
Please enter User PIN: ****
Key pair generated:
Private Key Object; RSA
  label:      key01
  Usage:      decrypt, sign, unwrap
Public Key Object; RSA 2048 bits
  label:      key01
  Usage:      encrypt, verify, wrap
nico@lnxsu2:~> pkcs11-tool --module /usr/lib64/opencryptoki/libopencryptoki.so \
-O -l
Please enter User PIN: ****
Private Key Object; RSA
  label:      key01
  Usage:      decrypt, sign, unwrap
nico@lnxsu2:~> openssl req -engine pkcs11 -new -out lnxsu2.req \
-keyform engine -key slot_0-label_key01 \
-subj '/CN=lnxsu2.itso.ibm.co/OU=ITSO/O=IBM/L=Poughkeepsie/ST=New York/C=US'
engine "pkcs11" set.
PKCS#11 token PIN: ****
```

The certificate request in `lnxsu2.req` can now be sent off to a certificate authority for signing. For an example of how to use the z/VM certificate management, see "Implementing a certification authority in z/VM" on page 17. After the request has been signed by a CA and the certificate has been transferred back to the Linux server, it has to be imported into the token. See Example 5-28. Usually, the certificate is made available to you as an ASCII armored DER file, but **pkcs11-tool** requires that the certificate be in plain DER format, so we use OpenSSL to convert it.

Example 5-28 Importing the signed certificate into the token using pkcs11-tool

```
nico@lnxsu2:~> openssl x509 -in lnxsu2.crt -outform der -out lnxsu2.crt.der
nico@lnxsu2:~> pkcs11-tool --module /usr/lib64/opencryptoki/libopencryptoki.so -w
lnxsu2.crt.der -l --label 'key01' -y cert
Please enter User PIN:
Generated certificate:
Certificate Object, type = X.509 cert
  label:      key01
nico@lnxsu2:~> pkcs11-tool --module /usr/lib64/opencryptoki/libopencryptoki.so -L
-l -O
Available slots:
Slot 0           Linux 2.6.27.19-5-default Linux (ICA)
  token label:   TOK01
  token manuf:   IBM Corp.
  token model:   IBM ICA
  token flags:   rng, login required, PIN initialized, token initialized, other
flags=0x800040
  serial num  :  123
Please enter User PIN:
Private Key Object; RSA
  label:      key01
  Usage:      decrypt, sign, unwrap
Certificate Object, type = X.509 cert
  label:      key01
```

GSKit

To access and manage your PKCS#11 tokens, you can also use the Global Security Toolkit (GSKit) from IBM, an API that provides platform-independent functions for secure communication using SSL. GSKit supports access to cryptographic tokens and functions since version 7. GSKit is bundled with other IBM software, the most prominent being the IBM HTTP Server, which is discussed in "IBM HTTP Server" on page 178. See the respective manuals of those software packages for a description of how to install GSKit.

One of the utilities that is provided by GSKit is iKeyman. It allows you to effectively manage key stores through the GSKit interfaces and supports various key store formats by using a plug-in mechanism. One of the plug-ins enables GSKit to access PKCS#11 key stores, and you can use it to access and manipulate the contents of the security tokens managed by openCryptoki.

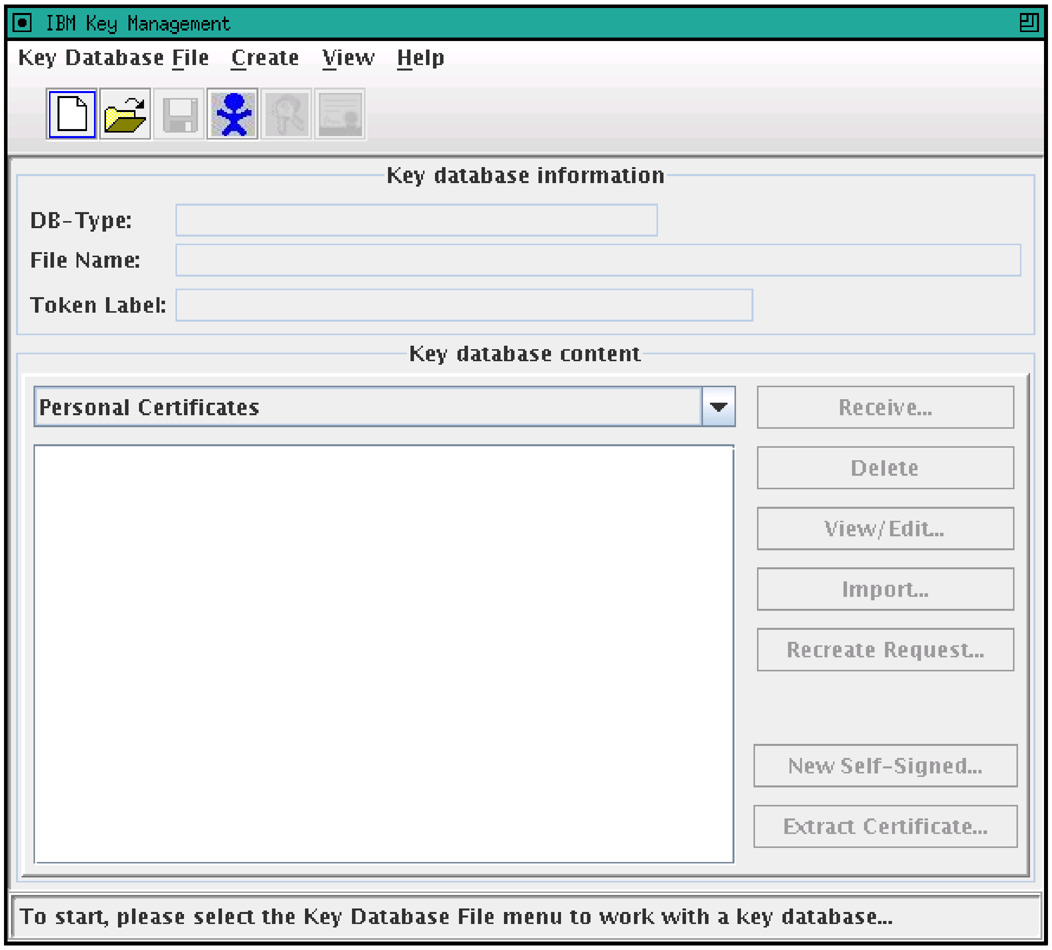
This section walks you through the process of:

1. Creating a new key on the security token.
2. Creating a singing request.
3. Importing the signed certificate for the key and the CA certificate.

Note: The iKeyman utility is GUI-based. You must have either a remote X server or a local X server and X forwarding to access it. Setting up an X environment is beyond the scope of this book; see the documentation of your Linux distribution. The following steps assume you have set up your X environment.

Follow these steps:

1. Make sure that you followed the steps for setting up the token outlined in "PKCS#11" on page 164 and then start iKeyman. The main window of the utility is shown in Figure 5-4. Because no database has yet been opened, no certificates are displayed.



To start, please select the Key Database File menu to work with a key database...

Figure 5-4 The iKeyman key store management utility main window

2. Open a database:

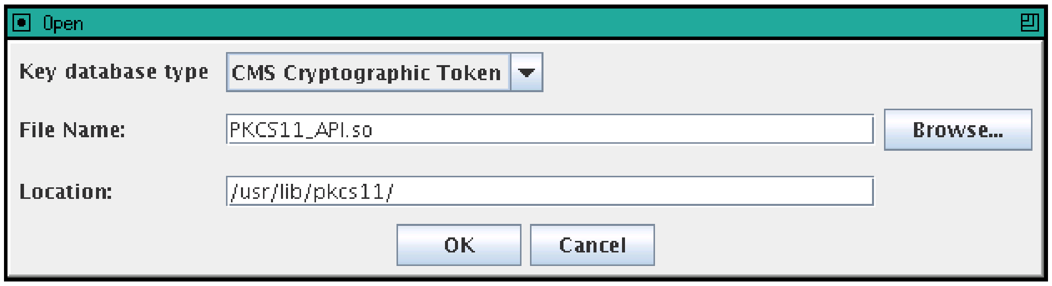
   a. Either select **Key Database** → **Open** or click the **Open a key database file** toolbar item. A new dialog opens, as shown in Figure 5-5 on page 172.

   b. You are prompted to select a database type, a file name, and a location. From the drop-down menu, select **CMS Cryptographic Token** for the database type.

Note: If you get the error message `The CMS Java Native library was not found.`, make sure you have applied the latest fix pack for your software product. The Technote database contains entries that might help you resolve this error. For example, the IBM HTTP Server is addressed in Technote 1247000, which can be found at:

`http://www.ibm.com/support/docview.wss?uid=swg21247000`

c. Depending on whether you have 64-bit support, select the appropriate `PKCS11_API.so` shared library from your file system and click **OK**.



Open

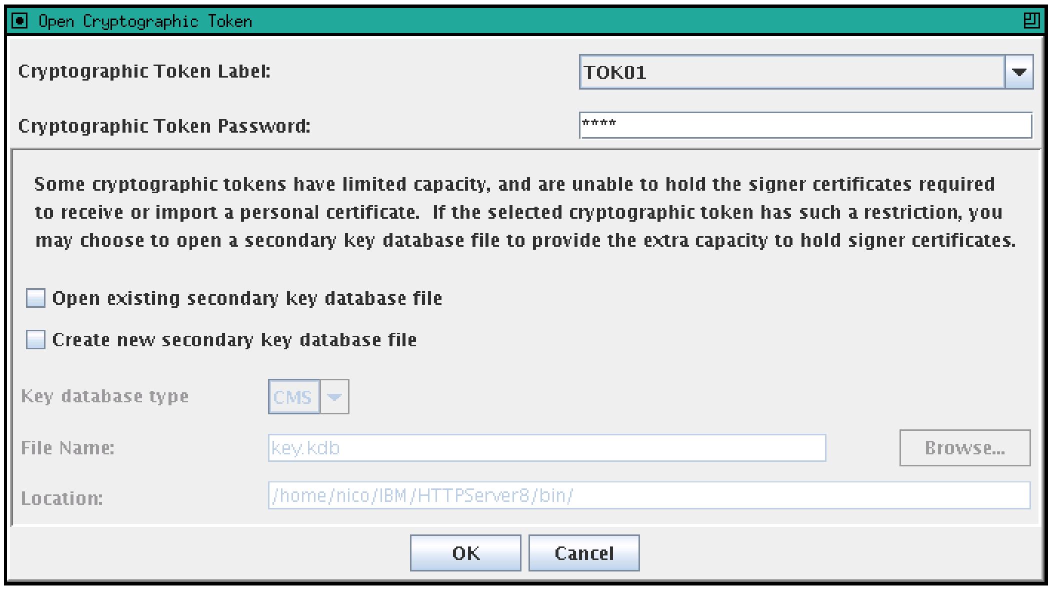
Key database type: CMS Cryptographic Token

File Name: PKCS11_API.so

Location: /usr/lib/pkcs11/

Figure 5-5 Opening a PKCS#11 key store with iKeyman

3. When opening a cryptographic token you are asked which token to access and provide the respective PIN. Make sure you clear both check boxes, as in Figure 5-6, to work only with the cryptographic token as a key store. Click **OK** to confirm your settings and open the token.



Open Cryptographic Token

Cryptographic Token Label: TOK01

Cryptographic Token Password: ****

Some cryptographic tokens have limited capacity, and are unable to hold the signer certificates required to receive or import a personal certificate. If the selected cryptographic token has such a restriction, you may choose to open a secondary key database file to provide the extra capacity to hold signer certificates.

☐ Open existing secondary key database file

☐ Create new secondary key database file

Key database type: CMS

File Name: key.kdb

Location: /home/nico/IBM/HTTPServer8/bin/

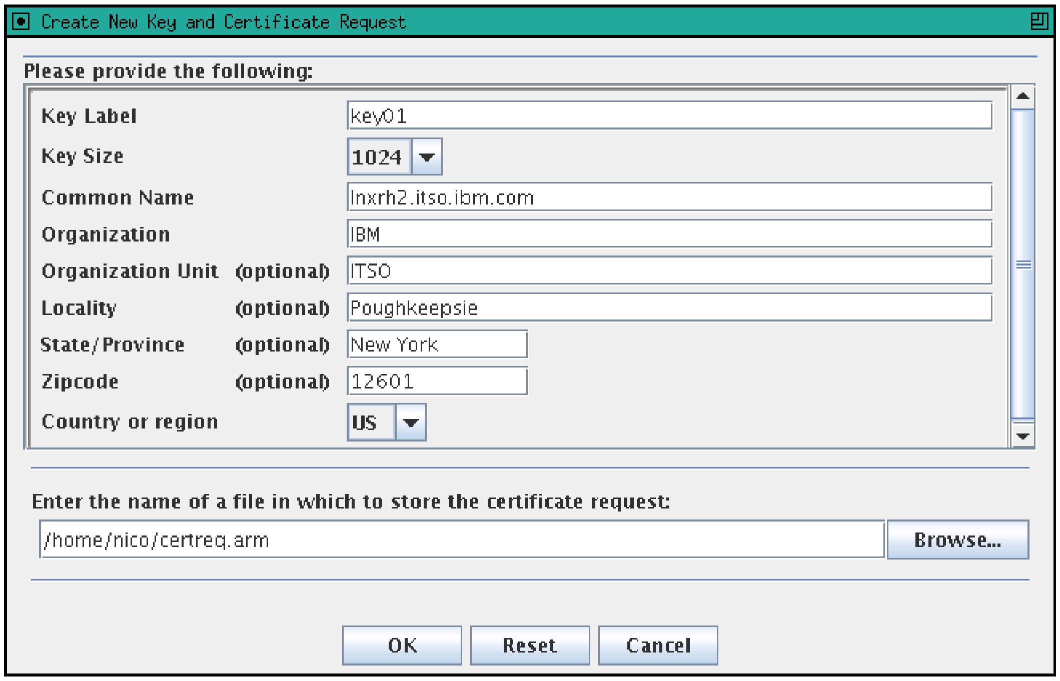
Figure 5-6 Options when opening PKCS#11 key store with iKeyman

4. If you did not create any keys previously, your iKeyman window appears almost exactly like the one in Figure 5-4 on page 171. To create a new key and certificate request:

   a. Select **Personal Certificate Request** from the drop-down list in the “Key database content” area.

   b. Click **New**. A new window like the one in Figure 5-7 on page 173 opens and you can enter properties for the new key and certificate request.

c. Click **OK** to create the request.


Create New Key and Certificate Request

Please provide the following:

Key Label		key01
Key Size		1024
Common Name		lnxrh2.itso.ibm.com
Organization		IBM
Organization Unit	(optional)	ITSO
Locality	(optional)	Poughkeepsie
State/Province	(optional)	New York
Zipcode	(optional)	12601
Country or region		US

Enter the name of a file in which to store the certificate request:

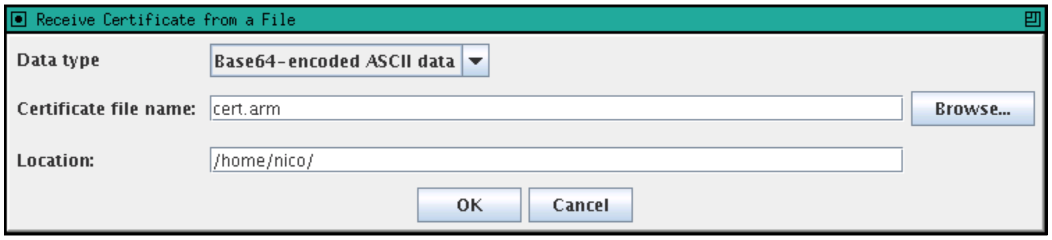
/home/nico/certreq.arm

Figure 5-7 Generating a new key signing request with iKeyman

The token now contains a new key and a certificate request has been saved to your file system. You can send this request off to a signing authority of your choice and receive a certificate.

To work with this certificate you have to import it into the token where the key is stored. You also have to import the whole chain of certificates of your signing authorities up to the root certificate authority. Make sure you have all certificates available in base 64 encoded DER format, preferably with the `.arm` file name extension.

5. Change back to the Personal Certificates view by selecting the corresponding entry from the drop-down list in the Key database content area and clicking **Receive**. The dialog shown in Figure 5-8 opens and you can select the certificate that was issued to you by the certificate authority. Click **OK** when you are done.

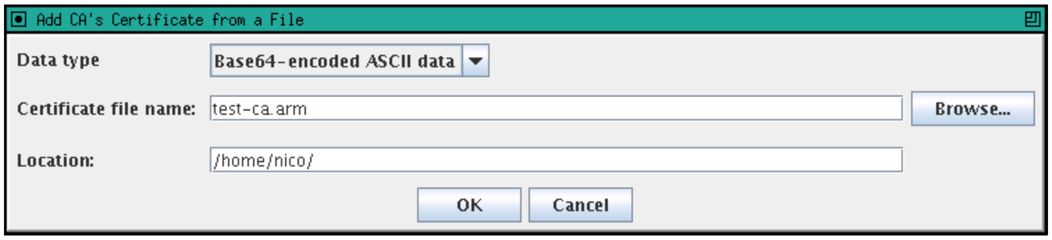

Receive Certificate from a File

Data type	Base64-encoded ASCII data
Certificate file name:	cert.arm
Location:	/home/nico/

Figure 5-8 Receive certificate into key store using iKeyman

You have now successfully received the certificate for your key into your PKCS#11 key store where it has automatically been associated with the matching key.

6. If you want to use this key-certificate pair with IBM HTTP Server, you also have to add the CA signing chain to the key store. Change to the Signer Certificates view and click **Add** to open the dialog shown in Figure 5-9 on page 174. Select your CA certificate name by using the **Browse** button and clicking **OK** to import.



Data type: Base64-encoded ASCII data
Certificate file name: test-ca.arm
Location: /home/nico/

Figure 5-9 Adding CA certificate to key store with iKeyman

After you imported all certificates you can now use the cryptographic token and your Crypto Express adapter with IBM HTTP Server. See "IBM HTTP Server" on page 178 for more information.

Keep in mind that GSKit is more than just iKeyman. It is a library that, on supported operating systems, offers you platform-independent access to cryptographic functionality and hardware, significantly reducing the effort that is required to migrate applications. For more information about GSKit, go to:

`http://www.ibm.com/developerworks/tivoli/library/t-gsk7/`

5.4 Securing communication and applications

Cryptography is not a self-serving functionality but is used to protect data, either in transport or statically on a storage medium. As such, it has to integrate with applications and communication between applications. This section has examples of how to add hardware-supported cryptography to selected sets of applications to benefit from the increased performance and security that is provided by CPACF and the Crypto Express feature.

OpenSSL

One of the most prominent cryptographic libraries in the open source environment is OpenSSL, a library that implements cryptographic and cryptography-related operations such as encryption, signing, and certificate functions. You can use the OpenSSL library in your own applications to reduce development effort, or if your company does not have the necessary cryptography knowledge.

The OpenSSL library has a plug-in mechanism that allows various *engines* to be used for cryptographic operations. One of these engines is the ibmca engine, usually located in:

`/usr/lib/openssl/engines/libibmca.so`

It uses CPACF and Crypto Express functionality if they are present in the system.

OpenSSL includes a utility called **`openssl`** that provides command-line access to major cryptographic operations of the OpenSSL library. Most commands support an option to select the encryption engine to use, which allows you to specify the ibmca engine to make use of the System z hardware acceleration.

Example 5-29 on page 175 shows how to query the engine for cryptographic operations that are supported by the ibmca library and a Crypto Express3 adapter in accelerator mode.

Example 5-29 Query ibmca engine for supported cryptographic operations

```
[nico@lnxrh2 ~]$ openssl engine ibmca -c
(ibmca) Ibmca hardware engine support
 [RSA, DSA, DH, RAND, DES-ECB, DES-CBC, DES-EDE3, DES-EDE3-CBC, AES-128-ECB,
AES-128-CBC, AES-192-ECB, AES-192-CBC, AES-256-ECB, AES-256-CBC, SHA1, SHA256]
```

To use the ibmca engine as an argument to an **openssl** command, specify the **-engine** option, as shown in Example 5-30, where it is used to generate a new RSA key with a key length of 2048 bits.

Example 5-30 Generating an RSA key using OpenSSL and the ibmca engine

```
[nico@lnxrh2 ~]$ openssl genrsa -engine ibmca -out lnxrh2.sserver.key 2048
engine "ibmca" set.
Generating RSA private key, 2048 bit long modulus
...+++
................+++
e is 65537 (0x10001)
```

For certain operations, such as certificate-request handling, an **-engine** option is not available because no cryptographic operation is being performed.

To show you more features of OpenSSL and how to use the ibmca engine, we must get a certificate for the key that was generated in Example 5-30. We set up a sample certificate authority in Example 5-31 which involves:

1. Creating an OpenSSL configuration file.
2. Initializing the index and serial number files.
3. Self-signing the key that has been generated in Example 5-30.

Example 5-31 Setting up a sample certificate authority with OpenSSL

```
[nico@lnxrh2 ~]$ mkdir /tmp/testca
[nico@lnxrh2 ~]$ echo '
[ ca ]
default_ca      = CA01

[ CA01 ]
dir             = /tmp/testca
certs           = $dir/certs
crl_dir         = $dir/crl
database        = $dir/index.txt
new_certs_dir   = $dir/newcerts
serial          = $dir/serial
crlnumber       = $dir/crlnumber
crl             = $dir/crl.pem
private_key     = $dir/private/cakey.pem
RANDFILE        = $dir/private/.rand
default_md      = sha1
policy          = pany
[ pany ]
commonName      = supplied
' > /tmp/testca/openssl.cnf
[nico@lnxrh2 ~]$ touch /tmp/testca/index.txt
[nico@lnxrh2 ~]$ mkdir /tmp/testca/newcerts
[nico@lnxrh2 ~]$ echo 00 > /tmp/testca/serial
```

```
[nico@lnxrh2 ~]$ openssl ca -engine ibmca -selfsign -days 365 \
-config /tmp/testca/openssl.cnf -keyfile lnxrh2.sserver.key \
-in lnxrh2.sserver.req -out lnxrh2.sserver.crt
Using configuration from /tmp/testca/openssl.cnf
engine "ibmca" set.
Check that the request matches the signature
Signature ok
[ ... ]
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

With the self-signed certificate in `lnxrh2.sserver.crt`, you may now use the **`s_server`** and **`s_client`** commands of **`openssl`** to create a simple secure TCP tunnel. Example 5-32 shows the steps involved. It queries the cryptographic driver statistics to verify that the CEX3A adapter is indeed being used in the SSL connection.

Example 5-32 Setting up a secure communication with openssl and the ibmca engine

```
[nico@lnxrh2 ~]$ openssl s_server -engine ibmca -key lnxrh2.sserver.key \
-cert lnxrh2.sserver.crt -accept 3344 -quiet &
[1] 6655
engine "ibmca" set.
[nico@lnxrh2 ~]$ jobs
[1]+  Running                 openssl s_server -engine ibmca -key lnxrh2.sserver.key -cert
lnxrh2.sserver.crt -accept 3344 -quiet &
[nico@lnxrh2 ~]$ cat /proc/driver/z90crypt  | grep 'open handles'
Total open handles: 1
[nico@lnxrh2 ~]$ cat /proc/driver/z90crypt | grep -A 8 'completed request counts'
Per-device successfully completed request counts
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000252 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[nico@lnxrh2 ~]$ openssl s_client -engine ibmca -CAfile lnxrh2.sserver.crt \
-connect localhost:3344 -quiet
engine "ibmca" set.
depth=0 /CN=lnxrh2-sserver
verify return:1
hello world
hello world
<press ctrl+c>

[nico@lnxrh2 ~]$ cat /proc/driver/z90crypt | grep -A 8 'completed request counts'
Per-device successfully completed request counts
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000258 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

You can use this setup, for example, to connect two programs that usually communicate through file system pipes. Another use of **`s_client`** is to enable scripts to perform HTTPS requests to secure servers such as Apache with mod_ssl.

Apache httpd and mod_ssl

The Apache Web server, also called httpd, provides encrypted communications through the HTTP over SSL and TLS through the use of additional modules. The most common module used for SSL encryption with the Apache Web server is mod_ssl, which uses the OpenSSL library for encryption. To use System z hardware cryptography support, the only configuration you have to change is for mod_ssl to use the ICA engine.

Note: You have to load the z90crypt module before you start or restart your Web server, otherwise you will not have Crypto Express support.

On SUSE Linux Enterprise Server 11, this step is done in `/etc/apache2/ssl-global.conf`, on Red Hat Enterprise Linux 5.4. If you have installed mod_ssl, the configuration is in the `/etc/httpd/conf.d/ssl.conf` file. In both cases, you have to add the statement from Example 5-33 and restart the Web server for the changes to take effect.

Example 5-33 Statement to set the mod_ssl crypto engine to use

```
SSLCryptoDevice ibmca
```

You can get the number of currently open handles to your Crypt Express adapter by reading the `/proc/driver/z90crypt procfs` entry. Enabling the ICA engine in the Web server should increase the number. See Example 5-34.

Example 5-34 Number of open handles to the Crypto Express adapter

```
nico@lnxsu2:~> cat /proc/driver/z90crypt  | grep 'open handles'
Total open handles: 1
```

You should also see an increasing number of completed requests in the driver status file for every new SSL connection. See Example 5-35.

Example 5-35 Statistics for total number of handled requests for the z90crypt driver

```
[nico@lnxrh2 ~]$ cat /proc/driver/z90crypt | grep -A 8 'completed request counts'
Per-device successfully completed request counts
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000217 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Keep in mind that simply clicking **Reload** in your Web browser might not work because the old SSL connection might be reused. As shown in Example 5-36 on page 178, you can emulate a browser request with the **openssl s_client** command. Make sure you are *not* using the ibmca engine, which would also cause the request count to increase.

Example 5-36 Using the s_client command of OpenSSL to make an HTTPS request

```
[nico@lnxrh2 ~]$ echo -en 'GET / HTTP/1.0\n\n' | \
openssl s_client -connect localhost:443
CONNECTED(00000003)
[...]
DONE
```

IBM HTTP Server

The IBM HTTP Server (IBM HTTP Server) offers you the advantage of using both the acceleration of the Crypto Express adapter and the PKCS#11 features to protect your private key. For a quick introduction of how to set up a PKCS#11 token with a Crypto Express adapter in accelerator mode, for use with IBM HTTP Server, see "GSKit" on page 170.

Every access to the cryptographic token requires you to authenticate yourself with a PIN. To have your Web server start automatically, you have to store the PIN in a *stash* by using the **sslstash** utility. Example 5-37 shows how to create a stash for PIN 1111 that will be used to access a cryptographic function.

Example 5-37 Sample command for creating a stash file holding the token PIN

```
/opt/IBM/HTTPServer/bin/sslstash -c /opt/IBM/HTTPServer/stash crypto 1111
```

The next step is configuring IBM HTTP Server to use the token for SSL encrypted communication. This also automatically enables usage of the Crypto Express adapter for processing and accelerating the cryptographic operations involved in the SSL encryption. Example 5-38 shows a sample configuration to make IBM HTTP Server use the following items:

- The key named key01 in the token TOK01
- The openCryptoki driver for PKCS#11 operations
- The stash file created earlier

You have to restart the Web server for the changes to take effect.

Example 5-38 Sample configuration for IBM HTTP Server using the PKCS#11 token with the Crypto Express adapter

```
Listen 443
<VirtualHost *:443>
    SSLEnable
    SSLServerCert TOK01:key01
    Keyfile /opt/IBM/HTTPServer/Plugins/etc/plugin-key.kdb
    SSLPKCSDriver /usr/lib/pkcs11/PKCS11_API.so
    SSLStashfile /opt/IBM/HTTPServer/stash
</VirtualHost>
```

If the configuration is correct and the access to the Crypto Express card has been established, you can see at least one additional open handle in the status listing of the z90crypt driver. You can also see an increasing number of handles by using the command found in Example 5-34 on page 177.

Java

The hardware cryptography extensions provided by both CPACF and the Crypt Express adapters are also available to you in the Java programming environment, enabling you to build and run applications with increased performance. Access to the hardware is mediated by the IBM Java PKCS#11 implementation (IBMPKCS11Impl) library that is, for example, provided in SUSE Linux Enterprise Server 11 with the IBM Java Runtime Environment (JRE).

This section provides a short introduction. For an extensive reference of IBM Java PKCS#11 implementation, go to:

http://www.ibm.com/developerworks/java/jdk/security/50/secguides/pkcs11implDocs/IBMJavaPKCS11ImplementationProvider.html

Your environment must meet the following prerequisites:

- A PKCS#11 key store has already been set up by using the iKeyman utility from the GSKit. See "GSKit" on page 170 for more information.
- A token, named TOK01, is in slot 0 and contains a key named key01.
- The IBM Java Developer Kit for System z is installed and configured correctly.

The first hardware-supported function we access is the random number generator. Example 5-39 is an example class that first loads and initializes the PKCS#11 provider and then uses an instance of the provider hardware random number generator to create a kilobyte of random data.

Because this class is reused in further examples, be sure it is available even if you do not plan to explicitly compile Example 5-39.

Example 5-39 PKCS11Test.java: Setting up IBMPKCS11Impl and accessing the hardware random number generator

```
import java.security.*;
import com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl;

public class PKCS11Test {
    static {
        try{
            IBMPKCS11Impl provider = new IBMPKCS11Impl();
            /*
             * Initialize the provider with
             *    - the native library
             *    - the slot
             *    - the user PIN for the slot
             */
            provider.Init(  "/usr/lib/pkcs11/PKCS11_API.so64:0",
                            "1111".toCharArray() );
            // Add the provider to the JVM's interal list of providers
            Security.addProvider(provider);
        }
        catch(Exception e){
            throw(new RuntimeException(e));
        }
    }

    public static void main(String[] args) throws Exception {
         // Get a new randum number generator instance.
        SecureRandom rnd = SecureRandom.getInstance("PKCS11DeviceRNG");
        // Buffer to hold the new random data.
        byte[] buffer = new byte[1024];
        // Read random data.
        rnd.nextBytes(buffer);
        // Print last random byte.
        System.out.println(0xFFl & buffer[1023]);
    }

}
```

Compiling and running the PKCS11Test class is performed in Example 5-40. The JVM specification mandates array values to be initialized to 0, so the changing output indicates that the array is really filled with random numbers.

Example 5-40 Compiling and running the random number test class

```
nico@lnxsu2:~/java> javac -sourcepath ./ PKCS11Test.java
nico@lnxsu2:~/java> java -cp ./ PKCS11Test
249
nico@lnxsu2:~/java> java -cp ./ PKCS11Test
132
nico@lnxsu2:~/java> java -cp ./ PKCS11Test
49
```

The IBMPKCS11Impl also provides access to the keys that are stored in the cryptographic token by using the standard Java KeyStore interface. This step makes migrating your applications as easy as changing the key store name and key label. Example 5-41 shows an example of how you can access the private key stored in the token and use it to sign data that is being read in from the system.

Example 5-41 PKCS11Test1.java - Signing data with a private key from the PKCS#11 key store in Java

```
import java.security.*;
import javax.crypto.*;
import java.io.*;

public class PKCS11Test1 extends PKCS11Test {
    public static void main(String[] args) throws Exception {
        KeyStore keystore = KeyStore.getInstance("PKCS11IMPLKS");
        keystore.load(null,null);
        Key key = keystore.getKey("key01",null);
        Signature signature = Signature.getInstance("SHA1withRSA");
        signature.initSign((PrivateKey)key);
        System.out.println("Enter data to sign. End with EOF (ctrl+d):");
        int in = 0;
        while((in = System.in.read()) != -1){
            signature.update((byte)(in & 0xFF));
        }
        System.out.println("Generating signature:");
        byte[] sig = signature.sign();
        for(int i=0;sig != null && i<sig.length;i++){
            System.out.print(Integer.toHexString((sig[i] & 0xFF))+":");
        }
        System.out.println();
    }
}
```

Compile and execute the test class. The result is a 256-byte signature that is created using the 2048-bit RSA key and SHA1 hashing algorithm.

Example 5-42 Compiling and running the data signing test class

```
nico@lnxsu2:~/java> javac -sourcepath ./ PKCS11Test1.java
nico@lnxsu2:~/java> java -cp ./ PKCS11Test1
Enter data to sign. End with EOF (ctrl+d):
test123
Generating signature:
64:8a:6a:fb:be:b5:a5:53:f3:c:fb:f5:72:3:d8:c9:f3:2b:62:a8:14:1c:71:23:44:9a:53:13:
77:58:c9:27:d3:73:31:4:9:85:23:e8:69:6:18:d0:4b:e8:81:d1:a6:76:df:3d:27:a3:2c:bf:2
6:a5:ac:ff:da:f1:cc:9e:a8:42:47:ee:c5:a1:20:99:25:eb:ad:e2:5a:12:35:8c:52:cb:19:55
:a8:56:5c:91:9c:60:ae:da:dc:d0:9f:4:2c:7b:e2:17:54:3d:63:11:6f:10:41:9b:87:5:16:f6
:e9:13:72:f1:9c:5c:25:d1:d:c9:8a:be:6e:62:8e:3d:bc:28:9c:87:c2:16:47:9b:bc:a:f7:43
:3c:14:fc:6b:93:c5:a:7c:be:70:c2:67:32:9b:83:cc:35:ad:67:e1:36:6c:90:cd:f3:ba:bb:c
4:cf:ce:30:5a:f4:d1:c9:5f:d0:7:2e:e3:d8:d:78:e6:49:e5:e6:fc:a5:a1:f0:b9:7d:4a:9c:4
a:c7:4e:4b:b2:8f:41:49:35:80:9d:f8:24:89:2b:b5:c2:c7:8d:85:c6:ae:69:a5:c7:ed:3d:b3
:29:bd:58:12:30:9f:cf:17:a1:24:5e:85:3:1b:b2:1c:8e:17:88:81:e8:c7:22:b:55:73:3e:98
:8d:fe:3:f3:6e:
```

As with all cryptographic operations that take advantage of the Crypto Express feature, you can determine whether the hardware support is being used by the increasing number of successfully completed requests in the `/proc/drivers/z90crypt` status file.

5.5 Statistics and performance

Cryptographic operations are used to protect data. However, the main reason for running applications is to produce the data that is being encrypted or signed. That is why companies want their applications to spend as much time as possible on processing the data instead of cryptography. This is the incentive for using dedicated hardware to run computing-intensive algorithms, such as RSA encryption, freeing up cycles on the general purpose processor, which improves application performance and can also lower costs.

Thomas Weber and the IBM Lab in Böblingen measured the performance of CPACF and Crypto Express3. You can view the complete results online, but we present some of their key findings in this book to give you an overview of what you can expect from the hardware support for cryptography on System z.

All measurements were taken on a System z10 machine. If you are using System z9, certain CPACF ciphers are not available to you. See 5.3, “Cryptographic APIs” on page 164 for the list of algorithms that are supported by CPACF. As of writing, there are no performance measurements for the Crypto Express4 adapters that are running with Linux on System z.

Figure 5-10 on page 182 shows the processor cost savings that can be achieved when using the ICA OpenSSL engine, which offloads certain ciphers to the CPACF (see “OpenSSL” on page 174 for more information about using the ICA engine). Especially interesting is the almost 12-times-faster computation of the triple DES (TDES) algorithm encryption and SHA hashing combination. The most prominent use of TDES is for encrypting payment card PINs, which is mandatory (as of 2010) for complying with the Payment Card Industry Data Security Standard (PCI DSS).

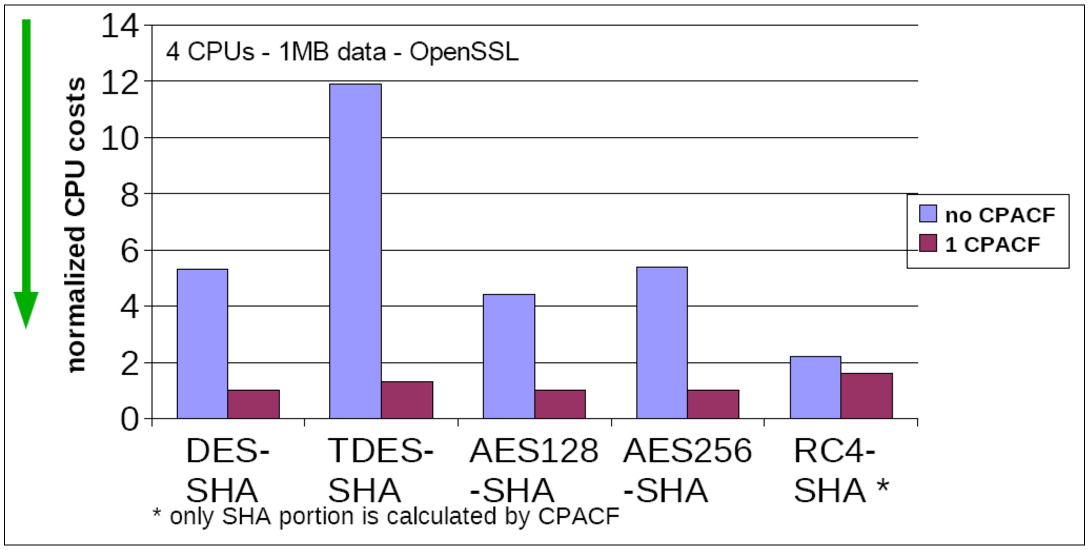


Figure 5-10 Performance measurements for CPACF

As described in “Apache httpd and mod_ssl” on page 177, the Crypto Express adapters in accelerator mode can be used to speed up SSL handshakes of web servers. Figure 5-11 is a performance comparison between a web server that is using standard SSL and one using CEX2 support. Obviously, offloading the encryption involved in the handshakes provides huge savings in processor time and the number of connections that can be processed per second.

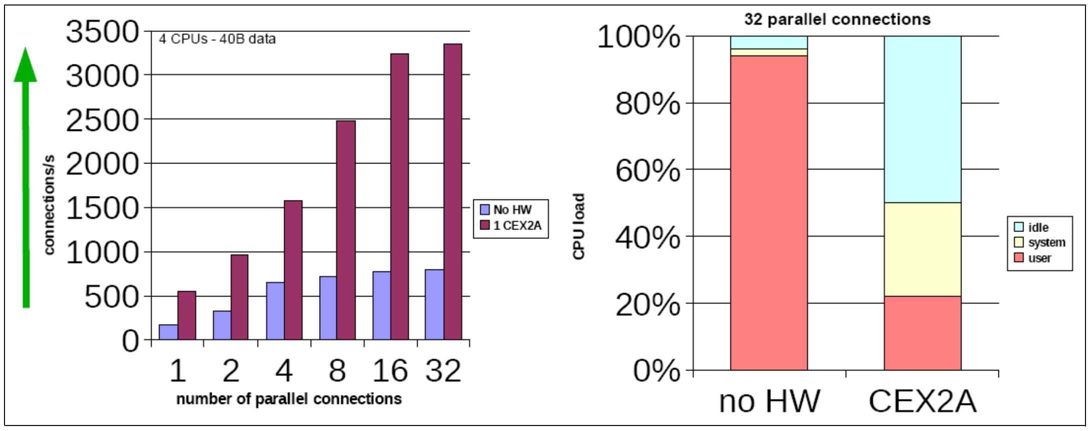


Figure 5-11 Performance measurements for CEX3A

The final step in improving your web server’s performance is to combine CPACF with CEX3A. Figure 5-12 on page 183 shows that you can practically double your throughput while reducing your processor costs to up to a sixth when compared to running without hardware cryptography support.

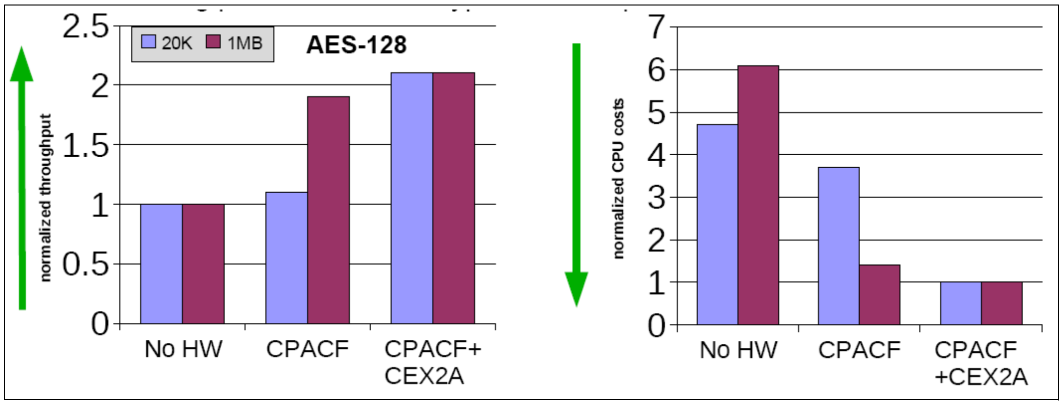


Figure 5-12 Performance comparison of full System z hardware supported cryptography

One of the important aspects to keep in mind when looking at these performance charts is that, on System z, you can leverage the results by using the virtualization techniques that are provided by LPAR and z/VM. Hundreds of virtual machines can access the cryptographic hardware, enabling you to balance the load and make more efficient use of your resources.

5.6 Secure Key Crypto

The System z Crypto Express2 (FC0863 or FC0870) and Crypto Express3 (FC0864 or FC0871) features provide processing offload support for a variety of cryptographic operations. Each Crypto Express feature contains one or two adapters[1]. Each adapter can be configured in *accelerator mode* or *coprocessor mode*. When operating in coprocessor mode (referred to as CEX2C) the adapter provides support for secure key as well as clear key cryptographic operations. If you do not need secure key operations, however, you should consider using the adapter in accelerator mode for improved performance.

5.6.1 Comparing secure key, clear key, and protected key operations

From the perspective of the actual cryptographic operation, no difference exists between clear key and secure key operations. A piece of original plaintext, encrypted by using the same algorithm and the same key, can produce the same ciphertext whether the key was clear or secure.

The difference between clear key and secure key is simply in the management of the keys. With clear key, the key is protected only by file system permissions. In other words, a sufficiently privileged user can locate where the key is stored and read it. A secure key is encrypted by using a different key (the master key) and is never visible in the clear outside the secure cryptographic hardware.

However, an enhancement to CPACF facilitates the continued privacy of cryptographic key material when used for data encryption. CPACF, using key wrapping, insures that key material is not visible to applications or operating systems during encryption operations.

IBM hardware that supports secure key operation provides protection for secure keys by employing tamper-sensitive storage that zeros the memory of the device if it is attacked. This protects the keys even when they are being used inside the hardware. The hardware can

[1] The features containing one adapter (FC0870 and FC0871) can be used only on the System z10 BC server and cannot be carried forward if the z10 BC is later to be upgraded to a z10 EC.

even support the changing of the master key, by decrypting the secure key and re-encrypting it using the new master key within the secure cryptographic hardware.

5.6.2 Secure key functions

In this section we discuss secure key functions.

FIPS compliance

The United States government has introduced a set of standards that define cryptographic algorithms and procedures to be used by government agencies and companies that work for the U.S. government. These standards are part of the Federal Information Processing Standard (FIPS) and include, for example, the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES) encryption algorithms.

The Crypto Express adapters for System z are designed and certified to work in a standard-compliant mode, freeing application programmers from dealing with the intricacies of these standards. In secure key mode, you do not have to configure the adapter especially for FIPS compliance.

RSA public and private key processing

Running in clear key mode, the Crypto Express adapter supports RSA encryption and decryption with key lengths of up to 2048 bits. In secure mode, this functionality is extended to key lengths up to 4096 bits. For an example of how to use an RSA key in Java to sign a message, see Example 5-41 on page 180.

DES and triple DES

DES and triple DES are common shared key encryption algorithms that are used in many applications today. Examples include smart cards, SSL communication, and disk encryption. The Crypto Express adapters in coprocessor mode provide an implementation of these algorithms, just as CPACF does. The difference lies in the asynchronous processing that is performed with the Crypto Express adapter.

Although using CPACF will block your PU from any other work until the cryptographic operation is completed, Crypto Express works asynchronously. Requests are queued and pushed to the adapter where they are processed while the PU is free to execute other code. Depending on your setup, the z90crypt driver then either polls the adapter for completed requests or is notified by the adapter itself, and the processed data is handed back to your application.

MAC processing

In cryptography a distinction exists between message confidentiality, which is provided by encryption, and message integrity, which can be provided either by signing the message or adding a message authentication code (MAC) to it.

Signatures use asymmetric keys to provide an advanced level of integrity and non-repudiability (ensuring the sender cannot deny sending a message) whereas message authentication codes use symmetric keys and thus only provide basic integrity verification.

In secure key mode, the Crypto Express adapter provides MAC functionality, which allows you to offload both MAC creation and MAC verification to the coprocessor. The CCA RPM package contains sample source code in `/opt/IBM/4764/samples/mac.c` for computing a MAC using the Crypto Express adapter.

5.7 Set up of CPACF, Crypto Express3, and Crypto Express4

The System z hardware platform provides two distinct features that operating systems can exploit to improve the performance of cryptographic operations or secure their cryptographic key handling:

- One is the Central Processor Assist for Cryptographic Function (CPACF), a set of special instructions that are available to processing units (PUs) on System z. On System z10 CPACF specifically provides instructions for performing DES, triple DES, and AES up to 256-bit key size encryption and decryption. It also performs SHA1, SHA224, SHA256, SHA384, SHA512 hashing, and can generate pseudorandom numbers. The new instructions are available in supervisor state and problem state, and do not require a special driver to be accessed.

  You might have to recompile your programs and libraries to include the new machine instructions though, and if your compiler does not support emitting code for the new instructions that are provided by CPACF, you must implement the access to CPACF yourself. For more information about the additional instructions provided by CPACF see "Message-Security Assist" in *z/Architecture Principles of Operation*, SA22-7832.

- The other hardware options that help your applications perform cryptographic operations faster and more secure are the Crypto Express3 (CEX2) and Crypto Express4 (CEX3) features. Both features contain at least one adapter that can be used to offload cryptographic operations from the PU. Every adapter can be programmed independently to run in one of two modes: Accelerator mode and coprocessor mode. In accelerator mode you have access to RSA encryption and signature verification algorithms with a key length of up to 2048 bits, which is designed to speed up SSL and TLS handshakes.

  In accelerator mode only clear key cryptography is supported. In coprocessor mode the card can additionally perform DES and triple DES operations, use RSA key sizes of up to 4096 bits, and act as a full cryptographic hardware token, providing full secure key processing. For a quick introduction to setting up a Crypto Express adapter, see "Configuring Crypto Express support for z/VM users" on page 186.

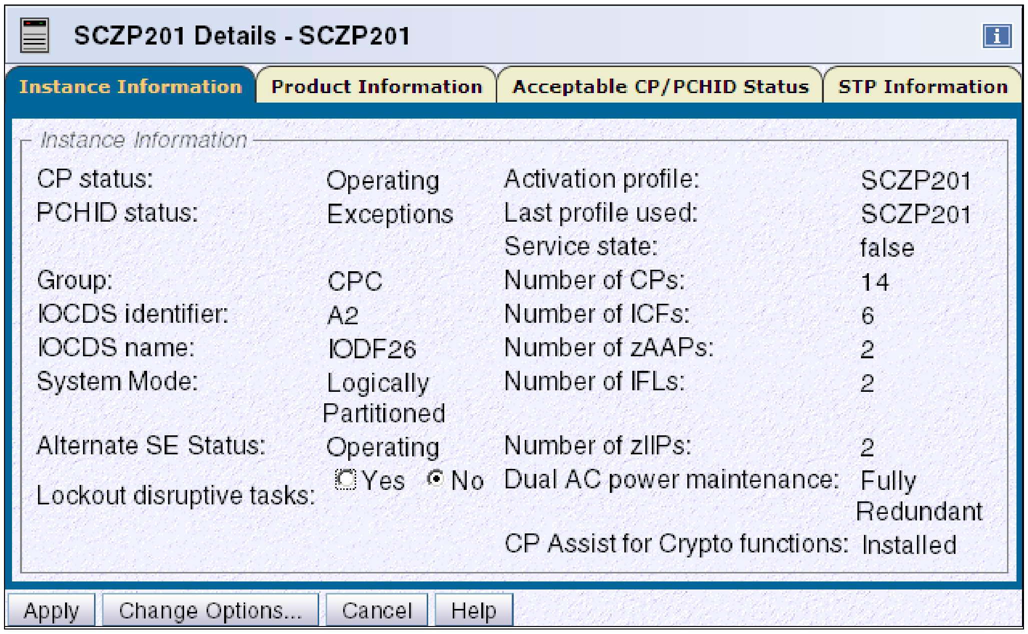
When planning your cryptographic capacity, keep in mind that, although the Crypto Express cards work asynchronously, freeing up the PU to do other work, CPACF is a synchronous facility, and cryptographic operations on it will block the PU until they are done. Furthermore, you must take into account that one CPACF is shared between two PUs and can therefore be subject to congestion that is caused by another z/VM user utilizing CPACF from another PU.

5.7.1 Toleration mode versus exploitation mode

Enabling CPACF

CPACF is a no-charge enablement feature on System z10 hardware. If you plan to enable CPACF, consider that it can only be enabled or disabled on a per central processor complex (CPC) basis and will affect all logical partitions (LPAR). The CPACF feature code is 3863. For information about how to enable a feature code see *System z10 Support Element Operations Guide*, SC28-6979.

To check whether CPACF is enabled for your CPC, open the CPC details in the Support Element interface and check the value of "CP Assist for Crypto functions" in the lower right corner of the dialog. If the value indicates "Installed" as in Figure 5-13 on page 186, CPACF is enabled. If not, you must enable it.


SCZP201 Details - SCZP201

Instance Information

CP status:	Operating	Activation profile:	SCZP201
PCHID status:	Exceptions	Last profile used:	SCZP201
		Service state:	false
Group:	CPC	Number of CPs:	14
IOCDS identifier:	A2	Number of ICFs:	6
IOCDS name:	IODF26	Number of zAAPs:	2
System Mode:	Logically Partitioned	Number of IFLs:	2
Alternate SE Status:	Operating	Number of zIIPs:	2
Lockout disruptive tasks:	Yes [✓ No]	Dual AC power maintenance:	Fully Redundant
		CP Assist for Crypto functions:	Installed

Figure 5-13 CPC details dialog in Support Element interface

After CPACF has been enabled, you can use the icainfo tool from the libica package on Linux to verify that your Linux guest has access to the hardware-supported cryptography functions. If the output does not look like Example 5-43, but you still have a few algorithms marked with `yes`, you are probably not running on a System z10 machine.

Example 5-43 The icainfo output with CPACF enabled

```
[nico@lnxrh2 ~]$ icainfo
The following CP Assist for Cryptographic Function (CPACF) operations are
supported by libica on this system:
SHA-1:     yes
SHA-256:   yes
SHA-512:   yes
DES:       yes
TDES-128:  yes
TDES-192:  yes
AES-128:   yes
AES-192:   yes
AES-256:   yes
PRNG:      yes
```

Configuring Crypto Express support for z/VM users

Note: To enable the Crypto Express3 or Crypto Express4 feature you have to enable CPACF first. See "Enabling CPACF" on page 185 for more information.

Every Crypto Express3 and Crypto Express4 feature contains one or more adapters, each of which can be configured in either accelerator or coprocessor mode. In accelerator mode, some features of the coprocessor mode are not available but the adapter will perform the remaining functions such as RSA and DSA operations at much higher speed.

The Crypto Express cards introduce a concept of cryptographic domains. Each domain is protected by a master key preventing access across domains and effectively separating the contained keys. A cryptographic domain can span multiple Crypto Express adapters, creating a pool of resources that can be used to operate on the keys contained in the domain. Cryptographic domains only have meaning for Crypto Express adapters in coprocessor mode, which is discussed in 5.6, “Secure Key Crypto” on page 183.

You can also find a white paper describing the setup in the additional materials for this publication in Appendix B, “Additional material” on page 319.

Support Element

Before you can use your Crypto Express adapter in accelerator mode, you have to configure it by using the Support Element.

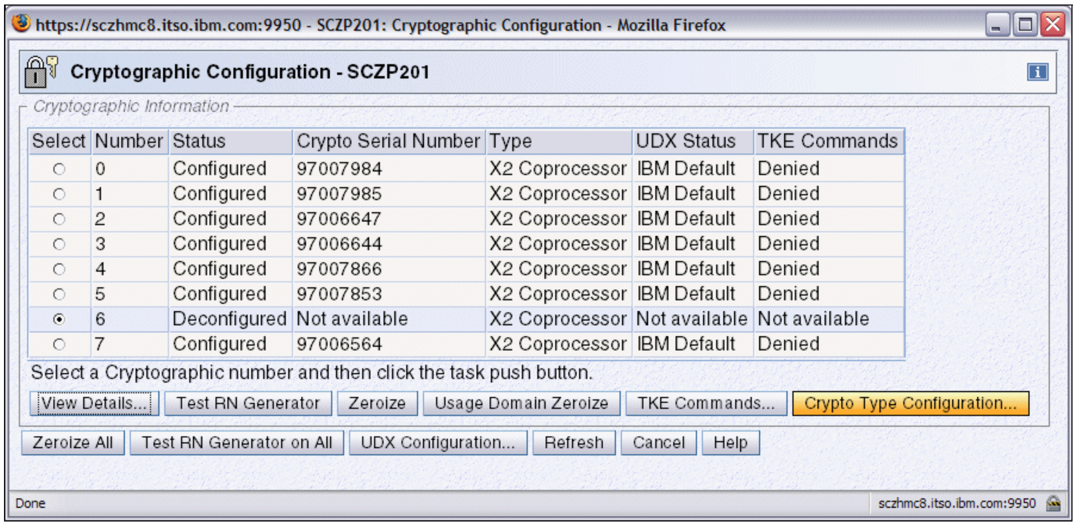
Note: The panel captures in this section have not been taken on the authors’ test system but are copies of the images that can be found in *IBM System z10 Enterprise Class Configuration Setup*, SG24-7571, therefore the names of devices and systems are not consistent with the following sections but still show the steps that are involved.

To configure the adapter:

1. In the CPC list, select the CPC that holds the cryptographic adapters that you want to reconfigure and select the **Cryptographic Configuration** task from the context menu. A new dialog, like the one in Figure 5-14, opens and lists the Crypto Express adapters that are installed.

   Make sure the cryptographic adapter you want to change is not used by any LPAR or you will not be able to select and reconfigure it.

2. Select the adapter you want to reconfigure and click **Crypto Type Configuration**.

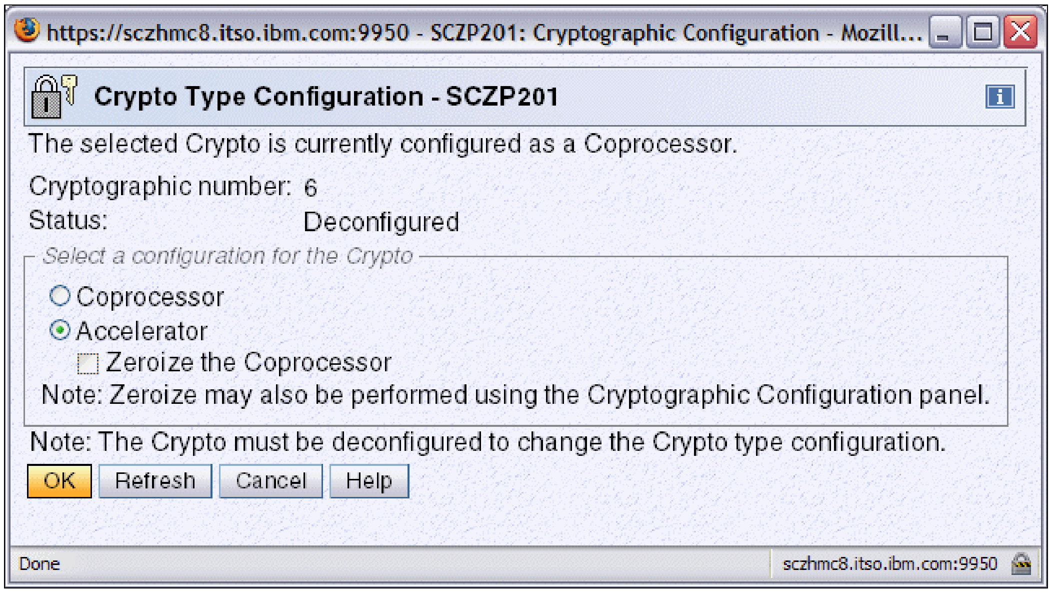

https://sczhmc8.itso.ibm.com:9950 - SCZP201: Cryptographic Configuration - Mozilla Firefox
Cryptographic Configuration - SCZP201
Cryptographic Information

Select	Number	Status	Crypto Serial Number	Type	UDX Status	TKE Commands
○	0	Configured	97007984	X2 Coprocessor	IBM Default	Denied
○	1	Configured	97007985	X2 Coprocessor	IBM Default	Denied
○	2	Configured	97006647	X2 Coprocessor	IBM Default	Denied
○	3	Configured	97006644	X2 Coprocessor	IBM Default	Denied
○	4	Configured	97007866	X2 Coprocessor	IBM Default	Denied
○	5	Configured	97007853	X2 Coprocessor	IBM Default	Denied
●	6	Deconfigured	Not available	X2 Coprocessor	Not available	Not available
○	7	Configured	97006564	X2 Coprocessor	IBM Default	Denied

Select a Cryptographic number and then click the task push button.

Figure 5-14 List of cryptographic adapters installed in CPC

3. In the dialog shown in Figure 5-15 on page 188, you can select the mode of operation that you want to set for the Crypto Express adapter. Choose **Accelerator** and click **OK**.


https://sczhmc8.itso.ibm.com:9950 - SCZP201: Cryptographic Configuration - Mozill...

Crypto Type Configuration - SCZP201

The selected Crypto is currently configured as a Coprocessor.

Cryptographic number: 6

Status: Deconfigured

Select a configuration for the Crypto

○ Coprocessor

[✓ ● Accelerator]

☐ Zeroize the Coprocessor

Note: Zeroize may also be performed using the Cryptographic Configuration panel.

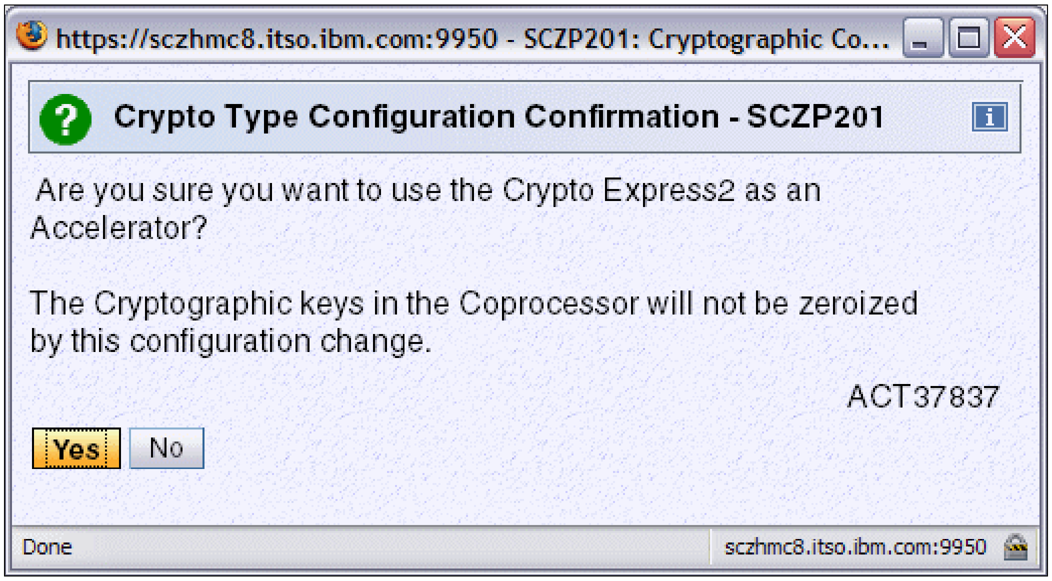
Note: The Crypto must be deconfigured to change the Crypto type configuration.

Figure 5-15 Crypto adapter reconfiguration dialog

4. The dialog in Figure 5-16 opens.

> **Note:** The word *zeroized* in this context means that all information such as keys and certificates are erased from the adapter. Make sure this is what you want before you click **Yes** in this step.

Confirm your selection by clicking **Yes**.


https://sczhmc8.itso.ibm.com:9950 - SCZP201: Cryptographic Co...

Crypto Type Configuration Confirmation - SCZP201

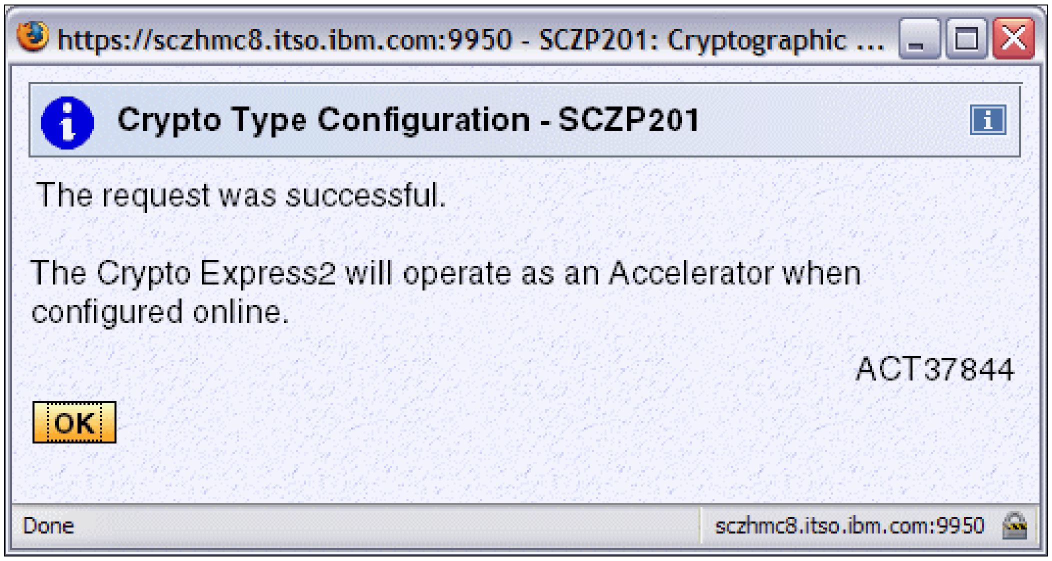
Are you sure you want to use the Crypto Express2 as an Accelerator?

The Cryptographic keys in the Coprocessor will not be zeroized by this configuration change.

ACT37837

Figure 5-16 Confirmation dialog for zeroizing the cryptographic adapter

When the adapter has been reconfigured and is ready for use the Support Element presents you with the message seen in Figure 5-17 on page 189.


https://sczhmc8.itso.ibm.com:9950 - SCZP201: Cryptographic ...

Crypto Type Configuration - SCZP201

The request was successful.

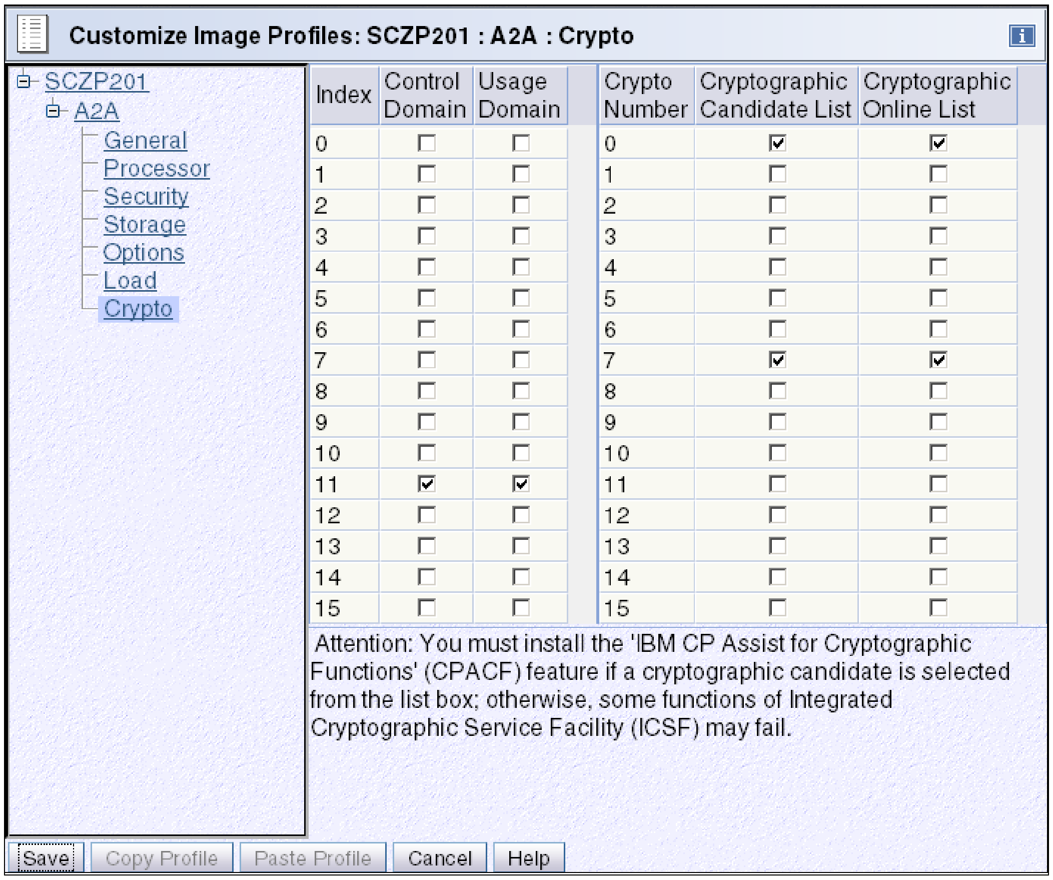
The Crypto Express2 will operate as an Accelerator when configured online.

ACT37844

Figure 5-17 Message dialog that confirms that the adapter is now in accelerator mode

After you have set up the Crypto Express adapter in the Support Element, you must allow access to it from your LPAR. You do this by using the Hardware Management Console (HMC). In Figure 5-18 on page 190, we have defined LPAR A2A to use and control the cryptographic domain number 11. It is also allowed to access the crypto adapters, numbers 0 and 7, and they will be brought online if they are present in the system. Again, for a more detailed discussion of planning the cryptographic configuration, see *IBM System z10 Enterprise Class Configuration Setup*, SG24-7571.

As you can see, there is no option to define the mode of the Crypto Express adapter on the LPAR level. You have to set it up in the Support Element. See “Support Element” on page 187 for more information about setting up the adapters.


Customize Image Profiles: SCZP201 : A2A : Crypto

Index	Control Domain	Usage Domain	Crypto Number	Cryptographic Candidate List	Cryptographic Online List
0	☐	☐	0	☑	☑
1	☐	☐	1	☐	☐
2	☐	☐	2	☐	☐
3	☐	☐	3	☐	☐
4	☐	☐	4	☐	☐
5	☐	☐	5	☐	☐
6	☐	☐	6	☐	☐
7	☐	☐	7	☑	☑
8	☐	☐	8	☐	☐
9	☐	☐	9	☐	☐
10	☐	☐	10	☐	☐
11	☑	☑	11	☐	☐
12	☐	☐	12	☐	☐
13	☐	☐	13	☐	☐
14	☐	☐	14	☐	☐
15	☐	☐	15	☐	☐

Attention: You must install the 'IBM CP Assist for Cryptographic Functions' (CPACF) feature if a cryptographic candidate is selected from the list box; otherwise, some functions of Integrated Cryptographic Service Facility (ICSF) may fail.

Figure 5-18 Cryptographic configuration for A2A LPAR

z/VM user directories

After you enabled and configured the adapter for your z/VM logical partition (LPAR) you have to grant your z/VM users access to the adapter. This is done with the CRYPTO statement in the user directory.

Example 5-44 shows a user directory that defines a shared virtual Adjunct Processor (AP). You can also define dedicated APs that are not shared among users. See “Cryptography on z/VM” in *Security on z/VM*, SG24-7471 for more information about the cryptography settings that are available. CRYPTO statements have to be defined directly after the USER statement or, if an INCLUDE statement is present, after the INCLUDE statement. For a detailed description of the CRYPTO statement and its options see *zVM: CP Planning and Administration*, SC24-6083.

Example 5-44 User directory with the CRYPTO APVIRT statement

```
USER LNXRH2 ****** 512M 1G G
   INCLUDE IBMDFLT
   CRYPTO APVIRT
   IPL CMS PARM AUTOCR
   MACHINE ESA 2
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH1
   NICDEF C300 TYPE QDIO LAN SYSTEM VSWITCH2
   MDISK 0191 3390 0241 0080 LX9U1R MR
   MDISK 0201 3390 0001 1000 LXC40B MR
   MDISK 0202 3390 1001 9016 LXC40B MR
```

Important: If you make major changes to a user directory's dedicated or shared cryptographic adapter settings and any of your users are no longer able to access their defined cryptographic adapters, you might have to re-IPL the z/VM system itself. For more information, see usage note number 9 in the "CRYPTO Directory Control Statement" section of *zVM: CP Planning and Administration*, SC24-6083.

The configuration we present in this chapter includes two adapters: one is in accelerator mode and the other is in coprocessor mode.

Note: Defining more than one adapter to a z/VM LPAR and having both coprocessor and accelerator modes present can result in the accelerator mode taking precedence over the coprocessor mode (if the adapters that are in coprocessor mode are not dedicated to specific users with the CRYPTO APDEDICATED directory control statement).

To check, from CP, whether your LPAR definition includes both a CEX2C and a CEX3A configuration, issue a QUERY CRYPTO AP statement. If the response looks like the one in Example 5-45, you will not be able to use the defined adapter in coprocessor mode unless you dedicate it to a user. The third line of Example 5-45 shows the output of a QUERY CRYPTO AP statement if a CEX3A adapter and a non-dedicated CEX2C have been defined. CP informs you that the coprocessor mode will not be available because a shared accelerator adapter has been defined too.

Example 5-45 QUERY CRYPTO AP output with one CEX2C and one CEX3A adapter

```
QUERY CRYPTO AP
AP 00 CEX2A Queue 11 is superseded by CEX2A
AP 07 CEX2C Queue 11 is superseded by CEX2A
```

After you change the user directory to dedicate the CEX2C adapter to a user, the output of QUERY CRYPTO AP indicates that the adapter can now be used in coprocessor mode, as shown in Example 5-46. Also, be sure to read the previous *Important* note (on page 191) about changing CRYPTO directory control statements.

Example 5-46 QUERY CRYPTO AP output with a dedicated CEX2C adapter

```
QUERY CRYPTO AP
AP 00 CEX2A Queue 11 is installed
AP 07 CEX2C Queue 11 is dedicated to LNXRH2
```

Linux

Access from Linux to the Crypto Express adapter is provided by the z90crypt Linux kernel module. Use the **lsmod** command to determine whether the z90crypt module is loaded. If not, use **modprobe** to load it and check the `/dev` file system to verify that the z90crypt device was created successfully. See Example 5-47.

Example 5-47 Check for z90crypt module and load it using modprobe

```
[nico@lnxrh2 ~]$ lsmod | grep z90crypt || echo Module not loaded
Module not loaded
[nico@lnxrh2 ~]$ sudo modprobe z90crypt
[nico@lnxrh2 ~]$ lsmod | grep z90crypt || echo Module not loaded
z90crypt             106668  0
[nico@lnxrh2 ~]$ ls -alF /dev/z90crypt
crw-rw-rw- 1 root root 10, 60 Sep 14 09:44 /dev/z90crypt
```

If you get an error message, like the one in Example 5-48, the CRYPTO APVIRT statement is probably not defined in the user directory and Linux has no access to any Crypto Express adapter.

Example 5-48 The modprobe output if no cryptographic device is defined for the virtual machine

```
nico@lnxsu2:~> sudo modprobe z90crypt
FATAL: Error inserting z90crypt
(/lib/modules/2.6.27.19-5-default/kernel/drivers/s390/crypto/z90crypt.ko): No such
device
nico@lnxsu2:~> dmesg | tail -n 1
ap.3677f7: The hardware system does not support AP instructions
```

To verify that a virtual cryptographic device has been defined for your z/VM user, issue the QUERY VIRTUAL CRYPTO through the vmcp interface on Linux. The output is similar to that in Example 5-49.

Example 5-49 Listing the virtual cryptographic devices through the vmcp interface

```
[nico@lnxrh2 mnt]$ sudo vmcp QUERY VIRTUAL CRYPTO
AP 47 CEX2A Queue 06 shared
```

Successfully loading the z90crypt driver also creates a new file in the `procfs` directory under `/proc/driver/z90crypt`. The contents of the file provide statistical information about the driver and the underlying (virtual) hardware. Example 5-50 shows the contents for a z/VM Linux user that has access to one CEX3A adapter that currently has no outstanding work, and has not yet completed any requests.

Example 5-50 Sample content of the /proc/driver/z90crypt driver statistics file

```
[nico@lnxrh2 ~]$ cat /proc/driver/z90crypt

zcrypt version: 2.1.1
Cryptographic domain: 6
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 0


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A
           0000000000000000 0000000000000000 0000000000000006 0000000000000000


Waiting work element counts
           0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

6

Physical and infrastructure security on System z

In this chapter we provide a discussion of how System z can be installed and configured to provide a secure infrastructure environment. Physical as well as logical security issues are discussed.

6.1 Physical environment

Infrastructure security begins with the physical environment. Buildings, rooms, infrastructure services, servers, access points, and operational equipment all play a part in a secure environment. The first element to consider is the security and integrity of the physical locations where the systems are installed and from which they can be accessed. Any security procedure or process can be bypassed if an attacker gains access to the system console. At a minimum, access to the physical system should be restricted to authorized personnel (using badge access, for example). Entry into and exit from the restricted areas should be monitored and tracked.

When the physical environment is established, one can move on to look more closely at the security possibilities offered by the server infrastructure. For System z and z/VM, everything begins with the integrity statements issued by IBM in the early days of the life of the mainframe. Although dated, the integrity statements are still valid today. They offer peace of mind to IT organizations that have chosen to use System z as their server platform.

The z/VM integrity statement describes how the z/VM Control Program (CP) has the ability to operate without any harm or interference from guest virtual machines, whether it is intentional or not. The z/VM integrity statement also describes how virtual machines are unable to circumvent access and other system security controls and how the CP can protect individual virtual machines from interfering with or harming each other. This collection of control and security measures is all based on a very strong architecture that is implemented across hardware and firmware.

All products and procedures to secure the z/VM environment are built on this architecture, including the use of RACF, directory maintenance procedures, secure access to the HMC and to consoles, as well as the way a secure infrastructure can be built by using firewall and multizoning technologies.

6.2 Minimal Installations

Minimizing the installation requirements or dependencies reduces the number of potential points of exposure. Ensuring the bare minimum gets installed not only simplifies administration but also avoids potential risks. Keeping installations at minimal dependency needs should be a constant, that is, if services are eventually disabled, it is a good practice to also remove those services along with their dependencies, if they are not required by other packages.

6.3 Protecting the Hardware Management Console

The Hardware Management Console (HMC) is a formidable and powerful entry point into controlling a System z environment. Ultimate power in the hands of a senior operator or a systems programmer should not be granted automatically. A strong suggestion is not to share user IDs and passwords on the HMC. Every single user who has a need to access the System z environment through the HMC should be given an individual user ID and an individual access profile, depending on what each user is allowed to do to which system resources. Control over auxiliary user IDs such as the HMC user ID should be taken as seriously as workstation logon IDs, that are usually among the first to be disabled when an employee leaves a company.

6.4 Protecting the configuration

Personnel who need access to z/VM resources should be granted access individually, with individual user IDs and authorities. One strong example is the ability to change the configuration through the Dynamic I/O commands. The feature to allow dynamic changes to the I/O configuration is available on the current System z product set. It allows for deletions and additions to an active System z environment. The authority to issue such commands should be restricted to a single LPAR on each server and access to the facility should be strongly protected.

6.5 Building a secure multizone application environment

Modern application infrastructures often have individual pieces of applications that are deployed on a number of distinct servers (a multitier structure). The classical structure of user presentation, application serving, and database serving represents an example with three separate servers or zones. As application systems evolve, we will see additional zones appear for infrastructure, applications and data. See Figure 6-1 for an example.

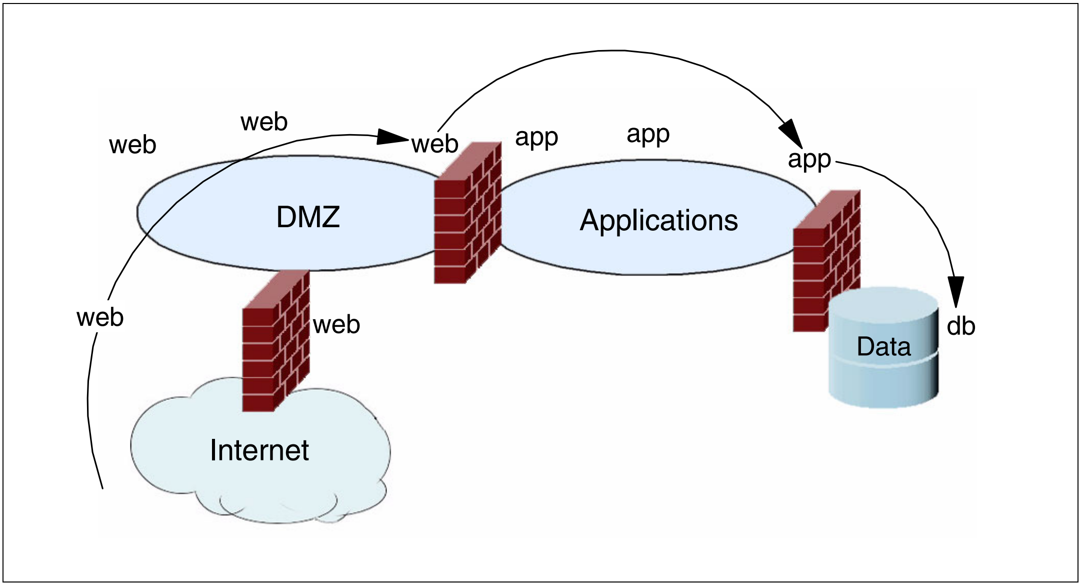


Figure 6-1 A multizone network

6.5.1 The multizone concept

In a multitier, or multizone, configuration of individual server images, or groups of server images, they are isolated from one another using firewall technologies. Firewalls implement a set of authorization rules to restrict user access or avoid certain data flows between zones.

Firewalls and where to host them

A firewall is a part of a computer system or a network designed to block unauthorized access and at the same time allow authorized access to computing resources such as servers, databases, and applications. A firewall can consist of one or more devices designed to permit or deny access between different security domains (zones) based on administrative security and access criteria. Firewall technologies have undergone major improvements over time from being simple packet-filtering appliances to become very sophisticated, application-sensitive, and protocol-sensitive watchdogs, protecting IT environments.

Where would you want to host your firewall technology? This is a very simple and short question, to which the answer is long and complex. Based on the System z architecture and virtualization, firewalls can be implemented in logical partitions or in z/VM virtual machines. The inherent isolation between logical partitions and between z/VM virtual machines makes such a solution as secure as having the firewalls executing in individual distributed servers. Similarly, one can securely implement multiple security zones across one single System z server. From a simplification perspective, that could be a desirable configuration because there will be no external boxes. A number of IBM System z clients have implemented such a configuration. If you want more information about how a firewall can be implemented in Linux on System z see 6.7, “Linux firewalls” on page 203.

However, from a pure capacity and cost perspective, a typical firewall workload does not represent a particularly suitable solution for the System z microprocessor architecture. Further, if your environment already has a firewall strategy based on stand-alone appliance-like firewall servers, you might have to use that as a starting point for your firewall strategy also for the System z environment.

For these reasons, among others, some clients have preferred to implement their zoning on System z using external firewalls, or a combination of the two. See Figure 6-2 for an example of a network that is based on System z with external and internal firewalls.

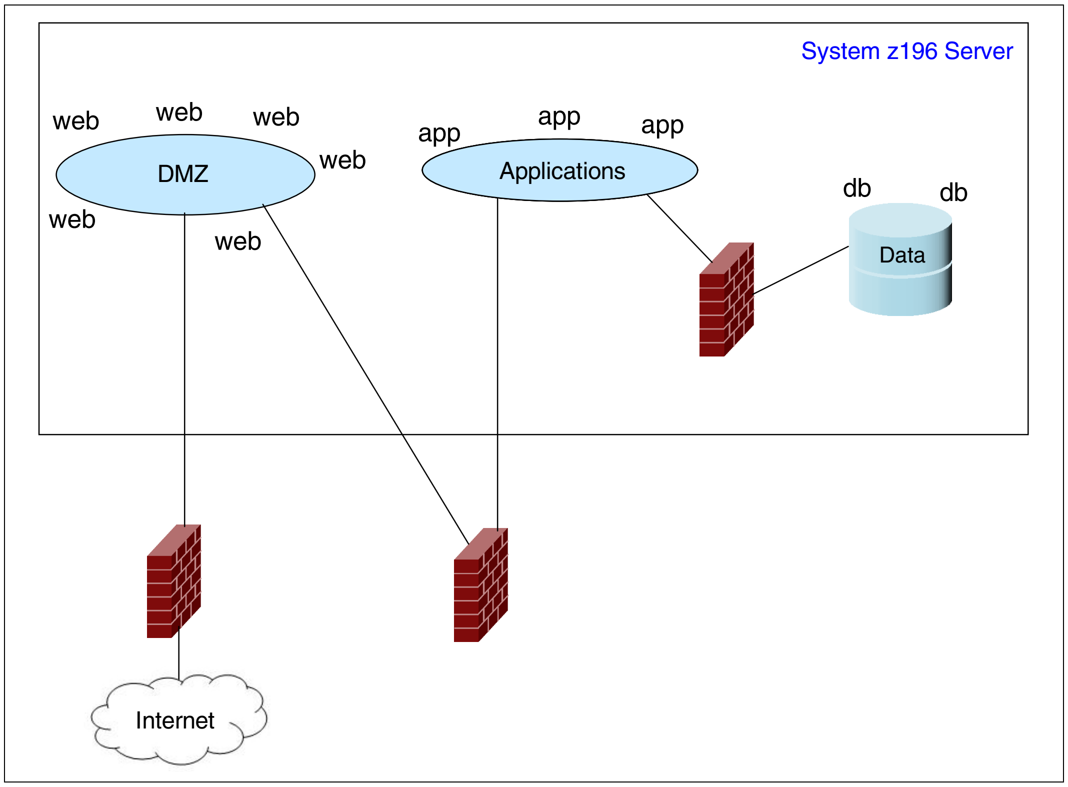


Figure 6-2 System z-based multizone network

6.5.2 Controlling the zones with RACF

The optional RACF feature on z/VM can be used to control your multizoned environment. Using Labeled Security, a dated, well-established concept in the industry, RACF can be initialized to implement multilevel security. Labeled Security is part of z/VM Common Criteria certification and RACF is the only external security manager (ESM) that offers this functionality. See Chapter 8, “Best practices” on page 265 for details about securing your zoned environment.

6.5.3 Using HiperSockets as part of your network solution

IBM HiperSockets represents yet another unique System z technology. Implemented as a standard network interface, HiperSockets offers an internal network connection between logical partitions. The implementation is a memory-to-memory transfer between network stacks, delivering a high-bandwidth low-latency vehicle for data transport. The participating network stacks see the HiperSockets interface as just another network interface card (NIC) so no special programming or driver is necessary. However, from a security perspective, the HiperSockets should be treated as any other network device, protected by some kind of firewall technology. Figure 6-3 shows a very basic example of where the HiperSockets provides connectivity from a z/VM environment into a z/OS logical partition. Note the internal firewall between the application zone and the HiperSockets.

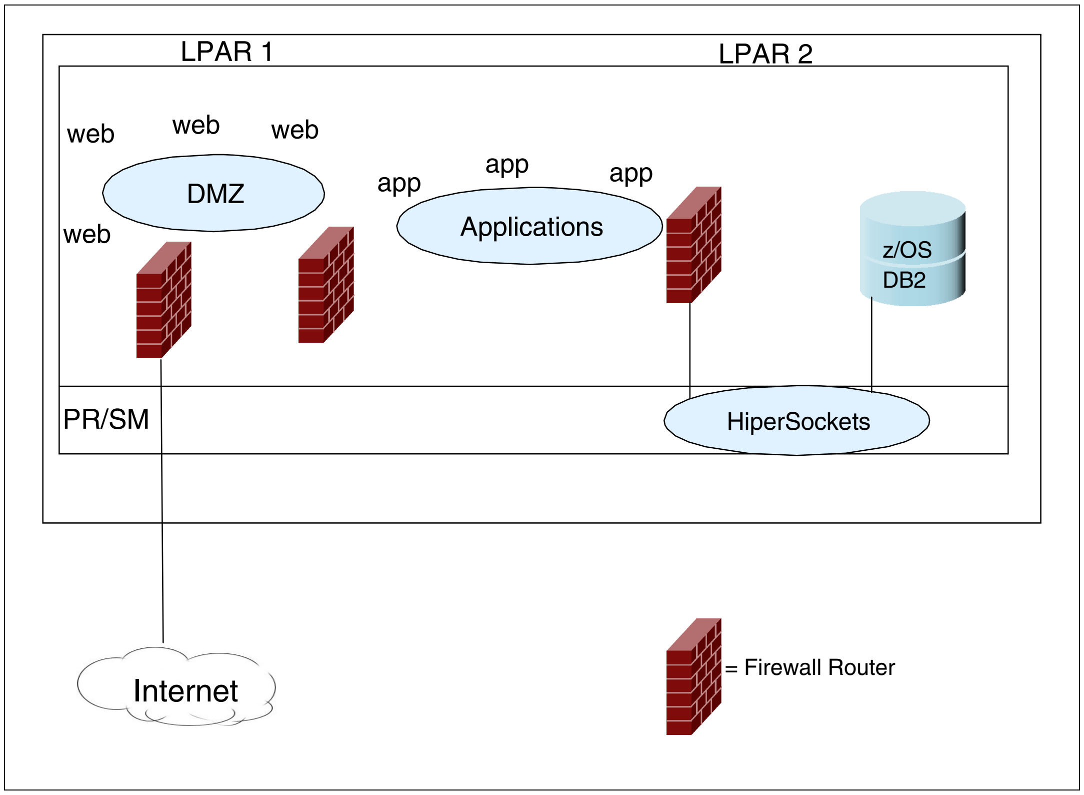


Figure 6-3 Using HiperSockets in your System z network

Numerous other possibilities exist for protecting the HiperSockets connections. Figure 6-4 on page 200 shows an example where the HiperSockets connection is protected on the z/OS side by using z/OS packet filters.

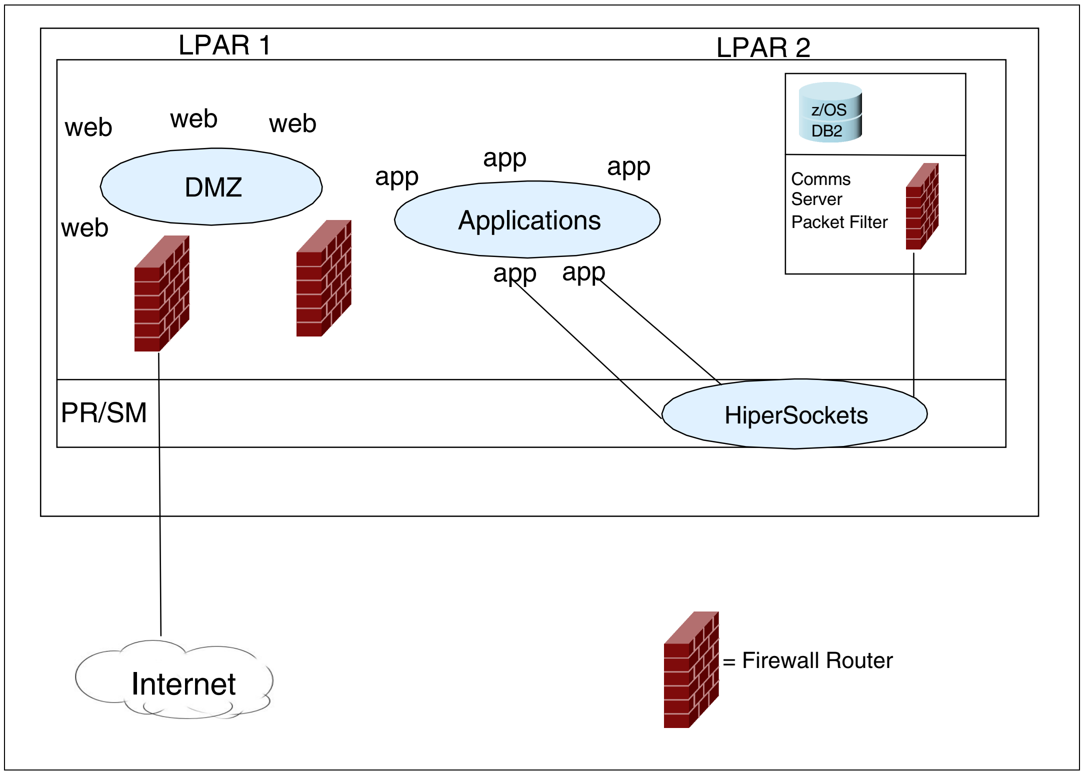


Figure 6-4 Using HiperSockets and z/OS packet filtering

Before deploying HiperSockets solutions, be sure to discuss the possible solutions with your network organization. The organization might already have similar solutions in operation, and might impose a specific configuration on your z/VM and Linux implementation.

For a discussion of auditing the infrastructure, see 2.6.5, "Centralized audit" on page 33.

6.6 IBM security solutions

The IBM world-class security solutions address risks across each aspect of a business, helping build a strong security posture that helps reduce costs, improve service, and manage infrastructure risks, cost-effectively embracing change and innovation without compromising security.

IBM Security Network Intrusion Prevention System, and IBM Tivoli zSecure™ Manager for RACF z/VM are particularly pertinent to be mentioned in this book. For more security solutions, refer to "Online resources" on page 321.

6.6.1 IBM Security Network Intrusion Prevention system

IBM Security Network Intrusion Prevention solutions provide comprehensive protection while reducing the cost and complexity associated with deploying and managing point solutions. This includes going beyond traditional network intrusion prevention, including the ability to:

- Deliver Advanced Threat Detection and Prevention to stop targeted attacks against high value assets.
- Protect your systems before you have time to patch with IBM Virtual Patch® technology.

- Protect web applications from threats such as SQL Injection and Cross-site Scripting attacks.
- Integrated Data Loss Prevention (DLP) monitoring data security risks throughout your network.
- Prevent network misuse or abuse from instant messaging and peer-to-peer file sharing.
- Integrate with other security solutions such as IBM Rational® AppScan® in order to leverage AppScan’s intelligence to proactively show what needs to be protected.
- Support flexible network deployments with new high performance security appliances and virtual security appliances.
- Provide Ahead of the Threat® protection backed by world renowned IBM X-Force® Research.

6.6.2 IBM Tivoli zSecure Manager for RACF z/VM

IBM Tivoli zSecure Manager for RACF z/VM provides administrators with tools to help unleash the potential of the mainframe system, enabling efficient and effective RACF administration, while helping use fewer resources by automating many recurring system administration functions. Features of IBM Security zSecure Admin and IBM Security zSecure Audit components for z/OS that form the foundation for the IBM Security zSecure Manager for RACF z/VM include:

- Automation of complex, time-consuming z/VM security management tasks with simple, one-step actions that can be performed without detailed knowledge of RACF command syntax
- Creation of efficient comprehensive audit trails and reports to measure and verify the effectiveness of the z/VM security and compliance policies without substantial manual effort
- Easing the burden of database and system consolidations
- Quick identification and prevention of problems in RACF before they become a threat to security and compliance
- Generation and viewing of customized audit reports with flexible schedule and event sections

6.6.3 IBM Tivoli Endpoint Manager for Patch Management

IBM Tivoli Endpoint Manager for Patch Management, built on BigFix technology, delivers an easy-to-manage, quick-to-deploy solution that supports comprehensive patch management capabilities among distributed endpoints, reducing business risk, costs, complexity and time while enhancing security. Features include:

- Automatically manage patches for multiple operating systems and applications across hundreds of thousands of endpoints regardless of location, connection type or status.
- Reduce security risk by slashing remediation cycles from weeks to days or hours.
- Gain greater visibility into patch compliance with flexible, real-time monitoring and reporting.
- Provide up-to-date visibility and control from a single management console.
- Efficiently deploy patches, even over low-bandwidth or globally distributed networks.
- Automatically remediate problems related to previously applied patches.

- Patch endpoints on or off the network, including roaming devices using Internet connections.

6.6.4 IBM Tivoli Endpoint Manager for Security and Compliance

IBM Tivoli Endpoint Manager for Security and Compliance, built on BigFix technology, is an easy-to-manage, quick-to-deploy solution that supports security among distributed endpoints, reducing costs and complexity of management while increasing business agility, speed to remediation, and accuracy. Features include:

- Automate time-consuming device configuration and change management tasks
- Effectively manage the compliance life cycle with an ongoing, closed-loop process
- Gain greater visibility into network resources in dynamic and complex environments
- Provide accurate, precise and up-to-the minute visibility into and continuous enforcement of security configurations and patches
- Centralize the management of functions that provide advanced anti-virus and firewall protection
- Employ a unified management infrastructure to coordinate among IT, security, desktop and server operations
- Reach endpoints regardless of location, connection type or status with comprehensive management for all major operating systems, third-party applications and policy-based patches.

6.6.5 IBM Tivoli Access Manager WebSEAL

IBM Tivoli Access Manager WebSEAL is a security manager for web-based resources. WebSEAL is a high performance, multithreaded web server that applies fine-grained security policy to the protected web object space. WebSEAL can provide single sign-on solutions and incorporate backend web application server resources into its security policy. You can use WebSEAL in conjunction with LDAP running on z/OS to protect web-based resources. The high-level scenario is depicted in Figure 6-5.

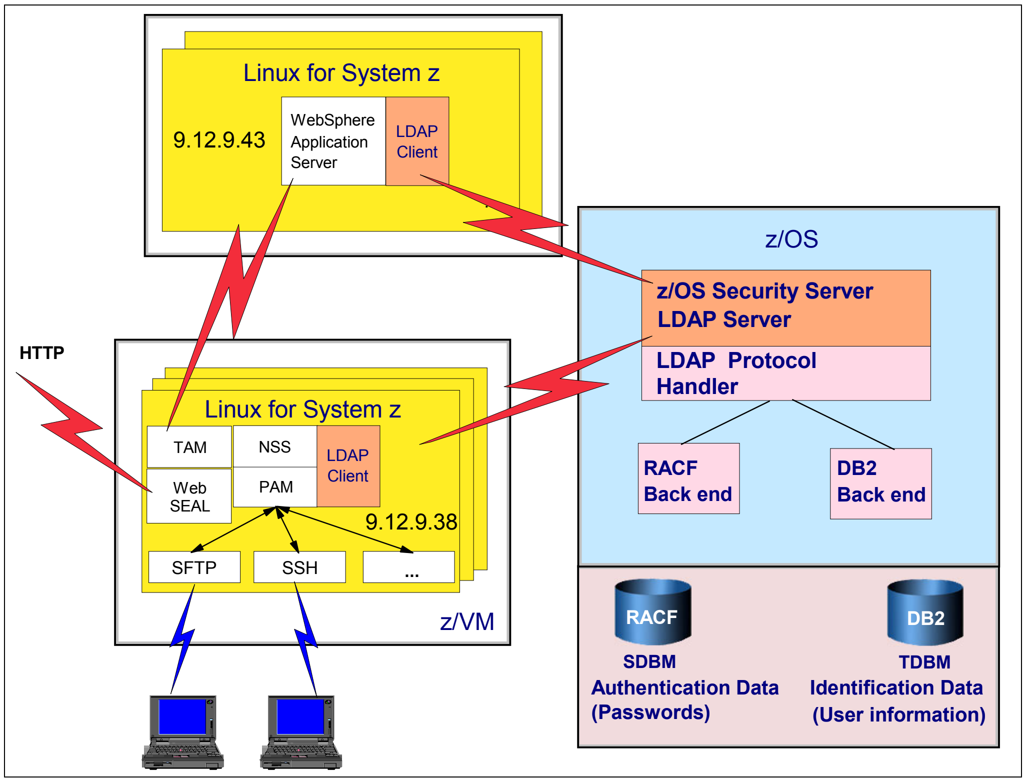


Figure 6-5 Authenticating users against a z/OS RACF database

In this scenario, we use a single Tivoli Access Manager instance. For scalability and high availability, several Tivoli Access Manager and Linux instances are required. A request-spraying mechanism distributes the workload across all instances.

For a more detailed description on how to implement these functions, see *Linux on IBM eServer zSeries and S/390: Best Security Practices*, SG24-7023.

6.7 Linux firewalls

Being a multiple purpose kernel, Linux includes a multitude of network functionalities, including sophisticated firewall features that can filter and manipulate packets, based on complex rules that you define. These features make Linux an ideal operating system for software firewalls, and because of the platform-independent nature of Linux you can exploit the features on System z as you can on any other platform.

In this chapter, we introduce you to *netfilter*, the Linux packet filtering framework. After a short introduction to the underlying concepts, we show how to set up your firewall manually by using a command-line utility and with the help of tools. We show examples based on a restrictive firewall policy rather than a permissive policy, by dropping (instead of rejecting) packets that are not explicitly allowed.

6.7.1 The ebtables tool

ebtables is a layer 2 filtering tool for Linux bridge firewalls, providing logging, DNAT/SNAT, and other features. It can be used along with other Linux filtering tools, and is available by default on Red Hat Enterprise Linux 6.2 as a module, so it can be loaded with the command shown in Example 6-1. It is not available on SUSE Linux Enterprise 11SP2, so you would have to build your own kernel, enabling ebtables on it.

Example 6-1 Loading the ebtables module on Red Hat Enterprise Linux 6.2

```
[root@lnxrh60a]# modprobe ebtables
```

Example 6-2 is simple yet useful for preventing MAC address spoofing if your Linux server is acting as bridge. It will not allow packets to get out if the source mac address is not 34:40:0e:d7:a5:d0, which we know is the one behind br0. MAC address spoofing can be used against an external network switch that is vulnerable to MAC flooding.

Example 6-2 ebtables command to prevent MAC spoofing

```
$ ebtables -A FORWARD -i br0 -s ! 34:40:0e:d7:a5:d0 -j DROP
```

6.7.2 The iptables tool

Iptables is an administration tool that controls the IPv4 packet filter rules and network address translation (NAT) in the Linux kernel. Iptables ensures that your systems are protected from external sources, such as hackers, and from seemingly benign internal sources. Forcing all traffic through the security systems ensures that your servers will remain intact even if a trusted system introduces a threat to your network. For IPv6 the ip6tables command-line tool should be used, almost fully with the same parameters. Note that netfilter for IPv6 does not yet include NAT features.

Iptables can be used for the following tasks:

- Protecting your internal network, which is connected to the Internet, from outside intruders
- Performing network address translation (NAT), which allows internally connected computers without a registered Internet address to reach Internet resources
- Filtering the packets that are going in or out of your internal network (or in or out of only one computer)

When data is sent from source to destination computers, it subdivides into a number of packets, which is a process that is done by using the network protocol. Each of these packets has a small part of the file data. These packets have source and destination system details and information of what type of packet it is. Most of the packets are for carrying the data, while some are for carrying protocols, which contain no data—they are for initiating communication between two systems. Upon receiving the transmission, the target computer reassembles the packets into the file.

When the packet is received it is checked against rules. The arrangement and reason of these rules can vary, but they usually identify a packet coming from or going to a particular IP address when using a particular protocol and network service. When packets match a particular rule, they are designated for a particular target or action to be applied to them.

A generic way of checking whether there are any active iptables rules is shown in Example 6-3 on page 205. By default the rules for the *filter* table are shown, since no -t parameter is passed. Table options are *filter*, *nat*, *mangle*, and *raw*. So, for example, you can check the rules in the *mangle* table by adding *-t mangle* to the command shown in Example 6-3.

Example 6-3 Checking the status of iptables on most Linux distributions

```
[root@lnxrh1 ~]# iptables -L
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
```

Iptables rules can be flushed, stopped, started, or restarted after booting from a command line. To permanently activate/deactivate iptables, use **`chkconfig`** on RHEL or **`SuSEfirewall2`** on SLES. Example 6-4 shows how to change the iptables status on RHEL.

Example 6-4 Starting, stopping, and restarting iptables on RHEL

```
[root@lnxrh1 ~]# service iptables start
iptables: Applying firewall rules: [  OK  ]
[root@lnxrh1 ~]# service iptables stop
iptables: Flushing firewall rules: [  OK  ]
iptables: Setting chains to policy ACCEPT: filter [  OK  ]
iptables: Unloading modules: [  OK  ]
[root@lnxrh1 ~]# service iptables restart
iptables: Flushing firewall rules: [  OK  ]
iptables: Setting chains to policy ACCEPT: filter [  OK  ]
iptables: Unloading modules: [  OK  ]
iptables: Applying firewall rules: [  OK  ]
```

Example 6-5 shows how to change the iptables status on SLES.

Example 6-5 Starting, and stopping iptables on SLES

```
lnxslesa:~ # SuSEfirewall2 start
SuSEfirewall2: Setting up rules from /etc/sysconfig/SuSEfirewall2 ...
SuSEfirewall2: batch committing...
SuSEfirewall2: Firewall rules successfully set
lnxslesa:~ # SuSEfirewall2 stop
SuSEfirewall2: batch committing...
SuSEfirewall2: Firewall rules unloaded.
lnxslesa:~ # SuSEfirewall2 status
SuSEfirewall2: SuSEfirewall2 not active
```

To view a list of all the rules assigned in iptables, see Example 6-6. A newly installed Linux image does not have any of the iptables rules.

Example 6-6 Check list of all assigned rules in iptables

```
[root@lnxrh1 ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
```

```
target     prot opt source               destination
```

Basic options in iptables

Iptables has three levels: tables, chains, and rules. They are discussed in this section.

Tables

Tables are results with a specific desired outcome. The built-in tables in the iptables system include:

- The filter table accepts or denies packets based upon a definable rule set.
- The NAT table translates the source or destination address of a packet based upon a definable rule set.
- The mangle table alters other aspects of a packet based upon a definable rule set.
- The raw table is used to set a mark on the packets so that they should not be handled by the connection tracking system.
- The security table for iptables enables mandatory access control (MAC) networking rules to be managed separately from discretionary access control (DAC) rules.

Chains

The Linux kernel starts with three lists of rules called chains (see Table 6-1). Chains can be used to apply the same action on different types of packets. In the table, the Chain column lists the names of the chains.

Table 6-1 Built-in chains and their functions

Chain	Chain function
INPUT	Handles incoming packets.
FORWARD	Handles packets being routed or forwarded.
OUTPUT	Handles outgoing packets.

In this section we discuss the most commonly used chain options. To create a new chain with the name `chainlist`, see Example 6-7.

Example 6-7 Create a chain

```
[root@lnxrh1 ~]# iptables -N chainlist
```

To delete a chain with the name `chainlist`, see Example 6-8.

Example 6-8 Delete a chain

```
[root@lnxrh1 ~]# iptables -X chainlist
```

To add rules to a chain, see Example 6-9.

Example 6-9 Add a rule to a chain

```
[root@lnxrh1 ~]# iptables -A chainlist -s 9.12.5.92/24 -j ACCEPT
```

To modify a chain to drop any packet that matches a rule, see Example 6-10 on page 207.

Example 6-10 DROP packets

```
[root@lnxrh1 ~]# iptables -A chainlist -s 9.12.5.90/24 -j DROP
```

To flush the rules out of the chain, see Example 6-11.

Example 6-11 Flush the chain

```
[root@lnxrh1 ~]# iptables -F chainlist
```

To reset the packet and byte counter for `chainlist`, see Example 6-12.

Example 6-12 Example to reset packets

```
[root@lnxrh1 ~]# iptables -Z chainlist
```

Rules

Rules are patterns that iptables uses to determine what action must be taken on a specific packet. To append a rule to the filter table into the built-in INPUT chain, use the following command:

```
[root@lnxrh1 ~]# iptables -A INPUT rule
```

To delete a rule from the INPUT chain in the filter table, use the following command:

```
[root@lnxrh1 ~]# iptables -D INPUT rule
```

To list current rules and their number, use the following command:

```
[root@lnxrh1 ~]# iptables --line-numbers -L
```

To replace a rule in the INPUT chain in the filter table the rule number must be informed, using the following command syntax:

```
[root@lnxrh1 ~]# iptables -R INPUT rule_number rule
```

To append a rule in the OUTPUT chain in the NAT table, use the following command:

```
[root@lnxrh1 ~]# iptables -t nat -A OUTPUT rule
```

Example 6-13 shows how packets can be accepted (policy ACCEPT). The numbered sections of the example are explained after the example.

Example 6-13 ACCEPT packets

```
1 [root@lnxrh1 ~]# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

```
2 [root@lnxrh1 ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain chainlist (0 references)
target     prot opt source               destination
```

```
3 [root@lnxrh1 ~]# iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
```

```
4 [root@lnxrh1 ~]# iptables -L
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:webcache
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain chainlist (0 references)
target     prot opt source               destination
```

The commands in the example are as follows:

1 Allows SSH to connect to external addresses.

2 Shows services that are blocked or allowed.

3 Enables alternative Web servers or services to connect to external addresses.

4 Lists the iptables again to show that both SSH and the `webcache` (port 8080) are now able to connect to external addresses.

Example 6-14 has a listing of the iptables rules and shows that only SSH connections and alternative web services are allowed.

Example 6-14 Listing of iptables

```
[root@lnxrh1 ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:webcache
DROP       all  --  anywhere             anywhere
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain chainlist (0 references)
target     prot opt source               destination
```

6.7.3 Linux firewall tools

Although one can build a firewall manually by using the iptables tool, many administrators want to use tools to generate rules from semantically more dense configuration files. Because techniques of firewalling with Linux on System z are the same as on any other Linux platform, the same tools can be used to manage the firewall.

In this section we present a popular firewall tool and how to set it up on Linux on System z, along with brief comments about Red Hat, and SUSE's own tools. Configuring actual firewalls is not covered because this topic is vast and is very dependent on the actual network topology.

Firewall Builder

No binary package for *fwbuilder* is currently available in either SUSE Linux Enterprise Server 11 SP2 or Red Hat Enterprise Linux 6.2, so we have to build one from source files. The fwbuilder developers provide a source RPM package that helps build a binary package by using the standard build tools on both SUSE and Red Hat.

Make sure you have the development packages available on a repository because some are required for building the fwbuilder package itself. Obtain the fwbuilder source RPM from either of the following locations:

- Firewall Builder Web site

  `http://www.fwbuilder.org`

- OpenSUSE repositories

  `http://download.opensuse.org/repositories/home:/worldcitizen/openSUSE_11.2/src/`

Note: Always be careful when downloading any software from a public source. Many ways exist for an attacker to alter the content of both binaries and sources either during transfer or on the server you download from. Unless you review all downloaded source code, building packages from source does not guarantee that you can trust the final binary. Be sure to take precautions to verify the integrity of all downloaded files by checking signatures if any are available. We also recommend that you verify the validity of keys, which are used for signing, by building chains of trust or obtaining key fingerprints through alternative channels.

Example 6-15 and Example 6-16 detail the commands that you issue to build the *fwbuilder* binary package for the s390x architecture on SUSE and Red Hat, respectively. If you receive any error messages in the **rpmbuild** steps you might be missing required libraries. Check the errors and, if necessary, install the indicated package dependencies.

Example 6-15 Building fwbuilder from source RPM on SUSE Linux Enterprise Server 11 SP2

```
root@linux:~ # rpm -i fwbuilder-5.1.0.3599-17.1.src.rpm
root@linux:~ # cd /usr/src/packages/
root@linux:/usr/src/packages # rpmbuild -bb SPECS/fwbuilder.spec
[ ... ]
root@linux:/usr/src/packages # zypper install
RPMS/s390x/fwbuilder-5.1.0.3599-17.1.s390x.rpm
```

Example 6-16 Building fwbuilder from source RPM on Red Hat Enterprise Linux 6.2

```
root@linux:~ # rpm -i fwbuilder-5.1.0.3599-17.1.src.rpm
root@linux:~ # cd /root/rpmbuild
root@linux:~ # sed -i ‘s/libqt4-devel/qt4-devel/’ SPECS/fwbuilder.spec
root@linux:/root/rpmbuild # rpmbuild -bb SPECS/fwbuilder.spec
[ ... ]
root@linux:/root/rpmbuild # yum install
RPMS/s390x/fwbuilder-5.1.0.3599-17.1.s390x.rpm
```

If successful, you now have fwbuilder installed on your Linux server, although no firewall has yet been defined. You must set up and configure a new firewall by using the fwbuilder GUI first, which requires a graphical environment.

The two major options for running GUI applications on Linux on System z are:

- Running an X Server on your client machine and forwarding connections through an SSH connection from the remote Linux system
- Using a VNC server on the remote system

We use the latter option because it is resilient to intermittent network errors because the VNC server continues running even if your connection is lost. You can simply reconnect to your VNC session as soon as the network is available again. Connect to the user account on the remote server and run the command shown in Example 6-17 to start the VNC server. Provide a session password if asked.

Example 6-17 Starting the VNC server

```
[user@linux ~]$ vncserver
Starting VNC server: 2:user
New 'linux:2 (user)' desktop is linux:2

Starting applications specified in /home/user/.vnc/xstartup
Log file is /home/user/.vnc/linux:2.log
```

To reset the VNC session password, run **vncpasswd**, as shown in Example 6-18.

Example 6-18 Setting or resetting the VNC server session password

```
[user@linux ~]$ vncpasswd
```

Note that the VNC server session shown in Example 6-17 was started on display :2, therefore you should connect to it from a client system using the command shown in Example 6-19.

Example 6-19 Connecting to the remote VNC session from a Linux client using vncviewer

```
[client@linuxworkstation ~]$ vncviewer 9.12.5.104:2
```

Note: To build a firewall configuration, you do not have to run VNC or the fwbuilder GUI as the root user. In fact the authors strongly advise against that approach because we suggest that the root account only be used when absolutely necessary. For using fwbuilder, you only have to use root if you apply the generated firewall settings to a server and for that we suggest using the *sudo* mechanism to provide appropriate access control and logging.

Figure 6-6 on page 211 shows the initial layout of the fwbuilder GUI. In the left tree view, define your firewall objects, or choose from a list of predefined objects by selecting a template from the drop-down list above the tree view. The right side displays firewall rules and object properties. The three ways to create a new firewall are:

- Use a preconfigured template firewall object.
- Create it from scratch.
- Use SNMP to create a firewall object with interfaces but an empty policy.

In this example, we create it from scratch.

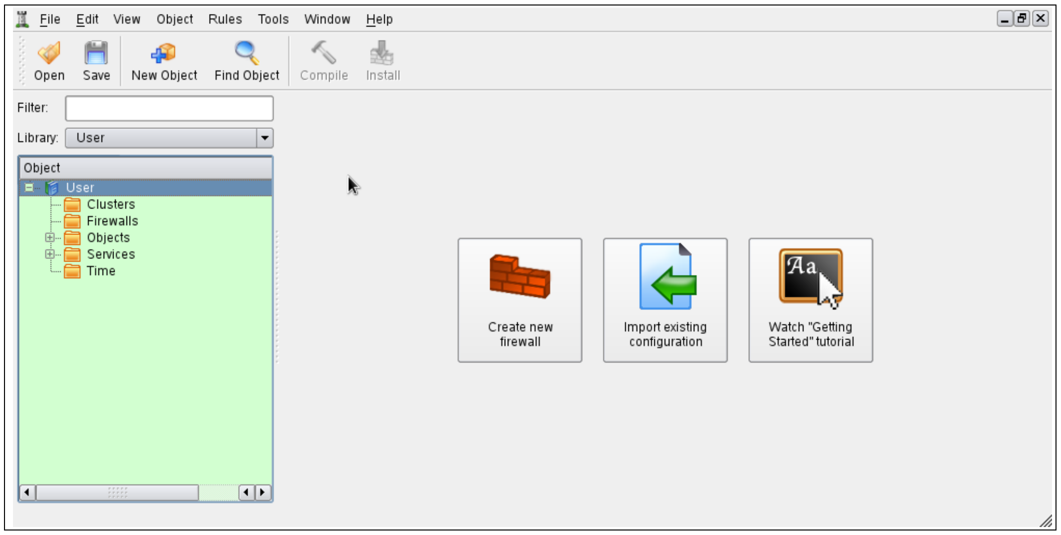
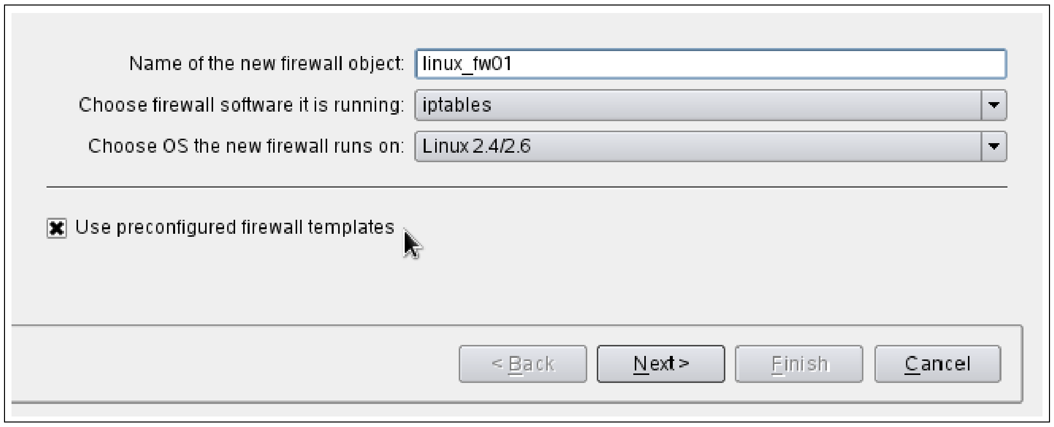


Figure 6-6 Initial view of fwbuilder GUI

Create a new firewall by clicking **Create new firewall**. A dialog opens, as shown in Figure 6-7. For this short introduction, select **Use preconfigured firewall templates** and set the operating system and firewall software. Then, click **Next** to move to the next window.



Name of the new firewall object: linux_fw01

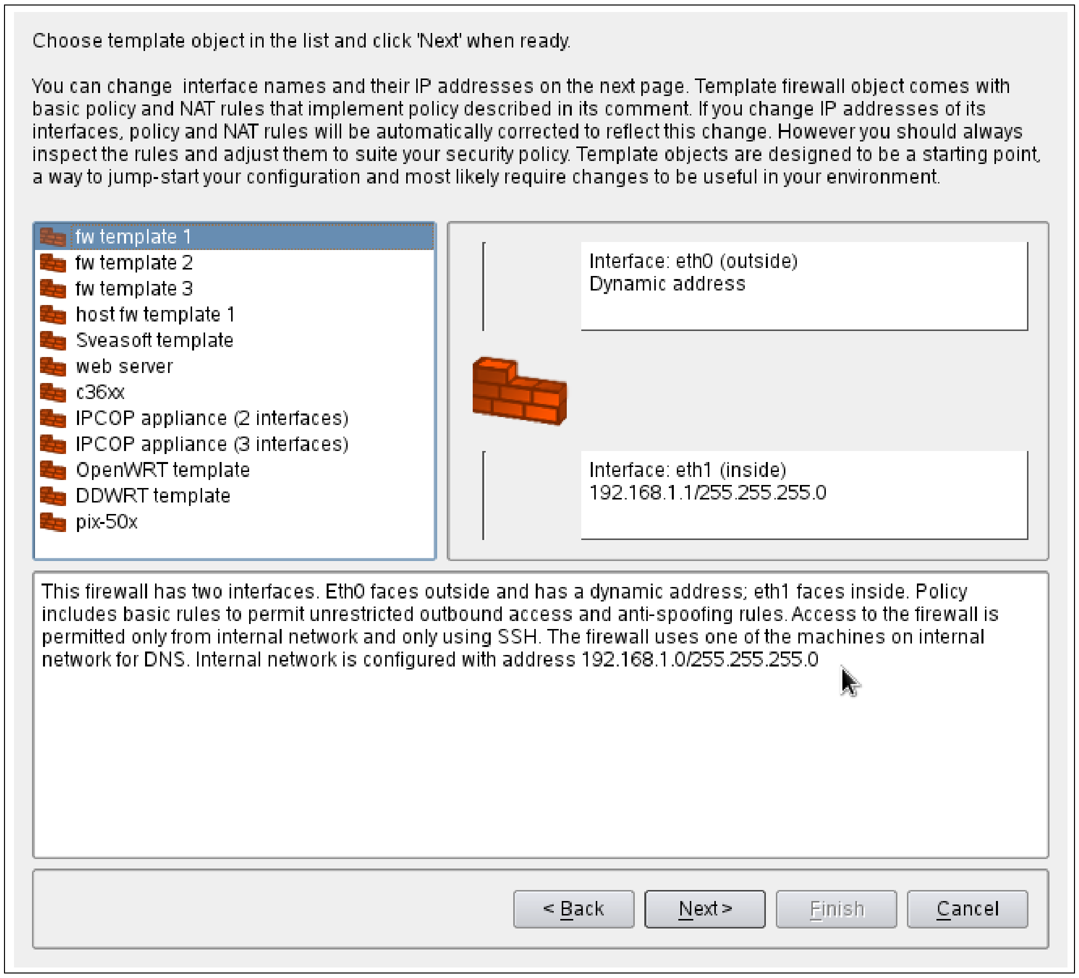
Choose firewall software it is running: iptables

Choose OS the new firewall runs on: Linux 2.4/2.6

☒ Use preconfigured firewall templates

Figure 6-7 The create new firewall dialog in fwbuilder

The dialog shown in Figure 6-8 on page 212 opens, so you may choose from a few preconfigured firewall templates. From the list, we select the first and most basic template, **fw template 1**. This template requires two network interfaces to be present on the system. You can use any virtual network device that you have defined for your Linux guest, whether it is an Ethernet or IP only device. Click **Next** to select and load the template.


Choose template object in the list and click 'Next' when ready.

You can change interface names and their IP addresses on the next page. Template firewall object comes with basic policy and NAT rules that implement policy described in its comment. If you change IP addresses of its interfaces, policy and NAT rules will be automatically corrected to reflect this change. However you should always inspect the rules and adjust them to suite your security policy. Template objects are designed to be a starting point, a way to jump-start your configuration and most likely require changes to be useful in your environment.

- fw template 1
- fw template 2
- fw template 3
- host fw template 1
- Sveasoft template
- web server
- c36xx
- IPCOP appliance (2 interfaces)
- IPCOP appliance (3 interfaces)
- OpenWRT template
- DDWRT template
- pix-50x

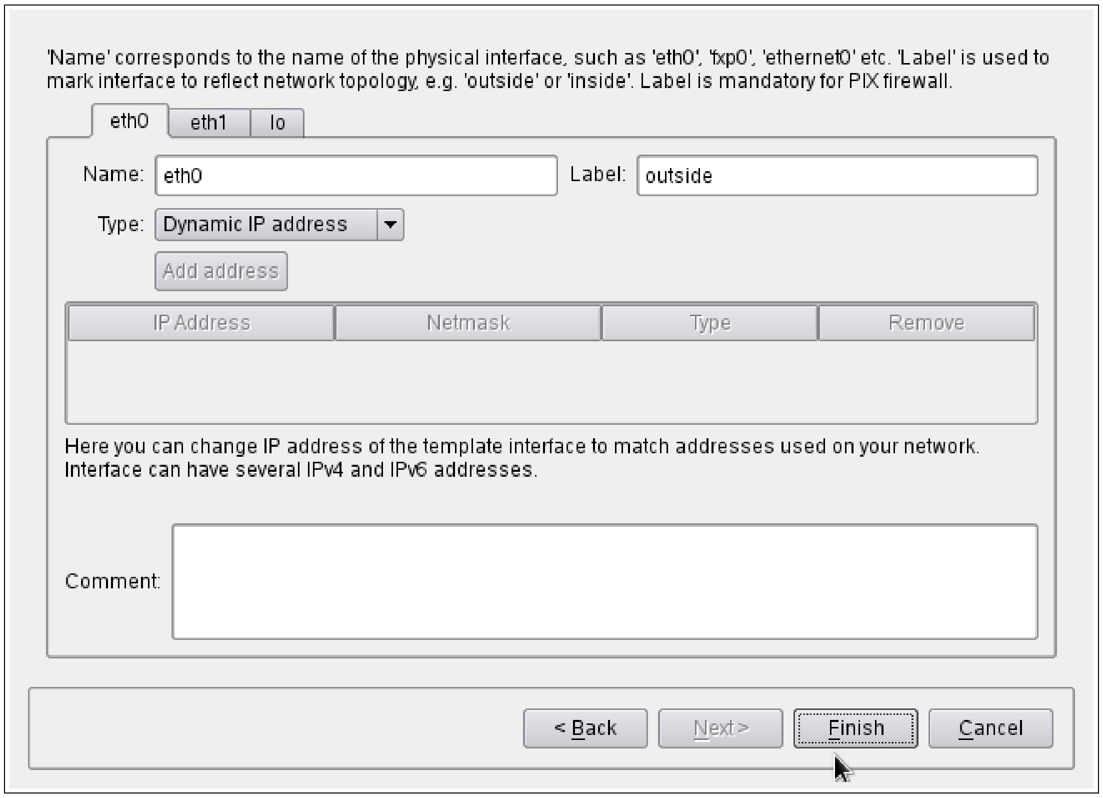
Interface: eth0 (outside)
Dynamic address

Interface: eth1 (inside)
192.168.1.1/255.255.255.0

This firewall has two interfaces. Eth0 faces outside and has a dynamic address; eth1 faces inside. Policy includes basic rules to permit unrestricted outbound access and anti-spoofing rules. Access to the firewall is permitted only from internal network and only using SSH. The firewall uses one of the machines on internal network for DNS. Internal network is configured with address 192.168.1.0/255.255.255.0

Figure 6-8 Template selection in fwbuilder GUI

On the next panel, shown in Figure 6-9 on page 213, you can give the interfaces different labels and add specific addresses. For this example we are not changing anything, so simply click **Finish**.


'Name' corresponds to the name of the physical interface, such as 'eth0', 'fxp0', 'ethernet0' etc. 'Label' is used to mark interface to reflect network topology, e.g. 'outside' or 'inside'. Label is mandatory for PIX firewall.

Name: eth0   Label: outside

Type: Dynamic IP address

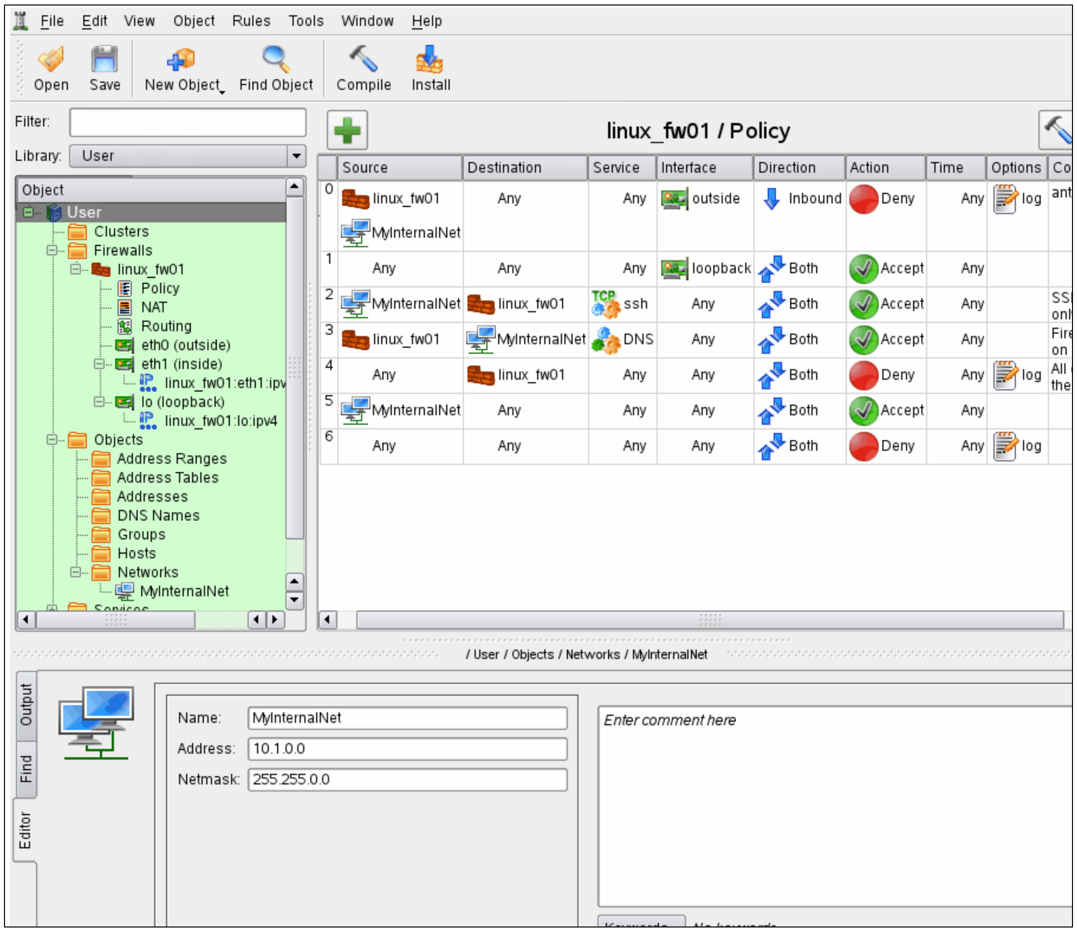
IP Address	Netmask	Type	Remove

Here you can change IP address of the template interface to match addresses used on your network. Interface can have several IPv4 and IPv6 addresses.

Comment:

Figure 6-9 Finishing the configuration

Starting from the template entries, you may customize your firewall rules to your needs. Figure 6-10 on page 214 shows a simple example where we replaced the default private network that is defined by the template with a new network, which covers the IP ranges that are considered to be local to the test system. As stated earlier, consult the fwbuilder documentation for an in-depth explanation of the available features.



linux_fw01 / Policy

	Source	Destination	Service	Interface	Direction	Action	Time	Options	Co
0	linux_fw01 MyInternalNet	Any	Any	outside	Inbound	Deny	Any	log	ant
1	Any	Any	Any	loopback	Both	Accept	Any		
2	MyInternalNet	linux_fw01	ssh	Any	Both	Accept	Any		SS onl
3	linux_fw01	MyInternalNet	DNS	Any	Both	Accept	Any		Fire on
4	Any	linux_fw01	Any	Any	Both	Deny	Any	log	All the
5	MyInternalNet	Any	Any	Any	Both	Accept	Any		
6	Any	Any	Any	Any	Both	Deny	Any	log	

Name: MyInternalNet
Address: 10.1.0.0
Netmask: 255.255.0.0

Enter comment here

Figure 6-10 Example of a modified template

When you are done configuring your firewall, right-click the firewall item in the object tree and then select the **Compile** menu entry. This step starts the process of compiling the firewall configuration for your target system. It also checks for configuration errors and reports them.

As you can see in Example 6-20, the compiled firewall is a simple shell script, containing the iptables commands that set the firewall rules. The script may also set various kernel parameters, depending on your configuration. Beware, though: Any manual changes that you make to this file are not subject to further error checks and can cause unexpected behavior.

Example 6-20 Firewall script generated by fwbuilder

```
#!/bin/sh
#
#  This is automatically generated file. DO NOT MODIFY !
#
#  Firewall Builder  fwb_ipt v5.1.0.3599
[ ... ]
$IPTABLES -N In_RULE_0
    for i_eth0 in $i_eth0_list
    do
    test -n "$i_eth0" && $IPTABLES -A INPUT -i eth0   -s $i_eth0   -j In_RULE_0
    done
    $IPTABLES -A INPUT -i eth0   -s 192.168.1.1   -j In_RULE_0
    $IPTABLES -A INPUT -i eth0   -s 10.1.0.0/16   -j In_RULE_0
    for i_eth0 in $i_eth0_list
    do
    test -n "$i_eth0" && $IPTABLES -A FORWARD -i eth0   -s $i_eth0   -j In_RULE_0
    done
    $IPTABLES -A FORWARD -i eth0   -s 192.168.1.1   -j In_RULE_0
    $IPTABLES -A FORWARD -i eth0   -s 10.1.0.0/16   -j In_RULE_0
    $IPTABLES -A In_RULE_0  -j LOG  --log-level info --log-prefix "RULE 0 -- DENY 
"
    $IPTABLES -A In_RULE_0  -j DROP
[ ... ]
```

The generated script is completely architecture-independent and can be used on any platform. You can also generate such a firewall script on a different Linux platform, such as an IBM System p®, and transfer it to Linux on System z.

Note: The authors advise not to install the fwbuilder GUI on your production firewall but to generate the firewall script on a separate dedicated machine and transfer it to the firewall server. Also, be sure that the integrity of the script is preserved, for example by using digital signatures.

To activate the new firewall, you have to execute the generated script as root. The firewall settings are applied instantly. Depending on the configuration itself, you could lose connection to the Linux system, so make sure your access through the z/VM 3270 console is still working.

Before using fwbuilder, be sure to check the fwbuilder web site:

http://www.fwbuilder.org

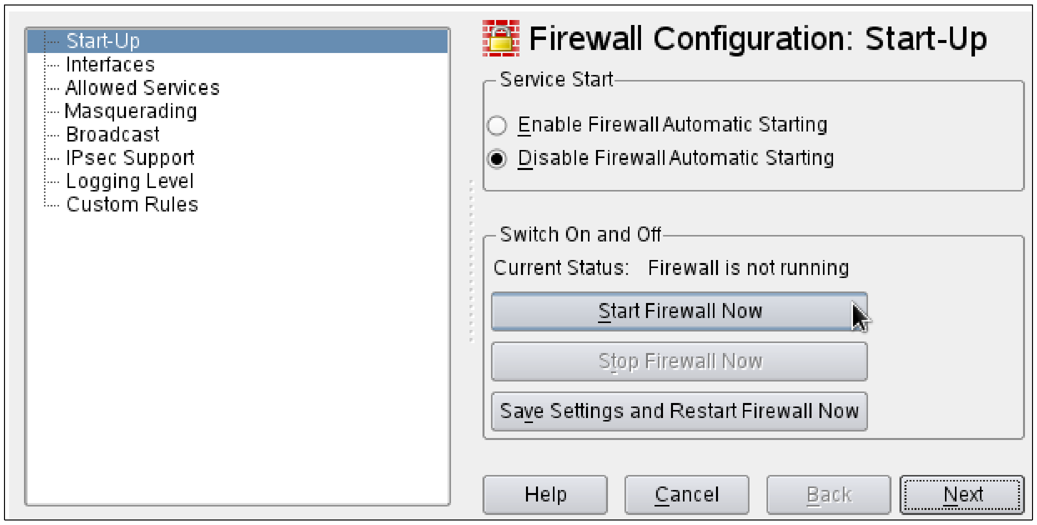
SUSE’s firewall configuration tool

SUSE Linux Enterprise offers a firewall configuration tool via Yast that works both in text and graphical modes. You can run it in graphical mode by connecting from a Linux desktop to the remote server via SSH using the X-forwarding parameter, as shown in Example 6-21.

Example 6-21   Starting yast2 firewall in graphical mode

```
[user@usersystem ~]$ ssh -X root@9.12.4.103 yast2 firewall
```

Figure 6-11 shows the Yast2 firewall window that pops up.



Firewall Configuration: Start-Up

Service Start

[ ] Enable Firewall Automatic Starting
[✓ Disable Firewall Automatic Starting]

Switch On and Off

Current Status: Firewall is not running

Figure 6-11 Yast2 firewall window

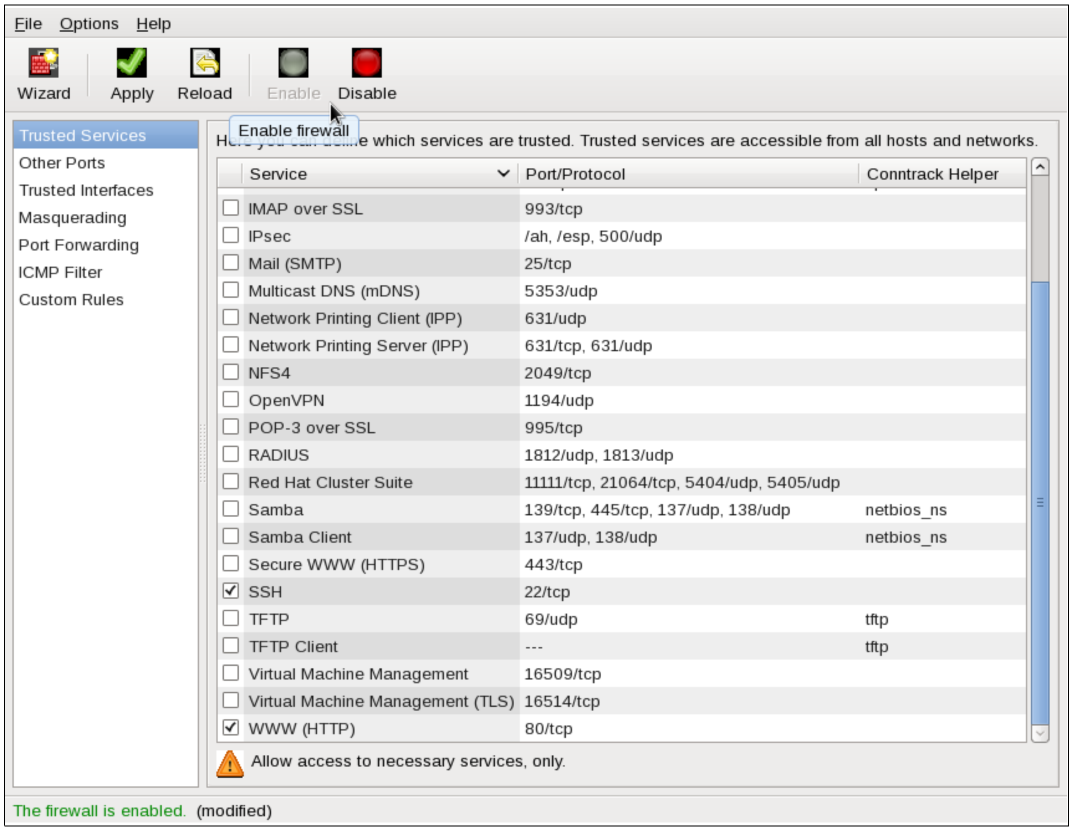
Red Hat's firewall configuration tool

Red Hat Enterprise Linux offers a firewall configuration tool called system-config-firewall, that works both in text and graphical modes. You can run it in graphical mode by connecting from a Linux workstation desktop to the remote server via SSH using the X-forwarding parameter, as shown in Example 6-22.

Example 6-22 Starting system-config-firewall in graphical mode

```
[user@usersystem ~]$ ssh -X root@9.12.4.103 system-config-firewall
```

Figure 6-12 on page 217 shows system-config-firewall that pops up. Start off by clicking **Enable** to enable the firewall. This will not apply any rules, but simply allow configuring the rules. Once all rules are set you will have to click **Apply** to activate use of the rules. Once enabled, you will be able to select what ports to allow for "Trusted Services" and other types of rules, including custom ones.



Here you can define which services are trusted. Trusted services are accessible from all hosts and networks.

	Service	Port/Protocol	Conntrack Helper
☐	IMAP over SSL	993/tcp	
☐	IPsec	/ah, /esp, 500/udp	
☐	Mail (SMTP)	25/tcp	
☐	Multicast DNS (mDNS)	5353/udp	
☐	Network Printing Client (IPP)	631/udp	
☐	Network Printing Server (IPP)	631/tcp, 631/udp	
☐	NFS4	2049/tcp	
☐	OpenVPN	1194/udp	
☐	POP-3 over SSL	995/tcp	
☐	RADIUS	1812/udp, 1813/udp	
☐	Red Hat Cluster Suite	11111/tcp, 21064/tcp, 5404/udp, 5405/udp	
☐	Samba	139/tcp, 445/tcp, 137/udp, 138/udp	netbios_ns
☐	Samba Client	137/udp, 138/udp	netbios_ns
☐	Secure WWW (HTTPS)	443/tcp	
✓	SSH	22/tcp	
☐	TFTP	69/udp	tftp
☐	TFTP Client	---	tftp
☐	Virtual Machine Management	16509/tcp	
☐	Virtual Machine Management (TLS)	16514/tcp	
✓	WWW (HTTP)	80/tcp	

Allow access to necessary services, only.

The firewall is enabled. (modified)

Figure 6-12 system-config-firewall window

6.8 Disk security

We discussed the security of data at the operating system level and how we can use system constructs like mandatory access control to increase the data security of your environment. We have also discussed the physical security of the computer hardware, to make sure that changes cannot be made that will compromise the integrity of the computing platform.

The importance of the physical devices on which data is kept, DASDs or *disk drives*, must not be underestimated when creating a secure environment. If disk files can be accessed by other means, creating a network and application environment that prevents unauthorized access is worthless.

6.8.1 Traditional mainframe environments

In the past, the isolation of mainframe computing peripherals (in terms of compatibility with other platforms) meant that there was little need to think about keeping mainframe disks isolated. Because nothing else could connect to mainframe disks, the disks were isolated *by design*. The on-disk data format was significantly different from other platforms, and even if those aspects were avoided, the “EBCDIC problem” still had to be overcome. Although none of these barriers were insurmountable, they were difficult enough to solve so that defeating them was infeasible.

Note: The notion of *feasibility* is one that appears quite often in discussions on security. For example, developers of public-key cryptography systems do not refer to their algorithms as being *impossible* to break, but rather that they are *computationally infeasible* to break. This means that the effort required to break the security exceeds the resources available (usually time and money) to break it.

6.8.2 Modern environments

Today, many of these barriers to data portability are being removed. Linux on System z is an ASCII implementation, so data does not require character set or code page conversion between platforms. Storage area networks (SANs) are becoming increasingly prevalent in data centers, so the same storage servers are being used for almost all computing platforms in an organization. Also, the amount of computing power available for data conversion continues to increase, so conversion of different on-disk formats (even created when needed) becomes feasible.

6.9 Protecting ECKD disk

Management of access to disk devices in the mainframe environment has traditionally been done in the input/output control program (IOCP), or the *hardware definition*. For the logical partitions to be used on a particular CPC, IOCP defines the physical resources that the LPAR will have access to. These resources include processors, memory (storage), network connections, and disk subsystems.

6.9.1 Shared-DASD configurations

When the number of disk devices in an environment was small, systems programmers took care to define only the devices that had to be visible to each LPAR. Then, at around the same time, two technological improvements occurred: the number of accessible devices became much larger, and the ability to share resources between LPARs became a reality.

Because systems administrators required the flexibility to change where applications ran in the system, many of the rules about definition of the hardware became relaxed. Now, a common approach is to define every LPAR with access to almost every resource in the environment, and trust the higher-level security layers (RACF, application security) to protect data.

Although this approach works well when all the operating systems share a common security model, care must be taken when introducing a different platform into the environment.

6.9.2 LPAR configuration for Linux workloads

Because Linux (and, very likely, z/VM) uses a different security model than z/OS, a recommendation is to be sure that DASDs, which are used for Linux data, are not part of the same shared-disk LPAR configuration as z/OS disks.

Note: We commonly hear z/OS administrators, when considering avenues for data exposure or loss, say "I make sure those Linux systems cannot see any of my z/OS disks, who knows what they might do!" Yet, they see no threat in giving z/OS access to all the Linux disks, even using z/OS to perform backups of Linux. An important point to realize is the opportunity for data to be leaked in *either* direction. If z/OS can see Linux disks, a z/OS utility could be used to image a Linux file system and recreate that file system somewhere else in the environment. The fact that z/OS security is *stronger* than Linux security might not be a barrier to z/OS being used as a vector for leakage of data from Linux.

If Linux disks are present to z/OS or other systems, ensuring that the security configuration of the z/OS systems protects the Linux volumes from being accessed by unauthorized z/OS tasks is essential.

The Linux Compatible Disk Layout

All current Linux distributions use the Compatible Disk Layout (CDL) format for storing data on ECKD volumes. CDL uses a volume label and other metadata that is readable by z/OS, so if a DASD that is formatted as CDL is brought online to z/OS, it can be recognized (and not used as scratch).

When a z/OS system accesses a CDL volume, z/OS sees a fixed-record-length data set representing each CDL partition on the volume. It is true that z/OS cannot mount these data sets or read their contents, but if they are not adequately protected, they can be read from the CDL volume and written to other media, or over the network, as the first stage of extracting their contents.

RACF rules to protect CDL volumes

The data set name that is written to the VTOC by the **fdasd** command has the following format:

```
LINUX.Vlabel.PART000n.NATIVE
```

Where *label* is the volume label written by **fdasd**, and *n* is the partition number (either 0, 1, or 2). A suggestion is that data set names be protected by appropriate ESM rules in z/OS to ensure they are not exposed.

Note: When you run Linux under z/VM and use z/VM minidisks, getting visibility of these volumes to z/OS is more difficult because, under normal circumstances, z/OS would see the DASD label that is written by z/VM and not the Linux label. However, although it is more difficult, it is not impossible (for example, a track-to-track copy offset by one cylinder), so you must still take precautions.

6.10 Protecting Fibre Channel Protocol (FCP) disks

The security configuration of Fibre Channel SANs is usually done differently from mainframe disk environments. As mentioned previously, mainly because of the homogeneity of the mainframe environment, a mainframe disk is generally configured somewhat openly. This situation arose because the systems attaching to a mainframe disk were generally all the same operating system, using the same security model, and configured in the hardware and software to provide consistent and secure access to disks.

SANs are usually much more tightly controlled within the network (and within the storage devices themselves), in part because of the lack of a uniform and consistent method to manage SAN access at the operating system or machine microcode level. Techniques such as zoning and LUN masking are essential for controlling access to SAN volumes, especially in a heterogeneous computing environment where devices from many manufacturers, using significantly different security architectures, all connect to the same fabric.

Observation

Many parallels exist between storage networking and *traditional* data networking. Think of the ESCON/FICON DASD configuration as an SNA network, where only known devices can attach and connect over paths that were hard-coded into the definition of the network. The idea of an *SNA firewall* almost makes no sense, because the network endpoints themselves (under the supervision of the VTAM® SSCP, the *traffic cop*) provide a level of control in the way the network operates. Within the network, devices do not have to restrict or filter access.

Contrast this idea with the Fibre Channel disk environment, which we imagine as a TCP/IP network to which almost any kind of device can attach and anything can (and does) communicate with anything else. TCP/IP firewalls are common in all but the smallest networks to prevent unauthorized connections and control the types of traffic flow. Devices within the network must be used to provide security and traffic control not provided consistently by the devices at the endpoints.

6.10.1 Using FBA emulation in z/VM

A growing number of installations are using FCP disk for their Linux on System z workloads. Many sites with traditional mainframe workloads are using ECKD disks for z/VM and Linux system disks, while taking advantage of the performance and flexibility of FCP disks for data and application storage in Linux.

First delivered with z/VM 5.1, FCP provides a capability known as *FBA emulation*. This capability allows storage LUNs in a SAN to be presented to z/VM as emulated fixed-block architecture (FBA) DASDs. These DASDs can then be used in exactly the same ways as traditional DASDs, including running your z/VM installation.

Note: To use SAN disks exclusively for your z/VM installation, your System z server must have the “IPL from SCSI” microcode feature.

FBA emulation is a powerful capability that gives system administrators the best of both worlds: the flexibility of FCP-based SAN disks, combined with the manageability and security of traditional mainframe disks.

The EDEVICE statement includes a new option called Equivalency Identifiers (EQID), which was introduced in z/VM 6.2 to be used with Single System Image (SSI) and Live Guest Relocation (LGR). This mechanism helps to create a unique identity to all devices within an SSI cluster. In order to set up your cluster, you need to define your CTC connections, Fibre Channel Protocol (FCP), Hipersockets and Open System Adapter (OSA) devices properly. The EQIDs (equivalency identifier) are mainly used to check whether a device is eligible or not for relocation.

Note: The EQID configuration has to be implemented in all z/VM cluster members.

The SET EDEVice command

FBA emulation devices are created with the CP SET EDEVice command. The command syntax is shown in Figure 6-13 on page 221.

```
SET EDEVICE

>>--Set--EDEVice--edev--.---------------.------------------------------------>
                        |            (1)|
                        |-EQid--eqid----|
                        '-NOEQid--------'

>--.-TYpe--FBA--ATTRibutes--.-1750-.--.------------------.--| Paths |-.-----><
   |                        |-2105-|  '-.-ADD----.--PATH-'            |
   |                        |-2107-|    '-DELete-'                    |
   |                        |-2145-|                                  |
   |                        |-XIV--|                                  |
   |                        '-SCSI-'                                  |
   '-CLEAR------------------------------------------------------------'

Paths:
   <----------------------------------------< (2)
|----FCP_DEVice--rdev--WWPN--wwpn--LUN--lun----------------------------------|

Notes:
(1)  Normally, system-generated EQIDs will be assigned to EDEVICES. The
     ability to manually assign EQIDs is provided as a means of assigning
     device EQIDs in the unexpected event that system generation of an EDEVICE
     EQID fails.

(2)  You can specify a maximum of 8 paths to the device.

Authorization

Privilege Class: B

Purpose

Use the SET EDEVICE command to define, modify, or clear the definition of an
emulated device that represents a real SCSI device.
```

Figure 6-13 SET EDEVICE command syntax

A similar construct is available in the z/VM file `SYSTEM CONFIG` to define FBA emulation devices at z/VM startup.

Defining emulated FBA disks

We defined one of our available LUNs to our z/VM LPAR using FBA emulation. Then, we defined a minidisk on the resulting emulated volume and assigned the minidisk to our Linux virtual machine.

The SCSIDISC utility provides helpful information for the Logical Unit Numbers (LUNs) and is a powerful tool that can be used to dynamically discover all SCSI disks, and their associated paths, accessible by a virtual FCP device. For more information, see *z/VM: CP Commands and Utilities Reference*.

First of all, we had to attach the FCP virtual devices to its own ID and ran the SCSIDISC utility to create a SCSI report as shown in Example 6-23 on page 222.

Example 6-23 Discovering all SCSI disks

```
Please choose a number corresponding to an FCP device, 'ALL' to select all FCP devices or 'QUIT'
    0) 0000B804     1) 0000B904
ALL
For virtual FCP device 0000B804
Please choose a number corresponding to a WWPN, 'ALL' to select all WWPNs or 'QUIT'
        0) 500507630500C74C                  1) 500507630508C74C
        2) C05076ECEA801070                  3) C05076ECEA8010F8
        4) C05076E5F9001D10
ALL
For virtual FCP device 0000B804 and WWPN 500507630500C74C
Please choose a number corresponding to a LUN, 'ALL' to select all LUNs or'QUIT'

        0) 4010400F00000000                  1) 4011400F00000000

ALL
For virtual FCP device 0000B804 and WWPN 500507630508C74C
Please choose a number corresponding to a LUN, 'ALL' to select all LUNs or'QUIT'

        0) 4010400F00000000                  1) 4011400F00000000

ALL
HCPRXS976I WWPN C05076ECEA801070 on virtual FCP device 0000B804 could not be opened
HCPRXS976I WWPN C05076ECEA801070 on virtual FCP device 0000B804 ignored
HCPRXS976I WWPN C05076ECEA8010F8 on virtual FCP device 0000B804 could not be opened
HCPRXS976I WWPN C05076ECEA8010F8 on virtual FCP device 0000B804 ignored
HCPRXS976I WWPN C05076E5F9001D10 on virtual FCP device 0000B804 could not be opened
HCPRXS976I WWPN C05076E5F9001D10 on virtual FCP device 0000B804 ignored
For virtual FCP device 0000B904
Please choose a number corresponding to a WWPN, 'ALL' to select all WWPNs or 'QUIT'
        0) C05076E5F9001D98                  1) 500507630500C74C
        2) 500507630508C74C                  3) C05076ECEA801070
4) C05076ECEA8010F8
ALL
HCPRXS976I WWPN C05076E5F9001D98 on virtual FCP device 0000B904 could not be opened
HCPRXS976I WWPN C05076E5F9001D98 on virtual FCP device 0000B904 ignored
For virtual FCP device 0000B904 and WWPN 500507630500C74C
Please choose a number corresponding to a LUN, 'ALL' to select all LUNs or'QUIT'

        0) 4010400F00000000                  1) 4011400F00000000

ALL
For virtual FCP device 0000B904 and WWPN 500507630508C74C
Please choose a number corresponding to a LUN, 'ALL' to select all LUNs or'QUIT'

        0) 4010400F00000000                  1) 4011400F00000000
ALL
HCPRXS976I WWPN C05076ECEA801070 on virtual FCP device 0000B904 could not be opened
HCPRXS976I WWPN C05076ECEA801070 on virtual FCP device 0000B904 ignored
HCPRXS976I WWPN C05076ECEA8010F8 on virtual FCP device 0000B904 could not be opened
HCPRXS976I WWPN C05076ECEA8010F8 on virtual FCP device 0000B904 ignored
Storage area network analysis complete
Ready; T=24.27/24.29 10:20:44
```

The HCPRXS976I messages are informational and can be ignored—we created a unique zoning and included all WWPNs. We were setting the FBA device in ITSOSSI5 LPAR and C05076ECEA801070 and C05076ECEA8010F8 are the WWPNs for the other z/VM LPAR (ITSOSSI6), as shown in Figure 6-14.

```
500507630500C74C -> WWPN for DS8300 (S/N L300) storage
500507630508C74C -> WWPN for DS8300 (S/N L300) storage
c05076e5f9001d10 -> WWPN for b804 in ITSOSSI5 LPAR
c05076e5f9001d98 -> WWPN for b904 in ITSOSSI5 LPAR
c05076ecea801070 -> WWPN for b804 in ITSOSSI6 LPAR
c05076ecea8010f8 -> WWPN for b904 in ITSOSSI6 LPAR
```

Figure 6-14 Zone E - Used by z/VM FBA Emulation (EDEV)

To prevent those messages of showing up during the LUN discovery, you need to update the SAN zoning to create separated zones for each host instead of only one zone including all hosts, as shown in Figure 6-15.

```
Zone G - Used by z/VM FBA Emulation (EDEV) for ITSOSSI5
500507630500C74C -> WWPN for DS8300 (S/N L300) storage
500507630508C74C -> WWPN for DS8300 (S/N L300) storage
c05076e5f9001d10 -> WWPN for b804 in ITSOSSI5 LPAR
c05076e5f9001d98 -> WWPN for b904 in ITSOSSI5 LPAR

Zone H - Used by z/VM FBA Emulation (EDEV) for ITSOSSI6
500507630500C74C -> WWPN for DS8300 (S/N L300) storage
500507630508C74C -> WWPN for DS8300 (S/N L300) storage
c05076ecea801070 -> WWPN for b804 in ITSOSSI6 LPAR
c05076ecea8010f8 -> WWPN for b904 in ITSOSSI6 LPAR
```

Figure 6-15 Separated zones

The SCSI report is saved to SCSIDISC OUT, as shown in Figure 6-16.

```
00001 SCSIDISC LEVEL 0002
00002 _FCP_CH_ ______WWPN______ _____LUN_ID_____ _____________________________
00003 0000B804 500507630500C74C 4010400F00000000
00004 0000B804 500507630500C74C 4011400F00000000
00005 0000B804 500507630508C74C 4010400F00000000
00006 0000B804 500507630508C74C 4011400F00000000
00007 0000B804 C05076ECEA801070 ---------------- -----------------------------
00008 0000B804 C05076ECEA8010F8 ---------------- -----------------------------
00009 0000B804 C05076E5F9001D10 ---------------- -----------------------------
00010 0000B904 C05076E5F9001D98 ---------------- -----------------------------
00011 0000B904 500507630500C74C 4010400F00000000
00012 0000B904 500507630500C74C 4011400F00000000
00013 0000B904 500507630508C74C 4010400F00000000
00014 0000B904 500507630508C74C 4011400F00000000
00015 0000B904 C05076ECEA801070 ---------------- -----------------------------
00016 0000B904 C05076ECEA8010F8 ---------------- -----------------------------
```

Figure 6-16 Output of the SCSIDISC.OUT file

We used the commands listed in Example 6-24 to add our FBA emulation devices to CP. Note that the output of the SCSIDISC command showed two LUNs allocated; therefore, we created the bc00 and bc01 FBA devices.

Example 6-24 Commands adding an FBA emulation device to CP

```
det b804 b904
FCP B804 DETACHED
FCP B904 DETACHED
Ready; T=0.01/0.01 10:28:03

vary offline b804 b904
B804 varied offline
B904 varied offline
2 device(s) specified; 2 device(s) successfully varied offline

set edev bc00 type fba attr 2107 fcp_dev b804 wwpn 500507630500C74C lun
4010400F00000000
EDEV BC00 was created.
Ready; T=0.01/0.01 11:52:10
set edev bc00 type fba attr 2107 add path fcp_dev b804 wwpn 500507630508C74C lun
4010400F00000000
EDEV BC00 was modified.
Ready; T=0.01/0.01 11:52:19
set edev bc00 type fba attr 2107 add path fcp_dev b904 wwpn 500507630500C74C lun
4010400F00000000
EDEV BC00 was modified.
Ready; T=0.01/0.01 11:52:32
set edev bc00 type fba attr 2107 add path fcp_dev b904 wwpn 500507630508C74C lun
4010400F00000000
EDEV BC00 was modified.
Ready; T=0.01/0.01 11:53:24


set edev bc01 type fba attr 2107 fcp_dev b804 wwpn 500507630500C74C lun
4011400F00000000
EDEV BC01 was created.
Ready; T=0.01/0.01 14:02:40
set edev bc01 type fba attr 2107 add path fcp_dev b804 wwpn 500507630508C74C lun
4011400F00000000
EDEV BC01 was modified.
Ready; T=0.01/0.01 14:03:23
set edev bc01 type fba attr 2107 add path fcp_dev b904 wwpn 500507630500C74C lun
4011400F00000000
EDEV BC01 was modified.
Ready; T=0.01/0.01 14:03:43
set edev bc01 type fba attr 2107 add path fcp_dev b904 wwpn 500507630508C74C lun
4011400F00000000
EDEV BC01 was modified.
Ready; T=0.01/0.01 14:04:07
```

Bringing the emulated FBA device online

The paths to the disk are validated when the emulated device is brought online. We issued the VARY ONLINE command to make the disk accessible, as shown in Example 6-25 on page 225.

Example 6-25 Bringing an FCP EDEV online to z/VM

```
vary online b804 b904
B804 varied online
B904 varied online
2 device(s) specified; 2 device(s) successfully varied online

vary online bc00
BC00 varied online
1 device(s) specified; 1 device(s) successfully varied online
Ready; T=0.01/0.01 11:55:03

vary online bc01
BC01 varied online
1 device(s) specified; 1 device(s) successfully varied online
Ready; T=0.01/0.01 14:06:12
```

If a problem exists with any of the defined paths, CP generates an error message. Example 6-26 shows a situation we had with one of our emulated FBA devices when we were getting the correct SAN zoning defined.

Example 6-26 FCP EDEV errors with defined paths

```
vary online bc00
14:25:18 HCPSZP8701I Path FCP_DEV B904 WWPN 500507630508C74C LUN 4011400F00000000
0 was deleted from EDEV BC00 because it is invalid.
14:25:18 BC00 varied online
14:25:18 1 device(s) specified; 1 device(s) successfully varied online
```

This error happened when we added the second path to the LUN but did not have the SAN zoning defined correctly for that path. CP reports the failed path and removes it from the EDEV definition, but because one path is still available, the device can still be brought online.

Querying the status of emulated FBA disks

The **`query edevice`** command allows us to get information about emulated devices in the system configuration. Example 6-27 shows the QUERY EDEVice DETAILS display for our two FBA emulation devices.

Example 6-27 QUERY EDEVice DETAILS

```
q edev bc00 details
EDEV BC00 TYPE FBA ATTRIBUTES 2107
  VENDOR: IBM PRODUCT: 2107900 REVISION: .120
  BLOCKSIZE: 512 NUMBER OF BLOCKS: 20971520
  PATHS:
    FCP_DEV: B804 WWPN: 500507630500C74C LUN: 4010400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B804 WWPN: 500507630508C74C LUN: 4010400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B904 WWPN: 500507630500C74C LUN: 4010400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B904 WWPN: 500507630508C74C LUN: 4010400F00000000
      CONNECTION TYPE: SWITCHED
  EQID: 6005076305FFC74C00000000000010F0C600000000013FFFFF
Ready; T=0.01/0.01 14:29:18
```

```
q edev bc01 details
EDEV BC01 TYPE FBA ATTRIBUTES 2107
  VENDOR: IBM PRODUCT: 2107900 REVISION: .120
  BLOCKSIZE: 512 NUMBER OF BLOCKS: 20971520
  PATHS:
    FCP_DEV: B804 WWPN: 500507630500C74C LUN: 4011400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B804 WWPN: 500507630508C74C LUN: 4011400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B904 WWPN: 500507630500C74C LUN: 4011400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B904 WWPN: 500507630508C74C LUN: 4011400F00000000
      CONNECTION TYPE: SWITCHED
  EQID: 6005076305FFC74C00000000000011F0C600000000013FFFFF
Ready; T=0.01/0.01 14:29:23
```

If you noticed in Example 6-27, the **query edevice** command listed all the properties of the FBA devices including the EQIDs that were automatically generated by CP.

We repeated the FBA configuration change for our other z/VM LPAR (ITSOSSI6), then brought both EDEVICEs online.

Installing z/VM to FCP disks using FBA emulation

To verify that installing z/VM to FCP disks is equivalent to installing on ECKD[1], we obtained an LPAR with FCP interfaces and the ability to attach to our SAN LUNs. We followed the process described in Chapter 5 of *z/VM: Guide to Automated Installation and Service*, (for V5R4), GC24-6099-05:

1. We used an FTP server as the source of all the installation files, including using the FTP server as the target for the HMC “Load from CD-ROM, DVD or Server” function.
2. When we reached task 4b of step 3, “Verify the Volumes Needed for Installation are Available”, we issued the appropriate **set edevice** commands to configure our SCSI disks. Example 6-28 shows the commands and the responses we received.

Example 6-28 SET EDEVICE commands and responses during z/VM install to SCSI disks

```
set edev bc00 type fba attr 2105 fcp_dev b800 wwpn 5005076300cc9589 lun 52120000
00000000
09:57:08 EDEV BC00 was created.
Ready; T=0.01/0.01 09:57:08
set edev bc01 type fba attr 2105 fcp_dev b800 wwpn 5005076300cc9589 lun 52130000
00000000
09:57:53 EDEV BC01 was created.
Ready; T=0.01/0.01 09:57:53
set edev bc02 type fba attr 2105 fcp_dev b800 wwpn 5005076300cc9589 lun 52160000
00000000
09:58:06 EDEV BC02 was created.
Ready; T=0.01/0.01 09:58:06
set edev bc03 type fba attr 2105 fcp_dev b800 wwpn 5005076300cc9589 lun 52170000
00000000
09:58:19 EDEV BC03 was created.
Ready; T=0.01/0.01 09:58:19
```

[1] By “equivalent” here we mean providing similar levels of security and isolation for virtual machines. Many operational differences exist between ECKD and FCP-attached-SCSI disks and that are separate concerns.

```
set edev bc00 type fba attr 2105 add path fcp_dev b900 wwpn 5005076300c89589 lun
5212000000000000
09:58:50 EDEV BC00 was modified.
Ready; T=0.01/0.01 09:58:50
set edev bc01 type fba attr 2105 add path fcp_dev b900 wwpn 5005076300c89589 lun
5213000000000000
09:59:08 EDEV BC01 was modified.
Ready; T=0.01/0.01 09:59:08
set edev bc02 type fba attr 2105 add path fcp_dev b900 wwpn 5005076300c89589 lun
5216000000000000
09:59:22 EDEV BC02 was modified.
Ready; T=0.01/0.01 09:59:22
set edev bc03 type fba attr 2105 add path fcp_dev b900 wwpn 5005076300c89589 lun
5217000000000000
09:59:39 EDEV BC03 was modified.
Ready; T=0.01/0.01 09:59:39
                                                                RUNNING IBMVMRAM
```

Note: Setting up the second path to each of the SCSI disks was not necessary to complete the installation. We mainly did it to prove that the function works during installation in the same way it does on a normal running system.

3. The next task in the installation, task 5 of step 3, asked us to vary on the EDEVices we just created. Example 6-29 shows the result of the **vary online** command we issued.

Example 6-29 Issuing vary online for the EDEVices

```
vary online bc00-bc03
10:03:21 BC00 varied online
10:03:21 BC01 varied online
10:03:21 BC02 varied online
10:03:21 BC03 varied online
10:03:21 4 device(s) specified; 4 device(s) successfully varied online
Ready;  0.01/0.01 10:03:21
                                                                RUNNING IBMVMRAM
```

From this point, the installation was almost the same as an installation to ECKD DASD.

6.10.2 Using N_Port ID Virtualization

When we first set up our FCP configuration, N_Port ID Virtualization (NPIV) was not in use. We activated NPIV in our *SCSI test* LPAR.

Activating FCP ports for NPIV

We used the following process to enable NPIV:

1. We logged on to the HMC and selected under Systems the system that is hosting our z/VM LPAR, as shown in Figure 6-17 on page 228.

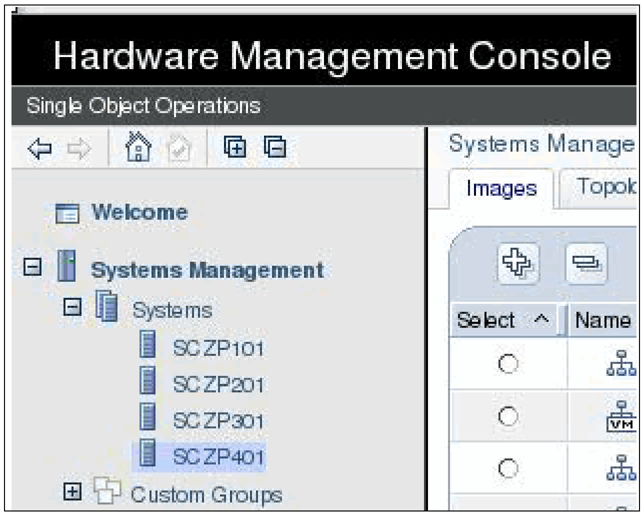


Figure 6-17 Accessing SCZP401 System

2. In the right pane, select **Single Object Operations** under Recovery, as shown in Figure 6-18.

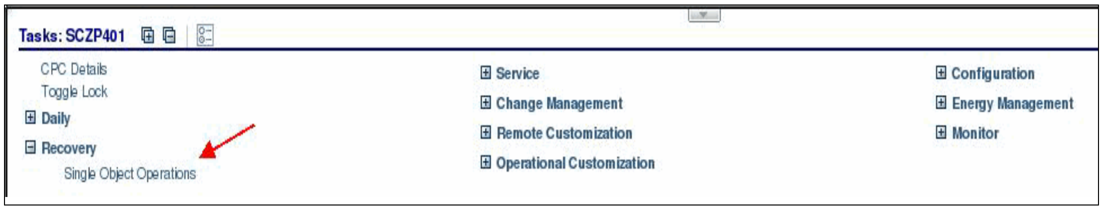
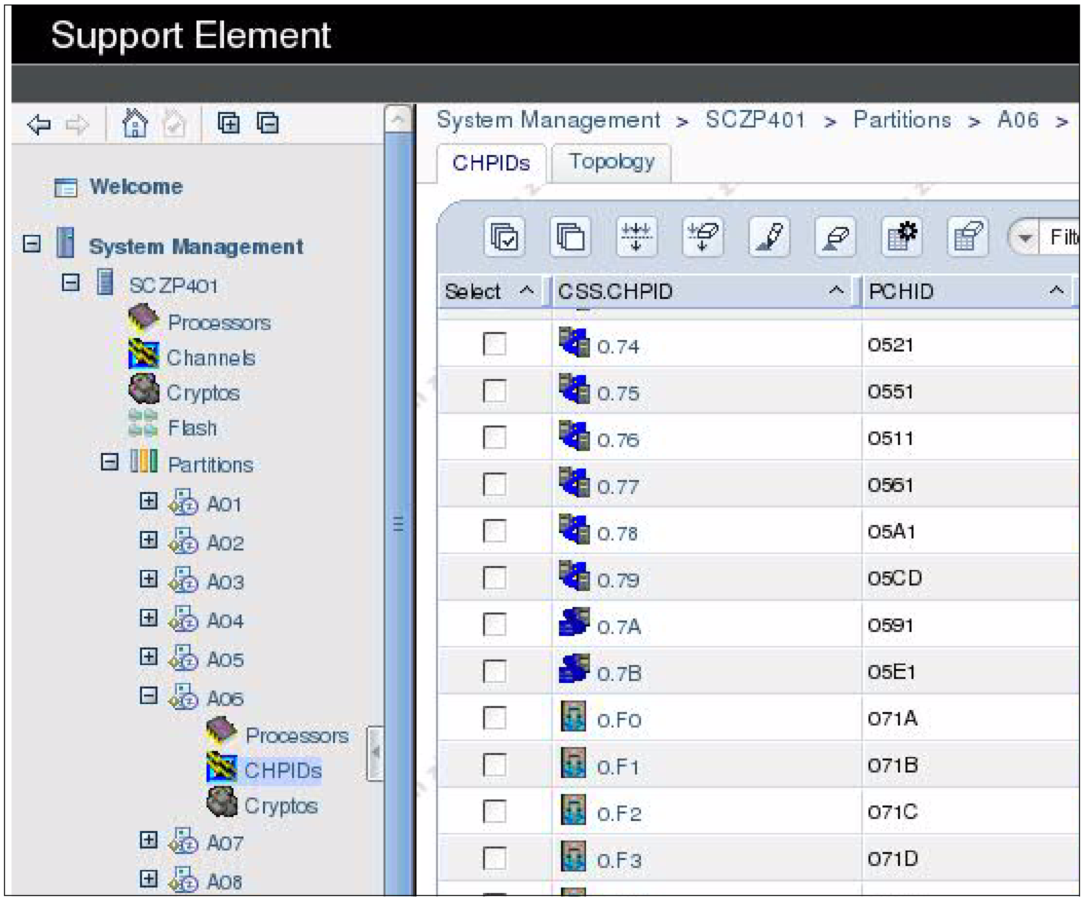


Figure 6-18 Accessing Single Object Operations

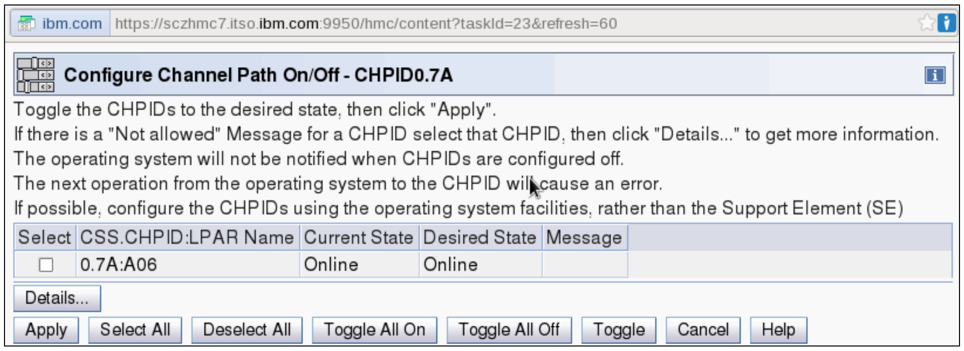
3. In the Support Element window, we went to **System Management** → **Partitions,** located our LPAR and selected **CHPIDs**, which showed the CHPIDs in the right panel, as shown in Figure 6-19 on page 229.



Select	CSS.CHPID	PCHID
☐	0.74	0521
☐	0.75	0551
☐	0.76	0511
☐	0.77	0561
☐	0.78	05A1
☐	0.79	05CD
☐	0.7A	0591
☐	0.7B	05E1
☐	0.F0	071A
☐	0.F1	071B
☐	0.F2	071C
☐	0.F3	071D

Figure 6-19 Listing CHPIDs

4. We scrolled to find our FCP CHPID. We selected it and the "View popup menu" appeared, then selected **CHPID Operations** → **Configure On/Off**.
5. The panel to perform the CHPID configuration was displayed. We selected the CHPID in the CHPIDs tab to take offline, and clicked **Toggle**, as shown in Figure 6-20.


ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/content?taskId=23&refresh=60

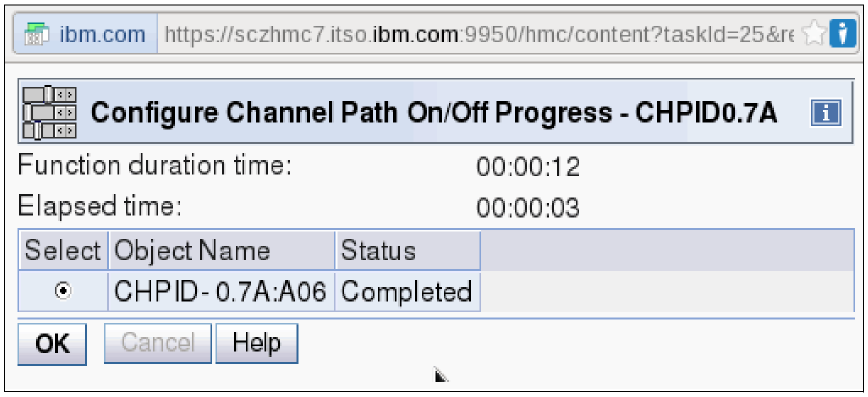
Configure Channel Path On/Off - CHPID0.7A

Toggle the CHPIDs to the desired state, then click "Apply".
If there is a "Not allowed" Message for a CHPID select that CHPID, then click "Details..." to get more information.
The operating system will not be notified when CHPIDs are configured off.
The next operation from the operating system to the CHPID will cause an error.
If possible, configure the CHPIDs using the operating system facilities, rather than the Support Element (SE)

Select	CSS.CHPID:LPAR Name	Current State	Desired State	Message
☐	0.7A:A06	Online	Online	

Figure 6-20 Configuring the FCP CHPID offline (2)

6. We selected the CHPID, **toggled** the CHPIDs to the desired state, then clicked **Apply**. HMC asked for our password. The progress dialog opened, indicating that the operation was completed. See Figure 6-21 on page 230.



ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/content?taskId=25&re

Configure Channel Path On/Off Progress - CHPID0.7A

Function duration time: 00:00:12

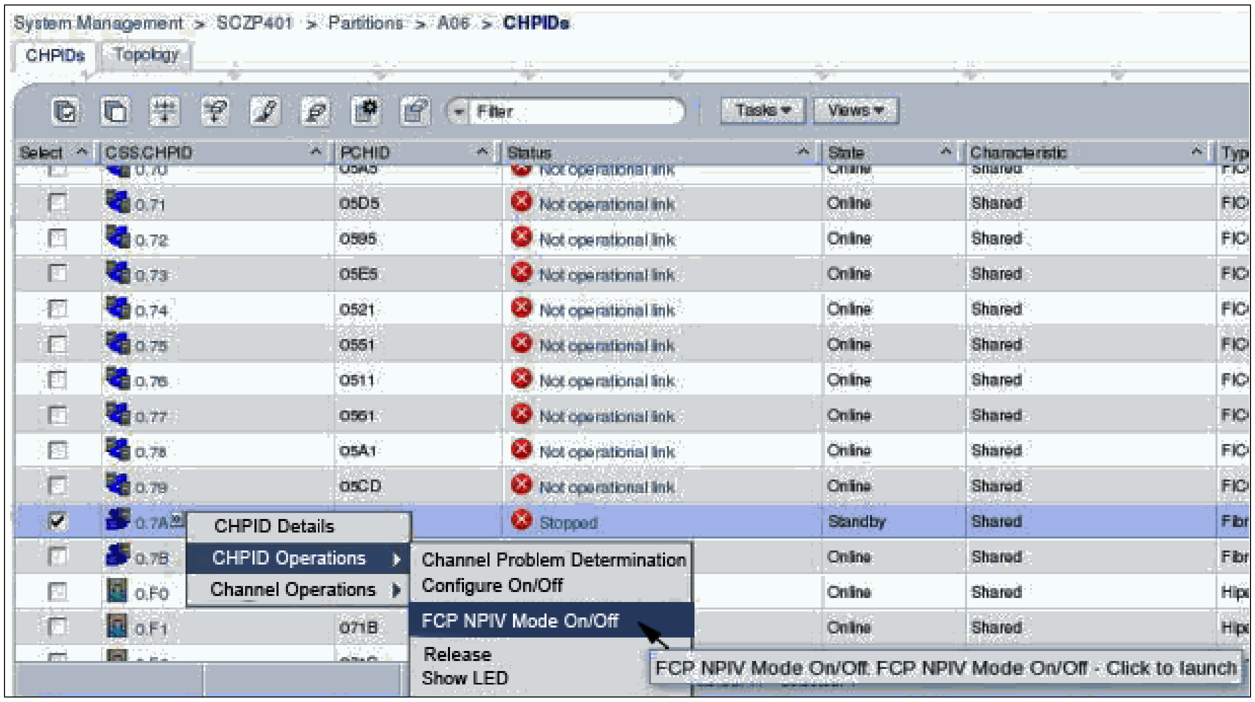
Elapsed time: 00:00:03

Select	Object Name	Status
●	CHPID - 0.7A:A06	Completed

OK | Cancel | Help

Figure 6-21 Configuring the FCP CHPID offline (3)

7. We clicked **OK**, then repeated the process for the other CHPID that had to be brought offline.
8. When the CHPIDs were offline, we could then change their NPIV state. We again selected the **CHPID** under the CHPID tab, but instead, we selected **CHPID Operations** → **FCP NPIV Mode On/Off**, as shown in Figure 6-22.



System Management > SCZP401 > Partitions > A06 > CHPIDs

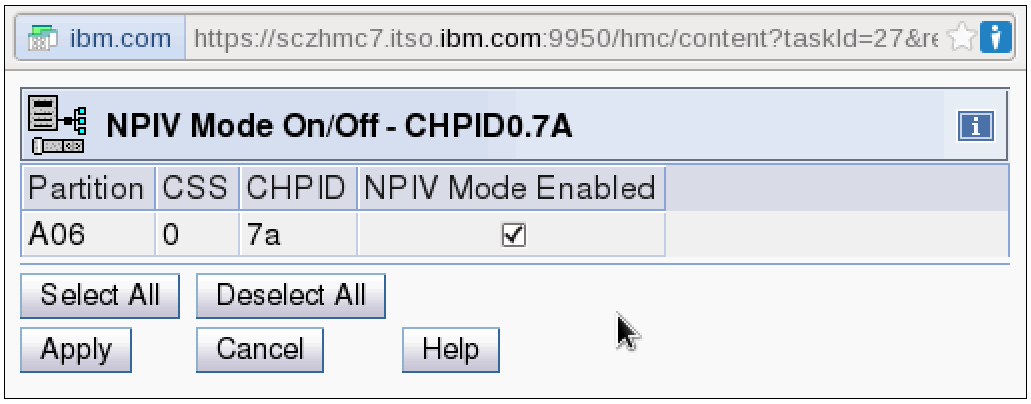
CHPIDs | Topology

Filter | Tasks | Views

Select	CSS.CHPID	PCHID	Status	State	Characteristic	Typ
☐	0.70	05A5	Not operational link	Online	Shared	FIC
☐	0.71	05D5	Not operational link	Online	Shared	FIC
☐	0.72	0595	Not operational link	Online	Shared	FIC
☐	0.73	05E5	Not operational link	Online	Shared	FIC
☐	0.74	0521	Not operational link	Online	Shared	FIC
☐	0.75	0551	Not operational link	Online	Shared	FIC
☐	0.76	0511	Not operational link	Online	Shared	FIC
☐	0.77	0561	Not operational link	Online	Shared	FIC
☐	0.78	05A1	Not operational link	Online	Shared	FIC
☐	0.79	05CD	Not operational link	Online	Shared	FIC
☑	0.7A		Stopped	Standby	Shared	Fibr
☐	0.7B			Online	Shared	Fibr
☐	0.F0			Online	Shared	Hip
☐	0.F1	071B		Online	Shared	Hip

Figure 6-22 Selecting CHPID Operations FCP NPIV Mode On/Off

9. The NPIV Mode On/Off function was displayed. We selected the **NPIV Mode Enabled** check box, as shown in Figure 6-23 on page 231, and clicked **Apply**.

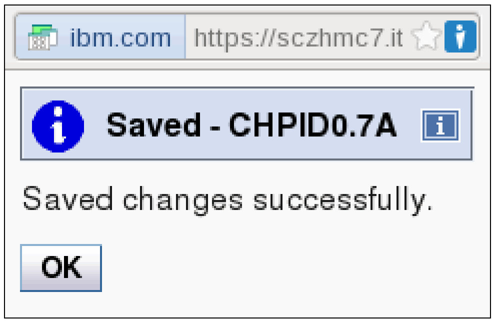


NPIV Mode On/Off - CHPID0.7A

Partition	CSS	CHPID	NPIV Mode Enabled
A06	0	7a	✓

Figure 6-23 FCP NPIV Mode On/Off

10. After a moment, the confirmation box opened, shown in Figure 6-24, indicating that the change was successful.



Saved - CHPID0.7A

Saved changes successfully.

Figure 6-24 NPIV Mode confirmation box

11. We repeated the NPIV configuration change for our other FCP CHPID, then brought both CHPIDs back online to our LPAR.

At that time, we were curious about whether all LPARs with access to the SAN through the same FCP ports had to enable NPIV when one LPAR was NPIV-enabled[1]. Our *main* z/VM system had one LUN from the SAN configured, and we found that even after the emulated device was varied offline and back online, all paths to it were still available. Therefore, NPIV configuration does not have to be enabled for all LPARs sharing an FCP port.

Note: Something to keep in mind from a security aspect is that if you have multiple systems that are accessing SAN through the same FCP ports, they might not all have the same security configuration.

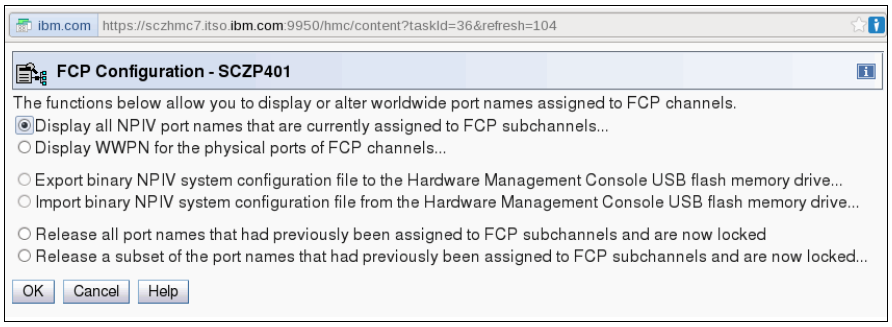
Storage configuration and SAN zoning

For proper access to the disk LUNs, both the disk subsystem and the SAN switch had to be properly configured for the NPIV worldwide port names (WWPNs). To determine what the assigned NPIV WWPNs would be, we again accessed Single Object Operations through the System z HMC.

[1] *Introducing N_Port Identifier Virtualization for IBM System z9*, REDP-4125 states that this is expected, but we still wanted to try it for ourselves!

We performed these steps:

1. We clicked in our system (SCZP401) under System Management in the left panel, then expanded **CPC Configuration** and selected **FCP Configuration**.
2. This operation displayed the FCP NPIV Port Names configuration selection panel, shown in Figure 6-25. The default action, "Display all NPIV port names that are currently assigned to FCP subchannels," is the operation we used and selected **OK**.



ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/content?taskId=36&refresh=104

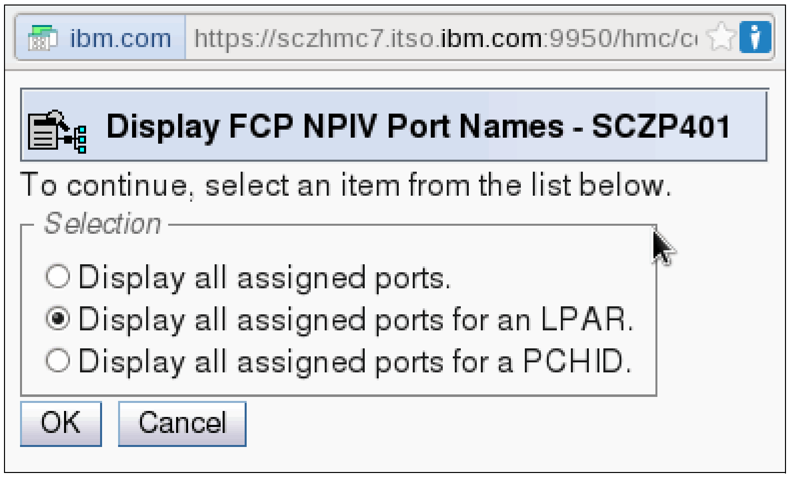
FCP Configuration - SCZP401

The functions below allow you to display or alter worldwide port names assigned to FCP channels.

- (●) Display all NPIV port names that are currently assigned to FCP subchannels...
- ( ) Display WWPN for the physical ports of FCP channels...
- ( ) Export binary NPIV system configuration file to the Hardware Management Console USB flash memory drive...
- ( ) Import binary NPIV system configuration file from the Hardware Management Console USB flash memory drive...
- ( ) Release all port names that had previously been assigned to FCP subchannels and are now locked
- ( ) Release a subset of the port names that had previously been assigned to FCP subchannels and are now locked...

Figure 6-25 NPIV Configuration selection panel

3. Another panel opened, shown in Figure 6-26, which allowed us to select the range of port names to be displayed. We selected **Display all assigned ports for an LPAR**, and then clicked **OK**.



ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/c

Display FCP NPIV Port Names - SCZP401

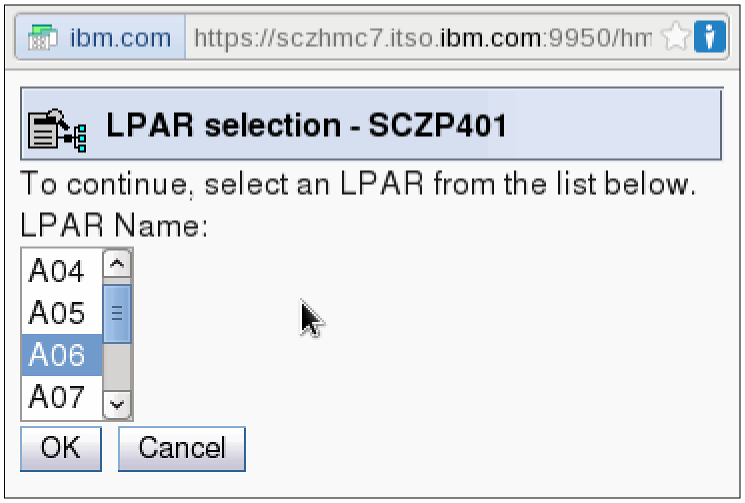
To continue, select an item from the list below.

Selection

- ( ) Display all assigned ports.
- (●) Display all assigned ports for an LPAR.
- ( ) Display all assigned ports for a PCHID.

Figure 6-26 Port name range selection panel

4. The panel shown in Figure 6-27 on page 233 opened, which allowed us to select the LPAR for which the assigned names would be displayed. We selected the name of our LPAR from the list, and clicked **OK**.



ibm.com https://sczhmc7.itso.ibm.com:9950/hm

LPAR selection - SCZP401

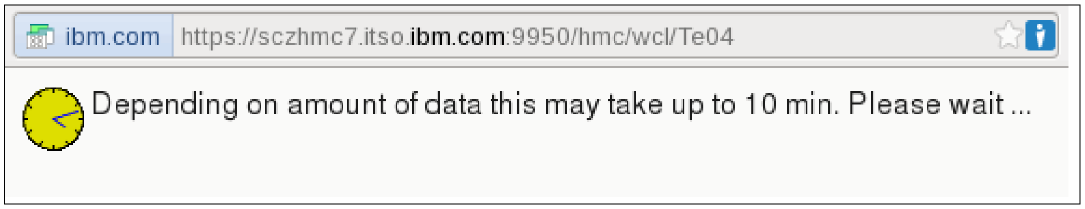
To continue, select an LPAR from the list below.

LPAR Name:

A04
A05
A06
A07

Figure 6-27 NPIV Port Names LPAR selection panel

The progress panel opened, shown in Figure 6-28.

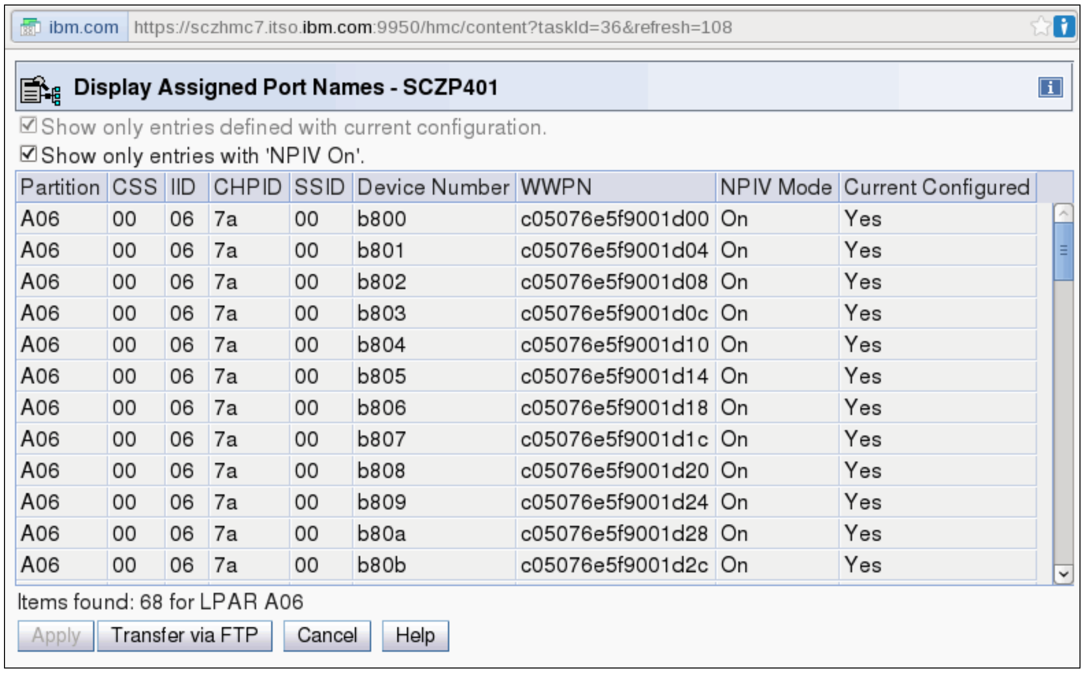


ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/wcl/Te04

Depending on amount of data this may take up to 10 min. Please wait ...

Figure 6-28 Progress panel for NPIV Port Name display

After several seconds, the Display Assigned Port Names list panel opened, shown in Figure 6-29.



ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/content?taskId=36&refresh=108

Display Assigned Port Names - SCZP401

☑ Show only entries defined with current configuration.

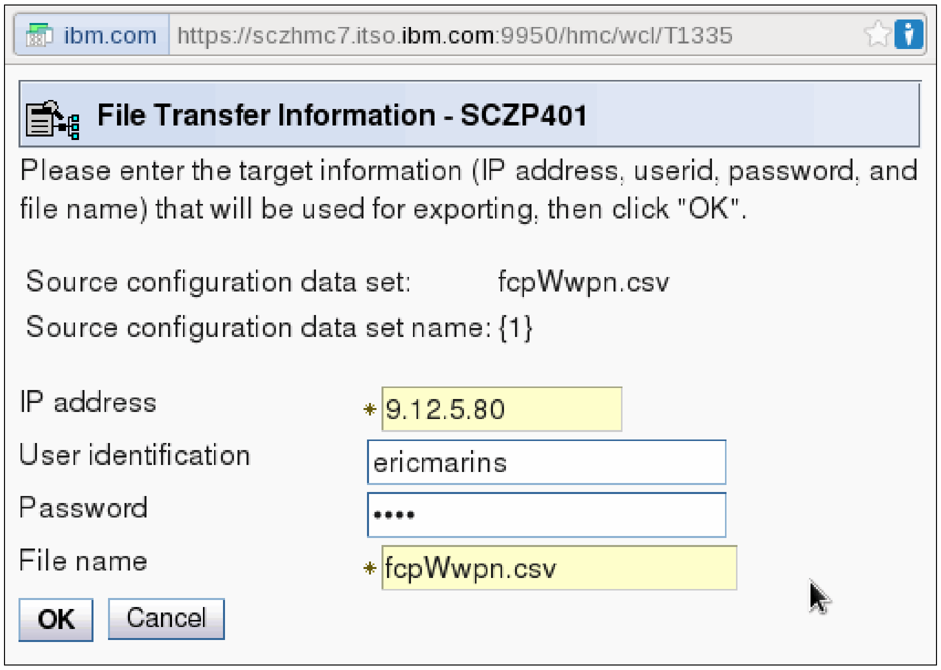
☑ Show only entries with 'NPIV On'.

Partition	CSS	IID	CHPID	SSID	Device Number	WWPN	NPIV Mode	Current Configured
A06	00	06	7a	00	b800	c05076e5f9001d00	On	Yes
A06	00	06	7a	00	b801	c05076e5f9001d04	On	Yes
A06	00	06	7a	00	b802	c05076e5f9001d08	On	Yes
A06	00	06	7a	00	b803	c05076e5f9001d0c	On	Yes
A06	00	06	7a	00	b804	c05076e5f9001d10	On	Yes
A06	00	06	7a	00	b805	c05076e5f9001d14	On	Yes
A06	00	06	7a	00	b806	c05076e5f9001d18	On	Yes
A06	00	06	7a	00	b807	c05076e5f9001d1c	On	Yes
A06	00	06	7a	00	b808	c05076e5f9001d20	On	Yes
A06	00	06	7a	00	b809	c05076e5f9001d24	On	Yes
A06	00	06	7a	00	b80a	c05076e5f9001d28	On	Yes
A06	00	06	7a	00	b80b	c05076e5f9001d2c	On	Yes

Items found: 68 for LPAR A06

Figure 6-29 NPIV Display Assigned Port Names panel

5. To make our job a little easier, we decided to use the “Transfer via FTP” function to send the file to our Linux server (for printing, or transfer to our PC). We clicked **Transfer via FTP**. In the next dialog that opened, we entered details of an FTP server to send the information to. See Figure 6-30.


ibm.com https://sczhmc7.itso.ibm.com:9950/hmc/wcl/T1335

File Transfer Information - SCZP401

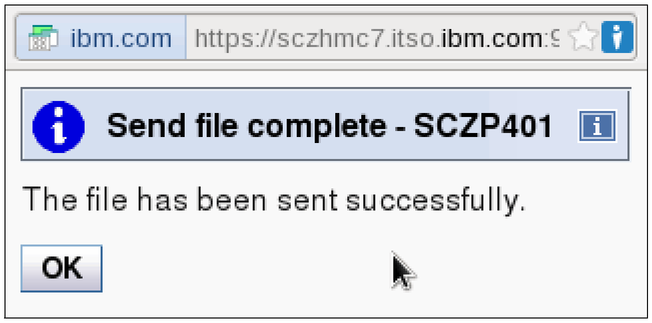
Please enter the target information (IP address, userid, password, and file name) that will be used for exporting, then click "OK".

Source configuration data set: fcpWwpn.csv
Source configuration data set name: {1}

IP address * 9.12.5.80
User identification ericmarins
Password ••••
File name * fcpWwpn.csv

Figure 6-30 NPIV File Transfer Information panel

6. When the FTP details were complete, we clicked **OK**, and after a little while we received the output shown in Figure 6-31.


ibm.com https://sczhmc7.itso.ibm.com:9

Send file complete - SCZP401

The file has been sent successfully.

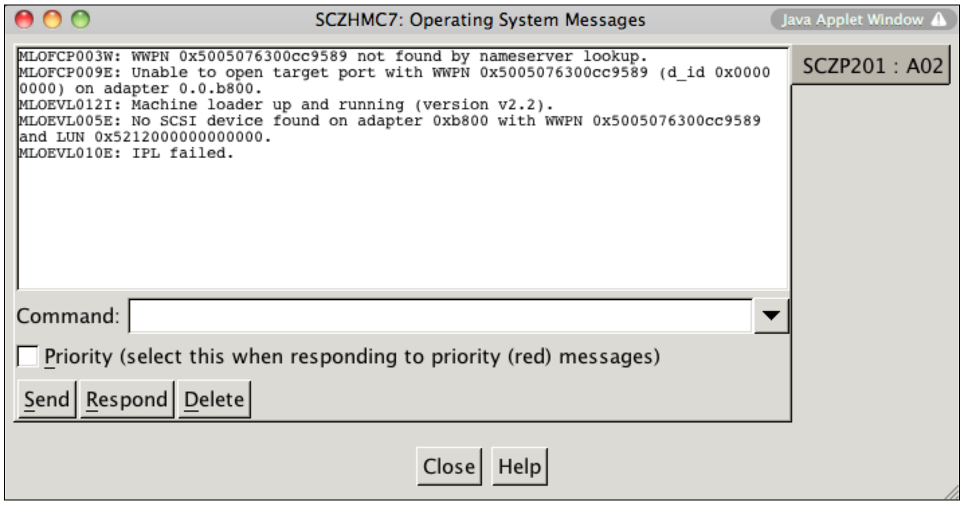
Figure 6-31 FTP file transfer confirmation

7. To verify the transferred file, we displayed it by using the `less` program on our Linux server; see Example 6-30 on page 235.

Example 6-30 Transferred NPIV port name report (portion, displayed in the less program

```
## Version: 1.0
## Machine serial number: 00002000B8D7
## Current configuration filter enabled: Yes
## NPIV ON filter enabled: Yes
## Items for LPAR: A06
## partitionName,cssId,iid,chpidId,ssId,deviceNumber,wwpn,npiv mode,current
configured,pchid,phys. wwpn,
A06,00,06,7a,00,b800,c05076e5f9001d00,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b801,c05076e5f9001d04,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b802,c05076e5f9001d08,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b803,c05076e5f9001d0c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b804,c05076e5f9001d10,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b805,c05076e5f9001d14,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b806,c05076e5f9001d18,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b807,c05076e5f9001d1c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b808,c05076e5f9001d20,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b809,c05076e5f9001d24,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b80a,c05076e5f9001d28,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b80b,c05076e5f9001d2c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b80c,c05076e5f9001d30,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b80d,c05076e5f9001d34,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b80e,c05076e5f9001d38,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b80f,c05076e5f9001d3c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b810,c05076e5f9001d40,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b811,c05076e5f9001d44,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b812,c05076e5f9001d48,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b813,c05076e5f9001d4c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b814,c05076e5f9001d50,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b815,c05076e5f9001d54,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b816,c05076e5f9001d58,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b817,c05076e5f9001d5c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b818,c05076e5f9001d60,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b819,c05076e5f9001d64,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b81a,c05076e5f9001d68,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b81b,c05076e5f9001d6c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b81c,c05076e5f9001d70,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b81d,c05076e5f9001d74,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b81e,c05076e5f9001d78,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b81f,c05076e5f9001d7c,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b8fc,c05076e5f9001d80,On,Yes,0591,c05076e5f9005911
A06,00,06,7a,00,b8fd,c05076e5f9001d84,On,Yes,0591,c05076e5f9005911
A06,00,06,7b,00,b900,c05076e5f9001d88,On,Yes,05e1,c05076e5f9005e11
A06,00,06,7b,00,b901,c05076e5f9001d8c,On,Yes,05e1,c05076e5f9005e11
A06,00,06,7b,00,b902,c05076e5f9001d90,On,Yes,05e1,c05076e5f9005e11
A06,00,06,7b,00,b903,c05076e5f9001d94,On,Yes,05e1,c05076e5f9005e11
A06,00,06,7b,00,b904,c05076e5f9001d98,On,Yes,05e1,c05076e5f9005e11
A06,00,06,7b,00,b905,c05076e5f9001d9c,On,Yes,05e1,c05076e5f9005e11
```

At this point we tried to IPL our z/VM system and as we expected, the IPL was unsuccessful. We checked the Operating System Messages function on the HMC, and saw the errors shown in Figure 6-32 on page 236.

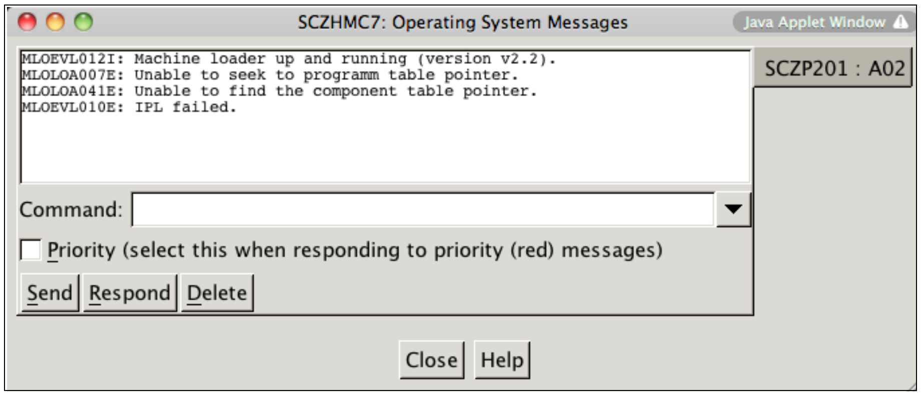

SCZHMC7: Operating System Messages
Java Applet Window
SCZP201 : A02
Command:
Priority (select this when responding to priority (red) messages)
Send
Respond
Delete
Close
Help

```
MLOFCP003W: WWPN 0x5005076300cc9589 not found by nameserver lookup.
MLOFCP009E: Unable to open target port with WWPN 0x5005076300cc9589 (d_id 0x0000
0000) on adapter 0.0.b800.
MLOEVL012I: Machine loader up and running (version v2.2).
MLOEVL005E: No SCSI device found on adapter 0xb800 with WWPN 0x5005076300cc9589
and LUN 0x5212000000000000.
MLOEVL010E: IPL failed.
```

Figure 6-32 IPL errors because of improper SAN zoning

The zoning in the SAN had not yet been updated to reflect the NPIV WWPN that z/VM was now using to access the SAN. Consequently, the SCSI IPL feature was unable to locate the WWPN to connect to the IPL LUN.

Our SAN administrator added the NPIV WWPN to the zone, and we retried the IPL. Once again, the IPL was unsuccessful, as seen in Figure 6-33.


SCZHMC7: Operating System Messages
Java Applet Window
SCZP201 : A02
Command:
Priority (select this when responding to priority (red) messages)
Send
Respond
Delete
Close
Help

```
MLOEVL012I: Machine loader up and running (version v2.2).
MLOLOA007E: Unable to seek to programm table pointer.
MLOLOA041E: Unable to find the component table pointer.
MLOEVL010E: IPL failed.
```

Figure 6-33 IPL error because of disk configuration

This time the failure was because of the IBM Enterprise Storage Server® configuration. The definitions for our LUNs had to be updated to allow the new NPIV WWPNs to connect to them. When this was done, we again retried our IPL, and this time it was successful.

Visibility of NPIV WWPNs to the SAN switch

We did encounter one unusual circumstance when attempting to zone the SAN for the NPIV WWPNs. Our SAN administrator was unable to find the NPIV port to add to the zone until after the first time the port (unsuccessfully) tried to access the fabric.

It also resulted in the second set of paths from our SCSI z/VM LPAR to the LUNs from not being present when we did our first IPL. Example 6-31 on page 237 shows error messages we received when we IPLed, because the NPIV WWPN on the second FCP port was not being added to the SAN zone.

Example 6-31 EDEVice errors because of incomplete SAN zoning

```
11:27:40 HCPSZP8701I Path FCP_DEV B900 WWPN 5005076300C89589 LU 510B000000000
000 was deleted from EDEV BC10 because it is invalid.
11:27:40 HCPSZP8701I Path FCP_DEV B900 WWPN 5005076300C89589 LU 5217000000000
000 was deleted from EDEV BC03 because it is invalid.
11:27:40 HCPSZP8701I Path FCP_DEV B900 WWPN 5005076300C89589 LU 5216000000000
000 was deleted from EDEV BC02 because it is invalid.
11:27:40 HCPSZP8701I Path FCP_DEV B900 WWPN 5005076300C89589 LU 5213000000000
000 was deleted from EDEV BC01 because it is invalid.
11:27:40 HCPSZP8701I Path FCP_DEV B900 WWPN 5005076300C89589 LU 5212000000000
000 was deleted from EDEV BC00 because it is invalid.
```

Getting the second path added to the zone was quite an effort. For reasons we were not able to determine, the only way that we could get a WWPN to be visible in our switch configuration utility was to attempt to IPL z/VM over it. We attached the desired FCP subchannel to MAINT and used the SET LOADDEV command to set the details of the IPL device, then IPLed the FCP device. The IPL was unsuccessful, but we could now see the WWPN in our SAN configurator and were able to add it to the zone.

6.11 Protecting z/VM minidisks

In this topic, we present aspects of protecting access to minidisks under z/VM. The underlying technology of the disk volumes themselves, either ECKD or emulated FBA, is not relevant to this part of the discussion; here, we are simply referring to the security of the minidisks within z/VM.

6.11.1 Minidisk access security

An essential point to know is that minidisks can only be accessed by the users they are intended for. If minidisk access is too lax, information can easily be leaked between virtual machines. This is especially significant if those virtual machines are supposed to be in different security zones; minidisks with insufficient access control can easily be used to move information from a high-security zone to a zone with lower security.

The method you use to provide minidisk access control largely depends on the overall level of security that is required for your installation. The basic level of minidisk access control, that is provided by CP with minidisk passwords in the user directory, is simple to use and provides a good degree of security.

Note: Minidisk passwords are described in “Protecting access to minidisks” on page 29.

As we discussed previously, passwords for minidisks are easily obtained by anyone who has access to the MAINT 2CC disk (or wherever you keep your USER DIRECT). Also, CP provides no auditing of minidisk accesses.

Note: Remember that a shared computing environment is only as secure as the weakest system in the environment. You might be tempted to have reduced security on a test or development system, but if that system connects to the same infrastructure as production systems, it *could* provide a way for secure data to be seen from an unsecured zone.

Using an ESM for controlling access to minidisks is the minimum requirement for some installations. An ESM provides greater granularity to access authorities, and extensive options for controlling auditing of both successful and unsuccessful accesses.

6.11.2 Overlapping minidisks

Minidisks are defined in the user directory based on the host volume, the start location, and the number of cylinders or blocks they cover. These definitions can be *overlapped*, meaning that more than one minidisk covering the same extent on disk may be defined. In general, defining minidisks that overlap is risky and insecure, but there are limited circumstances where it is useful or even required: MAINT's 123 minidisk is an example of an overlap minidisk that is required for z/VM system management purposes.

Note: Refer to *z/VM: Guide to Automated Installation and Service*, GC24-6099 for more information about why minidisks that overlap the system volumes are needed.

The z/VM administrators usually check that they have not created an overlapping minidisk at the time that they create a new user. Overlapping minidisks can be created after users have been defined, however, and administrators must regularly check whether such minidisks have been defined on their systems, to ensure secure operation. An overlapping minidisk definition provides a way for unauthorized access to system data to be obtained. Figure 6-34 shows two ways that this might occur.

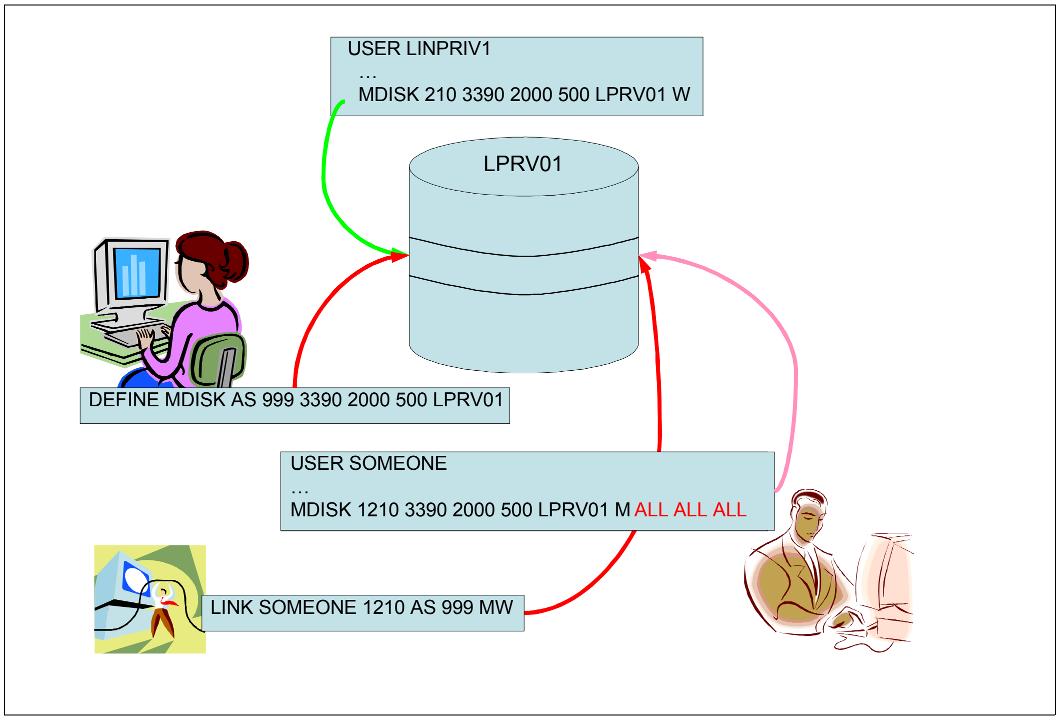

USER LINPRIV1
…
MDISK 210 3390 2000 500 LPRV01 W
LPRV01
DEFINE MDISK AS 999 3390 2000 500 LPRV01
USER SOMEONE
…
MDISK 1210 3390 2000 500 LPRV01 M ALL ALL ALL
LINK SOMEONE 1210 AS 999 MW

Figure 6-34 Overlapping minidisks causing data leakage

The minidisk is created and owned in the directory entry shown as the top of the diagram. The green arrow indicates the authorized way that the disk is accessed. System administrators defined the minidisk with no access passwords, meaning that the minidisk cannot be linked to in any mode.

The first scenario illustrates the simple case where a privileged user defines an overlapping minidisk to obtain access to a sensitive Linux file system (the red arrow on the left). When the minidisk is defined, the second virtual machine can access the file system even though no access passwords were defined for the original disk.

The second scenario is an extension of the first, and shows that even the best intentions can create a vector for leakage. A system administrator has created an overlapped minidisk for some legitimate purpose but defined the disk with weak or nonexistent access control. This approach allows a user from another virtual machine to create a link to that minidisk and bypass all access control on that minidisk.

Note: We cannot think of a legitimate circumstance where an overlap minidisk definition in a single z/VM LPAR would be useful in the context of Linux on System z; if concurrent access to a disk is required, a single user should define the minidisk and then other users LINK to it. Having said this, just because we cannot think of a legitimate use does not mean that such a legitimate use cannot exist (now or in the future).

A more likely scenario would be overlapping minidisks that are defined on different z/VM systems. This configuration might be used to support particular Linux systems being able to run on a number of different z/VM systems. This is discussed further in “Minidisk overlaps” on page 240.

With DirMaint in place, the definition of a minidisk that overlaps an existing minidisk will fail. However, this overlap checking can be circumvented by turning off extent checking with the DirMaint EXTNCHK OFF command. To prevent an administrator from turning off this extent checking, a DirMaint command class can be defined for your general administrators that does not include the EXTNCHK command.

Note: For more information about DirMaint command classes and altering administrator privilege, see *z/VM: Directory Maintenance Facility Tailoring and Administration*, SC24-6135.

If overlapping minidisks are defined, RACF might be able to protect against access depending on the way that it has been set up. First, the minidisk protection class (VMMDISK) must be active for RACF to be able to protect against this access. Second, unless you take additional precautions such as those described in 6.11.3, “Minidisk-owning user” on page 239, use discrete profiles for protecting minidisks. If generic profiles are used, a user can define an overlapping minidisk that matches the generic profile and is permitted access by RACF—a RACF generic profile allowing access to a user-defined minidisk.

In this example, the RACF administrator believes that he or she has set a sufficiently restrictive generic profile that allows both good security and configurability (because the administrator does not want to be involved every time the Linux administrators want to add a minidisk). However, by using a generic profile, an unscrupulous user can define the overlapping minidisk at a virtual device number that is accepted under the generic profile, thereby gaining access to the other minidisk.

6.11.3 Minidisk-owning user

Another useful method for securing DASD is to define all minidisks to one or more z/VM users rather than to each individual virtual machine. To access minidisks, the virtual machines link to the minidisks owned by the minidisk-owning user.

The security of this configuration is greatly enhanced when authority to create minidisks is removed from virtual machines. The MDISK command can be removed from the privilege level of general users, or if an ESM is in place the MDISK command can be restricted (using the VMCMD class in the case of RACF).

When multiple minidisk-owning users are employed, managing the security configuration of multiple virtual machines in different security zones becomes easier. A minidisk-owning user can be created for each security zone, and all minidisks owned by that user can be protected according to the definition of that zone.

6.11.4 Shared DASD considerations

We have touched on some considerations that arise when DASDs are shared between z/VM systems. Those issues are discussed further in this section.

Note: This discussion focuses simply on the security aspects of shared-DASD configurations. There are data integrity issues when using shared-DASD that you would need to satisfy as well, but those issues are not discussed here.

Minidisk overlaps

In 6.11.2, “Overlapping minidisks” on page 238, the exposure caused by the definition of minidisks that overlap other minidisks was presented. In that discussion, minidisks within a single z/VM were used for the examples. If we extend that discussion over multiple z/VM systems, however, we find that protecting disks from data exposure becomes more difficult.

When different z/VM systems share DASDs, DirMaint might not stop the overlapping minidisk definition even if EXTNCHK is on. The reason is that the minidisk that exists on the first system might not be defined in the directory of the second system, so DirMaint does not see the overlap when the minidisk is defined on the second system.

7

Security implications of z/VM SSI and LGR

This chapter covers the security implications of using z/VM 6.2 with single system image (SSI) and live guest relocation (LGR).

This new z/VM feature provides additional flexibility for your environment by allowing Linux guests to be moved from one LPAR to another. It also provides a set of shared resources for member systems and their Linux guests. Rather than managing security of a single implementation on a single device, administrators now are able to manage the security of two or more operating systems. This control of access is a useful mechanism for helping to protect your data. In your SSI cluster, use storage area network (SAN) zoning and logical unit number (LUN) masking to ensure that data will be available only for servers that should access it.

This chapter also explains the implementation steps to build an environment on which relocation of Linux on System z guests will be used to provide flexibility and availability to meet user demands for security and manageability. The objective of these tests is to gather information about security problems you might face when managing Linux guests while running in an SSI cluster.

7.1 Overview

Virtual machines (VMs) allow quick turnaround and flexibility for multiple projects and environments. To benefit from virtual machines, z/VM exploits virtualization and gives administrators the power to manage their resources on the System z platform.

With the IBM z/VM single system image in z/VM v6.2, a running Linux on System z virtual machine can be relocated from one member system to any other, a process known as *live guest relocation* (LGR). Support for LGR allows you to move Linux virtual servers without disruption to the business, thus helping you to avoid planned outages. The z/VM systems are aware of each other and can benefit from their combined resources.

LGR enables clients to avoid loss of service due to planned outages by relocating guests from a system requiring maintenance to a system that remains active during the maintenance period. This capability can be used to move workloads from one z/VM LPAR to another when needed. It also helps to reduce planned outages during hardware changes or a z/VM initial program load (IPL). However, the environment becomes more complex and requires special attention with the shared resources. With this new feature in place, it is important to know how to manage efficiently the configuration in a secure manner.

LGR brings more flexibility to your environment, not just Linux availability. Now you can use this new feature to build a reliable and secure infrastructure for Linux guests running under z/VM by workload balancing. Additionally, hardware and z/VM changes can be executed without affecting service availability.

Extended Count-Key-Data (ECKD) disks on z/VM system are commonly used to support high availability applications. But the demand for Fibre Channel Protocol (FCP)-based shared volumes is growing, and system administrators are facing new challenges to keeping this environment secure.

In a SAN environment, LUN security is typically accomplished through a combination of LUN masking and SAN zoning. These technologies ensure that a given LUN[1] can be accessed only by one guest, as identified by the worldwide port name (WWPN) of your SCSI adapter. To set up the SAN devices on a SSI cluster, you need to set equivalency identifiers (EQIDs). These identifiers are used to ensure that all members of the SSI cluster use the same physical devices and the devices attached over Fibre Channel Connection (FICON).

Also note, when moving VMs from one physical machine to another and Linux relocation completes successfully from the z/VM point of view, that Linux guests might face problems when accessing some devices if the external configuration is not properly set for the destination LPAR.

As described in the next section, we set up a test lab to help explain the importance of the EQIDs and the SAN zoning configuration from a security perspective. We also document various issues that you might face while working with an SSI cluster.

[1] Although an LUN represents a single managed unit of disk storage to an operating system instance, on modern storage subsystems it can be an individual hard disk device or logical disk image provisioned across a group of devices.

7.2 Lab environment configuration

The ITSO test lab consisted of a two-member cluster running on an IBM zEnterprise 196 (z196) machine and an IBM zEnterprise EC12 (short name zEC12) with logical configuration listed inTable 7-1.

Table 7-1 SCSI settings for the test LPARs

LPAR name	SCSI CHPIDs	NPIV mode	Machine
ITSOSSI5	7A and 7B	Yes	zEC12
ITSOSSI6	7C and 7D	Yes	z196

There are two methods available on System z to enhance security and protect shared SAN devices from being visible on more than one Linux guest:

- LUN Access Control (LAC)
- N_Port ID Virtualization (NPIV)

LUN Access Control

LUN Access Control (LAC) was the first primary method used by many system administrators to provide added security for SCSI devices connected to a SAN network on System z.

SAN connections are established between a host WWPN and a storage WWPN. The WWPNs are unique 16-digit numbers. On System z, the WWPN is burned into the FCP Channel Path Identifier (CHPID) port. Because many Linux guests might use a specific FCP port, add additional controls to restrict which guests can access which LUNs on the connection. LAC accomplishes this by writing rules that are loaded into the CHPIDs to control access.

Although LAC is still used, NPIV is now the preferred method to use for the following reasons:

- It is an industry standard.
- It virtualizes Fibre Channel (FC) card (sharing).
- It allows LUN masking and zoning for guests.
- No additional driver is required.
- It improves Fibre Channel card utilization.

N_Port ID Virtualization

N_Port ID Virtualization (NPIV) is an industry standard that allows a single physical Fibre Channel port to be shared among multiple systems. Using this technology, each FCP subchannel or device has a unique worldwide port name. LUNs are masked in the SAN to the WWPN of a specific subchannel or device. This protects those LUNs from being accessed from anywhere other than the mapped devices.

You can connect multiple systems to one physical port of a physical Fibre Channel adapter. Each system has its own unique WWPN associated with its own virtual FC adapter and “sees” only the virtual disks that have been presented to its WWPN.

NPIV simplifies SAN management by allowing each Linux guest to have its own CHPID. It helps to increase security because each Linux guest will only have access to specific LUNs.

Planning your SAN environment: It is advisable to plan your SAN environment up front, before assigning disks to the hosts, because it is difficult and costly to modify a working network. For details about how to set NPIV using the Hardware Management Console (HMC), see 6.10.2, "Using N_Port ID Virtualization" on page 227.

Information about SSI configuration is beyond the scope of this book. For details about configuring SSI, see the following websites:

http://www.redbooks.ibm.com/abstracts/sg248006.html?Open

http://www.vm.ibm.com/ssi

7.2.1 Background information

This section provides background information about equivalency identifiers and their association, which you will need to understand before we discuss our lab test.

Equivalency identifiers

With an SSI cluster, you will use equivalency identifiers (EQIDs). EQIDs are used to ensure that all members of the cluster use both the same physical devices and the devices attached over Fibre Channel Connection (FICON). During z/VM IPL, the EQID number is automatically generated and assigned to various devices, such as DASDs. However, for Fibre Channel Protocol (FCP) devices, you will need to explicitly set the EQID on the system config file located on the `maint` CF0 disk so that all cluster members can see the device as one device.

Before a Linux guest gets relocated to a different z/VM LPAR, the guest configuration is checked on the destination LPAR (Device condition) to ensure that the devices have the same EQIDs. Ensure that all necessary FCP configuration is well planned to avoid problems when relocating your Linux guests. To update the NPIV devices, review and if necessary update your EQIDs as well.

For the list of conditions, see the CP Planning and Administration book.

SAN devices x EQIDs

The FCP channels must be set up with an EQID identifier. Otherwise, the LGR will fail and an error message will be displayed.

Sample setup: We set up the EQIDs for the ITSOSSI6 LPAR simply to demonstrate the error message that might be displayed if the EQID configuration is not correct.

Figure 7-1 shows the relocation error message.

```
vmrelocate test lnxslesa to itsossi5
HCPRLH1940E LNXSLESA is not relocatable for the following reason(s):
HCPRLI1997I LNXSLESA: Source system virtual device B800 on real device B801
cannot be created on the destination system because an eligible equivalent real
device cannot be found
HCPRLI1997I LNXSLESA: Source system virtual device B900 on real device B901
cannot be created on the destination system because an eligible equivalent real
device cannot be found
Ready(01940); T=0.01/0.01 10:30:04
```

Figure 7-1 Relocation error message

7.2.2 Physical and logical configuration

The lab environment was built to simulate a typical production setup according to the logical diagram shown in Figure 7-2 on page 245. We built two zVM LPARs (ITSOSSI5 and ITSOSSI6), each with its own FCP adapter and configured as shown.

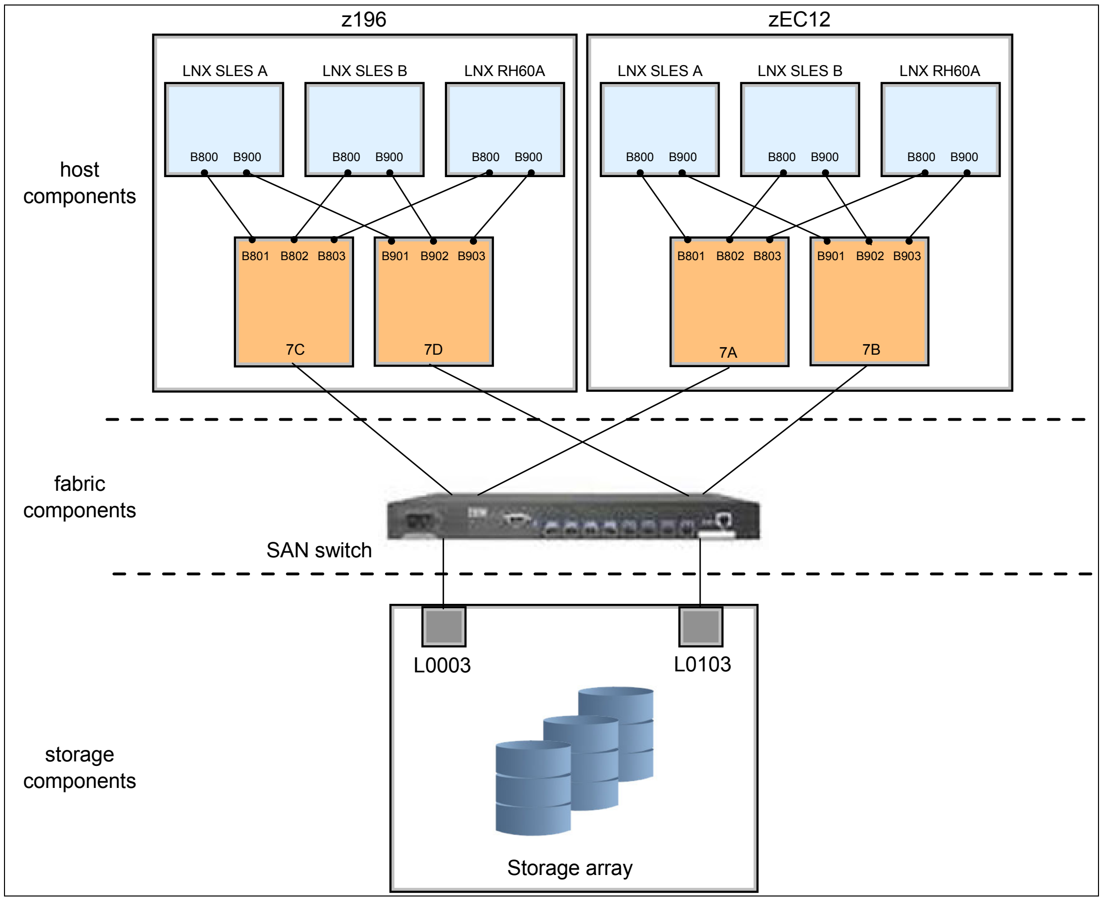


Figure 7-2 Physical and logical configuration example setup

SAN environment

For the SAN environment we formatted two LUNs on a disk subsystem and then assigned a group of worldwide port names that are representing Fibre Channel (FC) ports on a host, as shown in the following tables.

The DS8300 (S/N L300) storage is set up as listed in Table 7-2.

Table 7-2 DS8300 logical details

Device address	WWPN
L0003	500507630500C74C
L0103	500507630508C74C

The SAN information for ITSOSSI5 is listed in Table 7-3.

Table 7-3 SAN logical configuration for ITSOSSI5

CHPID	Device number	WWPN
7A	B800	c05076e5f9001d00
7A	B801	c05076e5f9001d04
7A	B802	c05076e5f9001d08
7A	B803	c05076e5f9001d0c
7A	B804	c05076e5f9001d10
7B	B900	c05076e5f9001d88
7B	B901	c05076e5f9001d8c
7B	B902	c05076e5f9001d90
7B	B903	c05076e5f9001d94
7B	B904	c05076e5f9001d98

The SAN information for ITSOSSI6 is listed in Table 7-4.

Table 7-4 SAN logical configuration for ITSOSSI6

CHPID	Device number	WWPN
7C	B800	c05076ecea801060
7C	B801	c05076ecea801064
7C	B802	c05076ecea801068
7C	B803	c05076ecea80106c
7C	B904	c05076ecea801070
7D	B900	c05076ecea8010e8
7D	B901	c05076ecea8010ec
7D	B902	c05076ecea8010f0
7D	B903	c05076ecea8010f4
7D	B904	c05076ecea8010f8

Linux virtual devices

In a Linux on System z environment, system administrators sometimes confuse the association between virtual devices and real devices. A *virtual device* is an abstraction of one real device and helps z/VM to isolate each Linux guest from interfering with each other. Each Linux guest will use the virtual device as though it had the real device attached.

Figure 7-3 on page 247 illustrates this association.

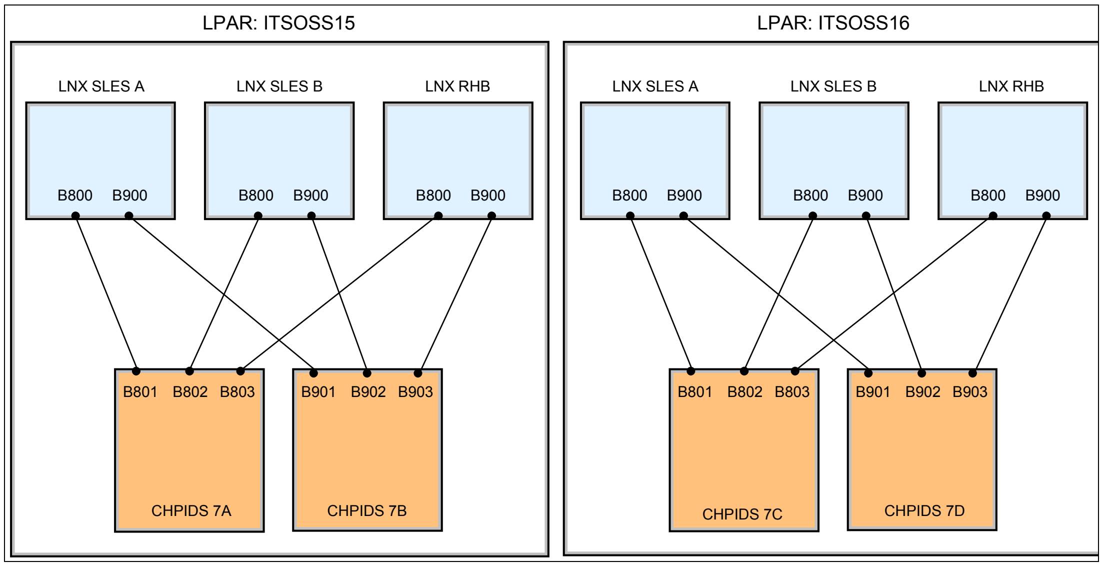


Figure 7-3 Association between virtual devices and real devices

Within z/VM, the term *real device* refers to the physical device (that is, a SCSI adapter attached to CHPID 7A).

SAN zoning

The zoning for our environment was created as explained here. Zone A (used by z/VM FBA Emulation (EDEVs) group 1) is shown in Example 7-1.

Example 7-1 Zone A for lnxrh60b server

```
Zone A (used by RedHat Linux 6.0):
   500507630500C74C
   500507630508C74C
   c05076e5f9001d08
   c05076e5f9001d90
   c05076ecea801068
   c05076ecea8010f0
```

Zone B (used by lnxslesa) is shown in Example 7-2.

Example 7-2 Zone A for lnxslesa server

```
Zone B1 (used by SuSE Linux Enterprise Server 11.2):
   500507630500C74C
   500507630508C74C
   c05076e5f9001d04
   c05076e5f9001d8c

Zone B2 (used by SuSE Linux Enterprise Server 11.2):
   500507630500C74C
   500507630508C74C
   c05076ecea801064
   c05076ecea8010ec
```

Zone C (used by lnxslesb) is shown in Example 7-3.

Example 7-3 Zone C used by lnxslesb

```
Zone C (used by z/VM FBA Emulation (EDEVs)):
   500507630500C74C
   500507630508C74C
   c05076e5f9001d10
   c05076e5f9001d98
   c05076ecea801070
   c05076ecea8010f8
```

Setting up the EQIDs

We used the steps that follow to assign the EQIDs SAN1001 and SAN1001 in our SSI cluster. You might want to use a convention for the EQIDs that best fits your configuration. Note that we used the same procedure to create the EQIDs for all devices used in our test.

These commands were executed in ITSOSSI5 and ITSOSSI6 LPARs.

1. To set up the EQID names, you must take the devices offline (Example 7-4).

   Example 7-4 Taking devices b801 and b901 offline

   ```
   vary offline b801 b901
   B801 varied offline
   B901 varied offline
   2 device(s) specified; 2 device(s) successfully varied offline
   ```

2. After the devices are offline, you can use the **set** command to define the EQID names for your devices, as shown in the next two examples. Example 7-5 illustrates setting the EQID name on b801.

   Example 7-5 Setting EQID name on b801

   ```
   set rdev b801 eqid SAN10001 type fcp
   HCPZRP6722I Characteristics of device B801 were set as requested.
   1 RDEV(s) specified; 1 RDEV(s) changed; 0 RDEV(s) created
   ```

   Example 7-5 illustrates setting the EQID name on b901.

   Example 7-6 Setting EQID name on b901

   ```
   set rdev b901 eqid SAN10002 type fcp
   HCPZRP6722I Characteristics of device B901 were set as requested.
   1 RDEV(s) specified; 1 RDEV(s) changed; 0 RDEV(s) created
   ```

3. Now you can vary your devices online (Example 7-7).

   Example 7-7 Varying devices b801 and b901 online

   ```
   vary online b801 b901
   B801 varied online
   B901 varied online
   2 device(s) specified; 2 device(s) successfully varied online
   ```

4. Example 7-8 shows the stanza for the FCP devices in SYSTEM CONFIG for ITSOSSI5 and ITSOSSI6 LPARs.

Example 7-8 FCP EQIDs definitions for ITSOSSI5 and ITSOSSI6

```
RDEV b801 EQID SAN10001 TYPE FCP
RDEV b901 EQID SAN10002 TYPE FCP
```

The SAN10001 and SAN10002 EQID names were used in our test lab, but you can use a nomenclature that will help you to associate EQID names to the real devices.

5. You can run the command shown in Example 7-9 to display the EQIDs for the b801 and b901 devices.

Example 7-9 Querying EQID on b801 and b901

```
q eqid for b801 b901
B801: EQID = SAN10001
B901: EQID = SAN10002
```

Setting up a SCSI device on a SUSE Linux Server

The setup steps are listed here:

1. Ensure that the user's directory entry contains information about the FCP channels, as shown in Example 7-10 for the lnxslesa server.

Example 7-10 User's directory for lnxslesa

```
DEDICATE B800 B801
DEDICATE B900 B901
```

2. Check the attached FCP channels (Example 7-11 on page 249).

Privileged ID: You must be on a privileged ID to run these commands.

Example 7-11 Checking attached FCP channels

```
lnxslesa:~ # modprobe vmcp
lnxslesa:~ # vmcp q fcp
FCP  B800 ON FCP   B801 CHPID 78 SUBCHANNEL = 000A
     B800 DEVTYPE FCP         VIRTUAL CHPID 78 FCP REAL CHPID 78
     B800 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     WWPN C05076F77A000204
FCP  B900 ON FCP   B901 CHPID 79 SUBCHANNEL = 000B
     B900 DEVTYPE FCP         VIRTUAL CHPID 79 FCP REAL CHPID 79
     B900 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     WWPN C05076F77A0008EC
```

3. Load the necessary modules (Example 7-12 on page 249).

Example 7-12 Loading necessary modules

```
modprobe zfcp
modprobe sg
```

Required module: The sg module is required by `lsluns` and other commands.

4. Enable the virtual devices (Example 7-13 on page 250).

Example 7-13 Enabling virtual devices

```
lnxslesa:~ # zfcp_host_configure 0.0.b800 1
Configuring device 0.0.b800
lnxslesa:~ # zfcp_host_configure 0.0.b900 1
Configuring device 0.0.b900
```

The **`zfcp_host_configure`** command will enable the FCP device and create the zFCP configuration files (`/etc/udev/rules.d/51-zfcp-*`).

5. After the virtual devices b800 and b900 are activated, you can run **`/usr/sbin/lsluns`** to list which LUNs are available on the storage system (Example 7-14 on page 250).

Example 7-14 Listing available LUNs

```
lnxslesa:~ # lsluns
Scanning for LUNs on adapter 0.0.b800
   at port 0x500507630500c74c:
      0x4010400c00000000
      0x4011400c00000000
   at port 0x500507630508c74c:
      0x4010400c00000000
      0x4011400c00000000
Scanning for LUNs on adapter 0.0.b900
   at port 0x500507630500c74c:
      0x4010400c00000000
      0x4011400c00000000
   at port 0x500507630508c74c:
      0x4010400c00000000
      0x4011400c00000000
```

6. Based on that output, you can create all the commands to activate each LUN (Example 7-15 on page 250).

Example 7-15 Activating LUNs

```
lnxslesa:~ # zfcp_disk_configure 0.0.b800 0x500507630500c74c 0x4010400c00000000 1
Configuring FCP disk 500507630500c74c:4010400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b800 0x500507630500c74c 0x4011400c00000000 1
Configuring FCP disk 500507630500c74c:4011400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b900 0x500507630500c74c 0x4010400c00000000 1
Configuring FCP disk 500507630500c74c:4010400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b900 0x500507630500c74c 0x4011400c00000000 1
Configuring FCP disk 500507630500c74c:4011400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b800 0x500507630508c74c 0x4010400c00000000 1
Configuring FCP disk 500507630508c74c:4010400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b800 0x500507630508c74c 0x4011400c00000000 1
Configuring FCP disk 500507630508c74c:4011400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b900 0x500507630508c74c 0x4010400c00000000 1
Configuring FCP disk 500507630508c74c:4010400c00000000
lnxslesa:~ # zfcp_disk_configure 0.0.b900 0x500507630508c74c 0x4011400c00000000 1
Configuring FCP disk 500507630508c74c:4011400c00000000
```

The **`zfcp_disk_configure`** command activates the new LUNs and updates the `zfcp` configuration files (`/etc/udev/rules.d/51-zfcp-*`).

7. Check whether the LUNs are online (Example 7-16 on page 250).

Example 7-16 Checking new LUNs

```
lnxslesa:~ # lsscsi
[0:0:0:1074544656]disk    IBM      2107900          .120  /dev/sde
[0:0:0:1074544657]disk    IBM      2107900          .120  /dev/sdf
[0:0:1:1074544656]disk    IBM      2107900          .120  /dev/sda
[0:0:1:1074544657]disk    IBM      2107900          .120  /dev/sdb
[1:0:0:1074544656]disk    IBM      2107900          .120  /dev/sdg
[1:0:0:1074544657]disk    IBM      2107900          .120  /dev/sdh
[1:0:1:1074544656]disk    IBM      2107900          .120  /dev/sdc
[1:0:1:1074544657]disk    IBM      2107900          .120  /dev/sdd
```

8. The `udev` daemon provides persistent naming for the SCSI devices. It creates files in the `/dev/disk` directory. To view the persistent names that been created for your SCSI devices, you can use command shown (Example 7-17 on page 251).

Example 7-17 Searching new LUNs online

```
lnxslesa:~ # ls -la /dev/disk/by-id/scsi-3*
lrwxrwxrwx 1 root root 10 Aug 23 14:44
/dev/disk/by-id/scsi-36005076305ffc74c000000000000100c -> ../../dm-0
lrwxrwxrwx 1 root root 10 Aug 23 14:44
/dev/disk/by-id/scsi-36005076305ffc74c000000000000110c -> ../../dm-1
lnxslesa:~ #
```

9. Create a valid partition for new LUNs (Example 7-18 on page 251).

Example 7-18 Creating a valid partition for the new LUNs

```
lnxslesa:~ # fdisk /dev/disk/by-id/scsi-36005076305ffc74c000000000000100c
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0x8a1aa3cb.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048): [ENTER]
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519):
Using default value 20971519

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

lnxslesa:~ # fdisk /dev/disk/by-id/scsi-36005076305ffc74c000000000000110c
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0x848cbf28.
Changes will remain in memory only, until you decide to write them.
```

```
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048): [ENTER]
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519):
Using default value 20971519

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Setting up a SCSI device on a Red Hat Linux server

1. Ensure that the user's directory contains information about the FCP channels, as shown in Example 7-19 for the lnxrh60b server.

Example 7-19 User's directory for lnxrh60b

```
DEDICATE B800 B802
DEDICATE B900 B902
```

2. Check the attached FCP channels (Example 7-20).

Example 7-20 Checking attached FCP channels

```
[root@lnxrh60b ~]# modprobe fcp
[root@lnxrh60b ~]# vmcp q fcp
FCP  B800 ON FCP   B802 CHPID 7A SUBCHANNEL = 000A
     B800 DEVTYPE FCP         VIRTUAL CHPID 7C FCP REAL CHPID 7A
     B800 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     WWPN C05076E5F9001D08
FCP  B900 ON FCP   B902 CHPID 7B SUBCHANNEL = 000B
     B900 DEVTYPE FCP         VIRTUAL CHPID 7D FCP REAL CHPID 7B
     B900 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     WWPN C05076E5F9001D90
```

3. Load the necessary modules (Example 7-21).

Example 7-21 Loading necessary modules

```
modprobe zfcp
modprobe sg
```

4. Use the **cio_ignore** command to remove the FCP adapter from the list of ignored devices and make it visible to Linux (Example 7-22 on page 253).

Ignored devices: In Red Hat Linux, you need to free an ignored devices from the ignore list. Otherwise, the device cannot be used; see Example 7-22. If you want to display the ignored devices, run the **`cat /proc/cio_ignore`** command.

Example 7-22 Removing FCP adapters from the list of ignored devices

```
[root@lnxrh60b ~]# cio_ignore -r b800
[root@lnxrh60b ~]# cio_ignore -r b900
```

5. To bring the FCP adapter device online, use the command shown in Example 7-23.

Example 7-23 Bringing FCP adapters online

```
[root@lnxrh60b ~]# chccwdev -e b800
Setting device 0.0.b800 online
Done
[root@lnxrh60b ~]# chccwdev -e b900
Setting device 0.0.b900 online
Done
```

6. Verify that the required WWPN is detected by the automatic port scanning of the `zfcp` device driver by running the command shown in Example 7-24.

Example 7-24 Verifying WWPNs for your FCP devices

```
[root@lnxrh60b ~]# lszfcp -P
0.0.b800/0x500507630508c74c rport-0:0-0
0.0.b800/0x500507630500c74c rport-0:0-1
0.0.b900/0x500507630508c74c rport-1:0-0
0.0.b900/0x500507630500c74c rport-1:0-1
```

7. List the available LUNs by running the command shown in Example 7-25.

Example 7-25 Listing available LUNs

```
[root@lnxrh60b ~]# lsluns
Scanning for LUNs on adapter 0.0.b800
   at port 0x500507630500c74c:
      0x4010400d00000000
      0x4011400d00000000
   at port 0x500507630508c74c:
      0x4010400d00000000
      0x4011400d00000000
Scanning for LUNs on adapter 0.0.b900
   at port 0x500507630500c74c:
      0x4010400d00000000
      0x4011400d00000000
   at port 0x500507630508c74c:
      0x4010400d00000000
      0x4011400d00000000
```

8. Activate the FCP LUN by adding the device map to the `/etc/zfcp.conf` file (Example 7-26). The syntax is as follows: <FCP_device> <WWPN> <LUN>

Example 7-26 Entries for the /etc/zfcp.conf file

```
0.0.b800 0x500507630500c74c 0x4010400d00000000
0.0.b800 0x500507630500c74c 0x4011400d00000000
0.0.b900 0x500507630500c74c 0x4010400d00000000
```

```
0.0.b900 0x500507630500c74c 0x4011400d00000000
0.0.b800 0x500507630508c74c 0x4010400d00000000
0.0.b800 0x500507630508c74c 0x4011400d00000000
0.0.b900 0x500507630508c74c 0x4010400d00000000
0.0.b900 0x500507630508c74c 0x4011400d00000000
```

9. Run **`/sbin/zfcpconf.sh`** script to read the `/etc/zfcp.conf` file and activate each disk.
10. Check whether LUNs are online (Example 7-27). Note that each SCSI device represents one path.

Example 7-27 Checking active LUNs

```
[root@lnxrh60b ~]# lsscsi
[0:0:0:1074610192]disk    IBM      2107900          .120  /dev/sda
[0:0:0:1074610193]disk    IBM      2107900          .120  /dev/sdb
[0:0:1:1074610192]disk    IBM      2107900          .120  /dev/sde
[0:0:1:1074610193]disk    IBM      2107900          .120  /dev/sdf
[1:0:0:1074610192]disk    IBM      2107900          .120  /dev/sdc
[1:0:0:1074610193]disk    IBM      2107900          .120  /dev/sdd
[1:0:1:1074610192]disk    IBM      2107900          .120  /dev/sdg
[1:0:1:1074610193]disk    IBM      2107900          .120  /dev/sdh
```

11. Search new LUNs online by running the command shown in Example 7-28.

Example 7-28 Searching new LUNs online

```
[root@lnxrh60b ~]# ls -la /dev/disk/by-id/scsi-3*
lrwxrwxrwx 1 root root 9 Aug 21 16:01
/dev/disk/by-id/scsi-36005076305ffc74c000000000000100d -> ../../sdc
lrwxrwxrwx 1 root root 9 Aug 21 16:01
/dev/disk/by-id/scsi-36005076305ffc74c000000000000110d -> ../../sdb
```

12. Create a valid partition for the new LUNs by running the command shown in Example 7-29.

Example 7-29 Creating a valid partition for the new LUNs

```
[root@lnxrh60b ~]# fdisk /dev/disk/by-id/scsi-36005076305ffc74c000000000000100d
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0xe48f2bf5.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-10240, default 1): [ENTER]
Using default value 1
```

```
Last cylinder, +cylinders or +size{K,M,G} (1-10240, default 10240):
Using default value 10240

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

[root@lnxrh60b ~]# fdisk /dev/disk/by-id/scsi-36005076305ffc74c000000000000110d
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0x19c7ce43.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-10240, default 1): [ENTER]
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-10240, default 10240):
Using default value 10240

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Setting up the multipath daemon

There are two methods available to set up the `multipath` daemon:

- Active/passive
- Active/active (with round-robin load balancing)

In the ITSO lab, we used the default value (round-robin), but you can select another available option.

To set up the multipath daemon, you need to create a file called `/etc/multipath.conf` and add the appropriate device entries for the storage subsystem. Example 7-30 contains the multipath configuration for the two LUNs of the lnxslesa server. We defined a symbolic name (alias) for each multipath device as shown in Table 7-5 on page 256 and Example 7-30 on page 256.

Table 7-5 Alias definition

WWID	Alias
36005076305ffc74c000000000000110c	svc1
36005076305ffc74c000000000000100c	svc2

Linux guest consideration: For a Linux guest managed by an SSI cluster, make sure the `queue_if_no_path` option is specified in the `/etc/multipath.conf` file. This ensures that I/O errors are retried and queued if all paths are failed in the `dm-multipath` layer.

Samples of the `multipath.conf` file can be found under the `/usr/share/doc/packages/multipath-tools/` folder for SUSE Linux 11 SP2, and in `/usr/share/doc/device-mapper-multipath-0.4.9` for Red Hat 6.0.

Example 7-30 Output of /etc/multipath.conf

```
multipaths {
multipath {
  }
  multipath {
    wwid 36005076305ffc74c000000000000110c
    alias svc1
  }
  multipath {
    wwid 36005076305ffc74c000000000000100c
    alias svc2
  }
}
blacklist {
  devnode "^(dasd)[0-9]*"
}
defaults {
   features "1 queue_if_no_path"
   user_friendly_names yes
}
```

You can then run the commands shown in Example 7-31 to load the `multipath` daemon and set up the startup boot script.

Example 7-31 Starting Multipath daemon and enabling startup script

```
lnxslesa:~ # /etc/init.d/multipathd start
Starting multipathd
done
lnxslesa:~ # chkconfig multipathd on
lnxslesa:~ #
```

Now you can use the **`multipath -ll`** command to review your multipath configuration (Example 7-32).

Example 7-32 Output of multipath command

```
lnxslesa:~ # multipath -ll
svc2 (36005076305ffc74c000000000000100c) dm-0 IBM,2107900
```

```
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  |- 0:0:1:1074544656 sda 8:0   active ready running
  |- 1:0:1:1074544656 sdc 8:32  active ready running
  |- 0:0:0:1074544656 sde 8:64  active ready running
  `- 1:0:0:1074544656 sdg 8:96  active ready running
svc1 (36005076305ffc74c000000000000110c) dm-1 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  |- 0:0:1:1074544657 sdb 8:16  active ready running
  |- 1:0:1:1074544657 sdd 8:48  active ready running
  |- 0:0:0:1074544657 sdf 8:80  active ready running
  `- 1:0:0:1074544657 sdh 8:112 active ready running
```

If you are using LVM, the new LUNs can be included into LVM (Example 7-33).

Example 7-33 Adding and showing new LUNs in LVM

```
lnxslesa:~ # pvcreate /dev/mapper/svc1_part1
  Physical volume "/dev/mapper/svc1_part1" successfully created
lnxslesa:~ # pvcreate /dev/mapper/svc2_part1
  Physical volume "/dev/mapper/svc2_part1" successfully created

lnxslesa:~ # pvscan
  PV /dev/mapper/svc2_part1                     lvm2 [10.00 GiB]
  PV /dev/mapper/svc1_part1                     lvm2 [10.00 GiB]
  Total: 2 [20.00 GiB] / in use: 0 [0   ] / in no VG: 2 [20.00 GiB]
```

Permanent settings

The following instructions are helpful if you need to make your FCP settings permanent.

On Red Hat

On Red Hat, you need to update the `/etc/zfcp.conf` file to include the new LUNs and run **`/sbin/zfcpconf.sh`** script. The syntax of this file is: <FCP_device> <WWPN> <LUN>

Example 7-26 on page 253 shows a sample of these settings.

On SUSE

On SUSE, we suggest you use the **`zfcp_host_configure`** and **`zfcp_disk_configure`** commands to set up the SCSI devices, because they will update the zfcp configuration files (`/etc/udev/rules.d/51-zfcp-*`) to make settings persistent during reboots.

Example 7-13 on page 250 and Example 7-15 on page 250 show samples of these commands.

7.2.3 Proposed tests

In this section we describe the following tests:

- Relocating a server (FBA Emulation)
- Relocating a server (SCSI directly attached)
- RACF service machine down relocation test

Relocating a server (FBA emulation)

Relocating a server by using a technique that allows z/VM to use SCSI disks in the same way as it uses ECKDs is called FBA emulation. Using this technique, Linux guests can be assigned minidisks that reside on SCSI LUN.

FBA emulation enhances security by allowing CP (and the external security manager, if installed) to manage access to the minidisks on the SCSI LUN. FBA emulation also enables easier configuration for the Linux administrator because the FBA emulation function in CP handles device recovery and multipath configuration.

For more detailed information about FBA emulation, see 6.10.1, "Using FBA emulation in z/VM".

We used the `SET EDEVICE` command to configure a SCSI disk LUN to the z/VM system as an emulated 9336 disk (Example 7-34 on page 258). The FBA device number used in our test is 5c00.

Example 7-34 Creating the 5c00 FBA device on ITSOSSI5

```
set edev 5c00 type fba attr 2107 fcp_dev b804 wwpn 500507630500C74C lun
4010400f00000000
EDEV 5C00 was created.
set edev 5c00 type fba attr 2107 add path fcp_dev b904 wwpn 500507630508C74C lun
4010400f00000000
EDEV 5C00 was modified.
```

Tip: Remember to update your SYSTEM CONFIG file to include the EDEVICE configuration.

To confirm if the FBA device was successfully created, run the command shown in Example 7-35.

Example 7-35 Listing details for 5c00 on ITSOSSI5

```
q edev 5c00 details
EDEV 5C00 TYPE FBA ATTRIBUTES 2107
PATHS:
FCP_DEV: B804 WWPN: 500507630500C74C LUN: 4010400F00000000
FCP_DEV: B904 WWPN: 500507630508C74C LUN: 4010400F00000000
```

To put the FBA device online, run the command shown in Example 7-36.

Example 7-36 Varying 5c00 device online on ITSOSSI5.

```
vary online 5c00
5C00 varied online
1 device(s) specified; 1 device(s) successfully varied online
```

Use the command shown in Example 7-37 to check whether the FBA device is online.

Example 7-37 Listing 5c00 FBA device - online paths on ITSOSSI5.

```
q edev 5c00 details
EDEV 5C00 TYPE FBA ATTRIBUTES 2107
VENDOR: IBM PRODUCT: 2107900 REVISION: .120
BLOCKSIZE: 512 NUMBER OF BLOCKS: 20971520
PATHS:
```

```
FCP_DEV: B904 WWPN: 500507630500C74C LUN: 4010400F00000000
CONNECTION TYPE: SWITCHED
FCP_DEV: B904 WWPN: 500507630508C74C LUN: 4010400F00000000
CONNECTION TYPE: SWITCHED
EQID: 6005076305FFC74C00000000000010F0C600000000013FFFFF
```

If the Linux guest is hosted in an SSI cluster, you need to define the EDEVICE in all z/VM cluster members. Example 7-38 on page 259 shows the commands that were used to create an FBA device (5c00) in the ITSOSSI6 LPAR.

Example 7-38 Creating the 5c00 FBA device onITSOSSI6

```
set edev 5c00 type fba attr 2107 fcp_dev b804 wwpn 500507630500C74C lun 4010400f
00000000
EDEV 5C00 was created.
Ready; T=0.01/0.02 16:47:51
set edev 5c00 type fba attr 2107 add path fcp_dev b904 wwpn 500507630508C74C lun
4010400f00000000
EDEV 5C00 was modified.
```

Use the command shown in Example 7-39 on page 259 to vary the 5c00 device online on ITSOSSI6.

Example 7-39 Varying 5c00 online on ITSOSSI6

```
vary online 5c00
5C00 varied online
1 device(s) specified; 1 device(s) successfully varied online
```

Example 7-40 shows the properties of the 5c00 device.

Example 7-40 Querying the 5c00 device on ITSOSSI6

```
q edev 5c00 details
EDEV 5C00 TYPE FBA ATTRIBUTES 2107
VENDOR: IBM PRODUCT: 2107900 REVISION: .120
BLOCKSIZE: 512 NUMBER OF BLOCKS: 20971520
PATHS:
FCP_DEV: B804 WWPN: 500507630500C74C LUN: 4010400F00000000
CONNECTION TYPE: SWITCHED
FCP_DEV: B904 WWPN: 500507630508C74C LUN: 4010400F00000000
CONNECTION TYPE: SWITCHED
EQID: 6005076305FFC74C00000000000010F0C600000000013FFFFF
```

In our case, we gave a Linux guest a minidisk (210) residing on an emulated FBA volume (Example 7-41).

Example 7-41 Listing FBA volume

```
lnxslesb:~ # lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0202   active      dasda     94:0    ECKD  4096   6339MB    1622880
0.0.0201   active      dasdb     94:4    ECKD  4096   703MB     180000
0.0.0203   active      dasdc     94:8    ECKD  4096   10546MB   2700000
0.0.0210   active      dasdd     94:12   FBA   512    7324MB    15000000
```

The EDEVICEs were successfully configured. As a test, we moved a Linux guest from ITSOSSI5 to ITSOSSI6 (Example 7-42).

Example 7-42 Relocating server to ITSOSSI6

```
vmrelote move lnxslesb to itsossi6
Relocation of LNXSLESB from ITSOSSI5 to ITSOSSI6 started
User LNXSLESB has been relocated from ITSOSSI5 to ITSOSSI6
```

The server relocation was successfully completed because the EDEVICE was properly set up in both cluster members. We noticed that the EQID was automatically created for the EDEVICE, as shown in Example 7-37 on page 258 and Example 7-40 on page 259.

For our next test, we deactivated the EDEVICE on ITSOSSI5 LPAR as detailed in Example 7-43 and tried to run a server relocation.

Example 7-43 Deactivating the 5c00 EDEVICE

```
q edev 5c00  details
EDEV 5C00 TYPE FBA ATTRIBUTES 2107
  VENDOR: IBM PRODUCT: 2107900 REVISION: .120
  BLOCKSIZE: 512 NUMBER OF BLOCKS: 20971520
  PATHS:
    FCP_DEV: B804 WWPN: 500507630508C74C LUN: 4010400F00000000
      CONNECTION TYPE: SWITCHED
    FCP_DEV: B904 WWPN: 500507630508C74C LUN: 4010400F00000000
      CONNECTION TYPE: SWITCHED
  EQID: 6005076305FFC74C000000000000010F0C600000000013FFFFF
Ready; T=0.01/0.01 15:08:26

det 5c00 system
DASD 5C00  DETACHED SYSTEM
Ready; T=0.01/0.01 15:08:58

vary offline 5c00
5C00 varied offline
1 device(s) specified; 1 device(s) successfully varied offline
Ready; T=0.01/0.01 15:09:04

set edev 5c00 clear
EDEV 5C00 was cleared.
Ready; T=0.01/0.01 15:09:23

q edev 5c00  details
EDEV 5C00 TYPE UNK ATTRIBUTES UNK
  PATHS:
    No paths exist.
```

As expected, it failed (Example 7-44).

Example 7-44 Output of the server relocation

```
at itsossi6 cmd vmrelocate test lnxslesb to itsossi5
HCPRLH1940E LNXSLESB is not relocatable for the following reason(s):
```

```
HCPRLI1997I LNXSLESB: Source system virtual device 0210 on real device 5C00 cannot
be created on the destination system because the real device does not exist there
```

Relocating a server (SCSI directly attached)

The test listed in this section illustrates the kind of problems you might face when using the **vmrelocate** command to move a server from a LPAR to another. Intentionally, we did not create the required security zoning on ITSOSSI5 LPAR for the lnxslesa server. Although the **vmrelocate** command will return a successfully relocation, the lnxslesa server will not have access to the SCSI disks (LUNs).

The point of this test is to demonstrate that although the configuration for the EQIDs is complete on z/VM side, that does not guarantee that the server relocation will work from the perspective of Linux.

Example 7-45 shows the configuration of the EQIDs, which are correctly configured on the ITSOSSI5 and ITSOSSI6 LPARs.

Example 7-45 Status of EQIDs on ITSSOSSI5 and ITSOSSI6

```
at itsossi5 cmd q eqid for b801 b901
B801: EQID = SAN10001
B901: EQID = SAN10002
Ready; T=0.01/0.01 09:10:56

at itsossi6 cmd q eqid for b801 b901
B801: EQID = SAN10001
B901: EQID = SAN10002
```

The commands listed in Example 7-46 show the status of the Linux server running on ITSOSSI6.

Example 7-46 Output of the multipath command listing the ready paths

```
lnxslesa:~ # vmcp q userid
LNXSLESA AT ITSOSSI6

lnxslesa:~ # multipath -ll
mpathb (36005076305ffc74c000000000000100c) dm-1 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  |- 1:0:1:1074544656 sdc 8:32 active ready running
  `- 0:0:0:1074544656 sdd 8:48 active ready running
mpatha (36005076305ffc74c000000000000100b) dm-0 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  |- 1:0:1:1074479120 sda 8:0  active ready running
  `- 0:0:0:1074479120 sdb 8:16 active ready running
```

Now we move a server from ITSOSSI6 to ITSOSSI5 (Example 7-47).

Example 7-47 Relocating server from ITSOSSI6 to ITSOSSI5

```
vmrelocate move lnxslesa to itsossi5
Relocation of LNXSLESA from ITSOSSI6 to ITSOSSI5 started
User LNXSLESA has been relocated from ITSOSSI6 to ITSOSSI5
```

Although the server relocation was successfully completed from a z/VM point of view, the Linux server now running on ITSOSSI5 does not have access to the SAN disks (LUNs) because of an incomplete SAN zoning.

Example 7-48 shows the output of the **multipath** command for the undefined (undef) paths.

Example 7-48 Output of Linux when zoning is not created on the target LPAR

```
lnxslesa:~ # vmcp q userid
LNXSLESA AT ITSOSSI5

lnxslesa:~ # multipath -ll
mpathb (36005076305ffc74c000000000000100c) dm-1 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=0 status=active
  |- 0:0:0:1074544656 sdb 8:16 active undef running
  `- 1:0:1:1074544656 sdd 8:48 active undef running
mpatha (36005076305ffc74c000000000000100b) dm-0 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=0 status=active
  |- 0:0:0:1074479120 sda 8:0  active undef running
  `- 1:0:1:1074479120 sdc 8:32 active undef running
```

The error messages for the multipath daemon reside in the `/var/log/messages` file (Example 7-49).

Example 7-49 Output of /var/log/messages file

```
rport-0:0-0: blocked FC remote port time out: removing target and saving binding
rport-1:0-1: blocked FC remote port time out: removing target and saving binding
sd 0:0:0:1074479120: [sda] Synchronizing SCSI cache
sd 1:0:1:1074479120: [sdc] Synchronizing SCSI cache
Asd 0:0:0:1074479120: [sda]  Result: hostbyte=DID_NO_CONNECT driverbyte=DRIVER_OK
sd 1:0:1:1074479120: [sdc]  Result: hostbyte=DID_NO_CONNECT driverbyte=DRIVER_OK
sd 0:0:0:1074544656: [sdb] Synchronizing SCSI cache
sd 0:0:0:1074544656: [sdb]  Result: hostbyte=DID_NO_CONNECT driverbyte=DRIVER_OK
sd 1:0:1:1074544656: [sdd] Synchronizing SCSI cache
sd 1:0:1:1074544656: [sdd]  Result: hostbyte=DID_NO_CONNECT driverbyte=DRIVER_OK
mpatha: devmap removed
mpatha: stop event checker thread (4397993236752)
mpatha: removed map after removing all paths
dm-3: remove map (uevent)
dm-3: devmap not registered, can't remove
sdb: remove path (uevent)
mpathb: failed to set dev_loss_tmo: error 16
mpathb: load table [0 20971520 multipath 1 queue_if_no_path 0 1 1 round-robin 0 1
1 8:48 1]
DM message failed [reinstate_path 8:48]
sdb: path removed from map mpathb
sdd: remove path (uevent)
multipath: adding disabled device 8:48
multipath: message: error getting device 8:48
mpathb: devmap removed
mpathb: stop event checker thread (4397993269520)
mpathb: removed map after removing all paths
```

This error occurs when a server gets moved to a second LPAR that does not have the SAN zoning defined properly for the SCSI disks. A Linux guest detects the failed path and removes it from its multipath definition. Because not all paths are reachable and available, the devices cannot be accessed by the Linux guest.

RACF service machine down relocation test

To demonstrate the importance of the RACF service machine for an SSI cluster, we took the racfvm machine down and ran a server relocation as shown in Example 7-50 and Example 7-51.

Example 7-50 Taking the racfvm machine down

```
force racfvm
Ready; T=0.01/0.01 15:33:36
USER DSC LOGOFF AS RACFVM USERS = 26 FORCED BY MAINT

q racfvm
HCPCQU045E RACFVM not logged on
```

The AT command: Using the **AT** command, you can manage all members in an SSI cluster from one member. The **AT** command can be used with most commands, other than class G commands. For a complete description of the **AT** command, see the *z/VM: CP Commands and Utilities Reference*.

Example 7-51 Trying to relocate lnxrh56 server to itsossi6

```
at itsossi5 cmd vmrelo test lnxrh56 to itsossi6
User LNXRH56 is eligible for relocation to ITSOSSI6
Ready; T=0.01/0.01 15:34:03

at itsossi5 cmd vmrelo move lnxrh56 to itsossi6
Relocation of LNXRH56 from ITSOSSI5 to ITSOSSI6 started
Relocation of user LNXRH56 from ITSOSSI5 to ITSOSSI6 did not complete. Guest has
not been moved
HCPRLH1939E Relocation of LNXRH56 to ITSOSSI6 is terminated for the following re
ason(s):
HCPRLU6525E The External Security Manager on ITSOSSI6 is unavailable.
```

To bring RACF service machine back online, you need to run the command shown in Example 7-52.

Example 7-52 Bringing the racfvm machine back online

```
send operator xautolog racfvm
Ready; T=0.01/0.01 15:36:34

q racfvm
RACFVM - DSC
```

You cannot perform a server relocation while the racfvm service machine is down because the RACF machine cleans the references of the user's presence on the source LPAR and prepares for the destination LPAR.

7.2.4 Considerations

The z/VM system automatically generates EQIDs for the FBA DEVICES (EDEVICEs). However, you need to set the EQID if Linux guests are accessing SCSI disks directly through an FCP subchannel.

It is possible that your storage system does not support automatically creating the EQID based on the information that the CP obtains from the storage system. You will need to verify on the storage system that you use whether a valid EQID is automatically generated. If not, you can specify the EQID parameter when creating the EDEVICE or attaching the FCP subchannel.

When z/VM performs a live guest relocation, one of the first things it does is to check whether the EQIDs are correctly set from the source z/VM system to the target. Ensure you have everything set up properly to avoid unexpected problems. Furthermore, if you are going to transition your z/VMs to an SSI cluster, keeping the LUN masking and SAN zoning properly set up is going to become even more important in terms of security.

The z/VM SSI and LGR capabilities imply that less experienced system administrators need to expand their knowledge set to include new concepts introduced by this technology. In addition to z/VM configuration, they now need to review the external security configuration such as zoning and LUN masking.

In this chapter we note that the racfvm service machine is a required and important component for LGR. If it is down, the relocation will not complete. We have also provided a few scenarios to illustrate cases where Linux guests obtain access to LUN disks that help manage and create a secure environment. If you are not using some kind of SCSI devices, consider doing so.

8

Best practices

In this chapter we provide a collection of best practices for securing System z when running a Linux environment, highlighting its unique features and also illustrating how taking advantage of such features can simplify system management.

The best practices to manage IT security are already well documented by several sources. The practices can certainly be used to also secure Linux on System z. However, several unique technologies that this platform leverages should be considered when you define the security policies. These technologies can harden the overall security by providing centralized management capabilities and reducing the number of control checks compared to a server farm that is based on physically distributed servers.

We discuss some of the best practices that we could gather for securing your data and your network, and also how to reinforce access control and authentication. It was written as an executive summary for the technologies and techniques described in this book. This chapter can serve as a quick implementation guide that you can use when assessing the security requirements for your environment.

8.1 Security checklist

First, set up your virtual environment in a secure fashion. Items necessary to check can be found in the following check list. Your company most likely has specific requirements for information security which should be taken into consideration when you build your own check list. Here is a high-level list that could be easily adapted and used for most environments:

- Protect your physical IT infrastructure.
- Secure the logical access to z/VM.
- Protect your data.
- Protect your virtual network.
- Secure the logical access to the Linux servers.
- Protect your environment from yourself by keeping consistent and auditable system logs.

In this section we discuss some of the suggested practices that could make your virtual IT infrastructure secure, resilient, and compliant with your company's security policies.

8.2 Physical security

When outlining security policies for IT environments, system programmers or server administrators commonly take physical security for granted. As the starting point to secure your environment, make sure that all the precautions to physically secure your data center have been completed. The System z platform offers the ability to reduce physical security requirements through the consolidation of a significant number of servers onto a platform that is able to provide the same, if not superior, service level compared to multiple servers spread in a distributed environment. More concepts and information regarding physical and infrastructure security on the platform can be found in Chapter 6, "Physical and infrastructure security on System z" on page 195.

8.3 Securing the logical access to z/VM

While running Linux virtual guests with z/VM as the hypervisor, z/VM security is as important as what is implemented at the Linux level. When the hypervisor security is compromised, all the virtual components of your virtual IT infrastructure are potentially being compromised as well.

The details about how to secure the z/VM environment are not meant to be addressed in this book, details can be found in *Security on z/VM*, SG24-7471. In this section, we highlight how to secure several key z/VM functions and features that are particularly important when supporting Linux servers as guests on this operating system.

8.3.1 z/VM user passwords

On a z/VM system, the passwords for users are not world-readable, and only z/VM administrators with very specific privileges are able to have access to the user directory. They are able to have access to all users' passwords. For the sake of manageability and individual accountability, a strong suggestion is to have External Security Management (ESM), such as RACF, as the backend database to hold the system users' passwords. An ESM can provide tools that make the privileges acquired through the authentication process expirable and subject to periodic evaluation and validation.

We want to reinforce the importance of the rules regarding the reusable password and their complexity as well. Some companies still use weak or even the default passwords on systems—you should ensure that your company is not one of them.

Additional information and reasons why to use an External Security Management tool to help secure your system can be found in Chapter 2, “The z/VM security management support utilities” on page 3.

8.3.2 Choosing the z/VM privilege class

The most basic level of security that z/VM provides, and is sometimes forgotten by Linux administrators, is a user’s privilege classes. The CP commands that users can execute on a z/VM system depends on the privilege class they are part of. The list of classes and their meanings can be found in Table 2-1 on page 27.

As a starting point, have your Linux virtual guests configured to only have the general user privilege class, which means they will only have access to CP control functions over their own virtual machines and resources. A virtual Linux guest should not have additional privileges to define system-wide parameters of the z/VM system nor other virtual guest settings no matter how tempting you might think it is to be able to perform such tasks from one of the Linux guests.

The use of RACF can help reduce even more privileges that a user with privilege class G is able to execute. Although the use of the several available classes is not discussed in this book, we have described the process to activate and use some of them. For specific information about the classes and which resources each of them controls, see *z/VM: RACF Security Server Security Administrator’s Guide* (for V5R4.0), SC24-6142.

8.3.3 z/VM network connection

Many alternative ways exist to connect to a z/VM system: through the hardware management console (HMC), with the service elements, through a physical console connected to a terminal controller, with the OSA-Express integrated console controller, or through the company’s network. All the methods listed, except the network, have their benefits in terms of security because the access can be physically controlled and secured. Alternatively, it is not usual or practical to have every individual that requires access to the system walking into the data center to access the consoles. So, the usual configuration is to have the system accessible through the network.

Telnet is the protocol that z/VM uses for connecting remote users. This would be just fine, but this protocol’s most famous characteristic is the fact that the data flows across the network in the clear (no encryption), therefore if someone were to be listening on your network, sniffing for security vulnerabilities would be easy.

A strong suggestion is to have all Telnet communications between the 3270 terminals and z/VM going through a secured channel, such as SSL. For more information about setting up the z/VM Telnet server, and configuring the clients, see 2.5, “Securing network access to z/VM” on page 14.

8.4 Securing the data

Ensuring the protection and security of an organization's information can be considered part of the foundation for today's successful businesses. Having strong security protecting your data is not a luxury you would want to live without; it should be deployed with careful planning and thought, understanding all the security requirements and business needs of your organization.

When defining security policies, a wise consideration is to start with a tight security policy and then grant extra access and privileges as the technical solution or the business require.

Besides choosing a platform that is capable of meeting your organization's requirements, choosing the appropriate external security manager (ESM) is one of the first steps to take when implementing your security. As for secure access to the various resources on the System z platform, the Resource Access Control Facility (RACF) can help protect your Linux server resources against unauthorized access. For a discussion about ESM, see 2.6, "Securing z/VM resources" on page 27.

In this section, we discuss best practices for securing your information, determining who can and who cannot access data from inside Linux and from z/VM. Further, we discuss how you can reduce vulnerability to harmful code and binaries by having fewer intrusion points, and how to encrypt your data to prevent information theft.

8.4.1 Securing your minidisks

The z/VM system was designed in such a way that unless otherwise granted, access to minidisks is denied for users who do not have the correct privileges. When an ESM is not in use, you must define passwords on the system's directory for the disks you want to share. Although this sounds like a fairly secure approach, a good practice is to use an ESM to make your configuration more resilient and less error-prone. For this section, we assume that RACF is in use to secure your minidisks. However, we provide examples of how to implement security when that is not the case.

You can use RACF to control who can link to z/VM minidisks by using profiles in the VMMDISK resource class. z/VM calls RACF for an authorization check in the VMMDISK class for two events:

- LINK command: One user's attempt to link to another user's minidisk
- MDISK event: A user linking to his or her own minidisk

MDISK events occur at logon time when a user's MDISK directory statements are being processed, or when a user issues a LINK command for a self-owned minidisk.

If the VMMDISK class was not previously enabled on your system, doing so is mandatory, for the sake of security and integrity of your minidisks. To enable it, you can run the command shown in Example 8-1.

Example 8-1 Activating the VMMDISK RACF class

```
RAC SETROPTS CLASSACT(VMMDISK)
```

Note: If you are a Linux administrator with no previous RACF experience, you should have your RACF or z/VM administrator perform this task for you.

For performance or management reasons, certain situations might exist where having shared disks is recommended, but for the best practices covered in this chapter, we assume that all the disks hold sensitive information worth protecting against unauthorized access. For each of those disks, a RACF profile should exist that determines who can access it and what type of privileges a user or group should have. Assuming that your Linux user ID is LNXSU1, to list the existing profiles for the minidisks it owns is shown in Example 8-2.

Example 8-2 Listing existing minidisks RACF profiles

```
RAC SEARCH CLASS(VMMDISK) MASK(LNXSU1.)
LNXSU1.201
LNXSU1.202
LNXSU1.401
LNXSU1.402
LNXSU1.405
```

If no profiles exist for your minidisks, you can either define the profiles yourself or ask your RACF or z/VM administrator to create them for you. Example 8-3 shows the command to create the RACF profile. We created the profile for the minidisk by using virtual address 201 for user LNXSU1.

Example 8-3 Creating a VMMDISK profile

```
RAC RDEFINE VMMDISK LNXSU1.201 UACC(NONE)
```

Reference: For detailed information about RACF commands and implementation, see *z/VM: RACF Security Server Security Administrator's Guide* (for V5R4.0), SC24-6142.

Anyone (even those with the most restrictive access possible, READ) would be able to copy the data files to another minidisk where they have control of the security characteristics, and then downgrade the security restrictions on those files. So, a good practice is to initially assign a universal access authority (UACC) of NONE, and then selectively, as required, grant access to the smallest number of users and groups possible.

To change the UACC of a minidisk if the profile was not defined this way, you might execute the command shown in Example 8-4.

Example 8-4 Changing the UACC to a specific minidisk

```
RAC RALTER VMMDISK LNXSU1.201 UACC(NONE)
```

If you want to change the UACC to all minidisks owned by the LNXSU1 user ID, proceed with the command shown in Example 8-5.

Example 8-5 Changing the UACC to all minidisks for a user ID

```
RAC RALTER VMMDISK LNXSU1.* UACC(NONE)
```

Remember, if privileges were previously assigned to any specific user or group, they will not get revoked with this command. Therefore, performing a complete assessment of your user ID's minidisk profiles is important.

To check the privileges that are granted on the LNXSU1.201 profile, you can execute the command shown in Example 8-6 on page 270.

Example 8-6 Listing the attributes of a minidisk profile

```
RAC RLIST VMMDISK LNXSU1.201 ALL
```

After the initial minidisk security assessment is complete, you can start granting additional privileges to the users or groups as required. The possible access authorities for minidisks are NONE, READ, UPDATE, CONTROL, and ALTER.

Reference: For detailed information about the access authority for minidisks, refer to *z/VM: RACF Security Server Security Administrator's Guide* (for V5R4.0), SC24-6142

On a system with RACF enabled, and assuming you had already taken all the suggested actions described, if you had not explicitly allowed a user to access a minidisk, a CP LINK command would result in an error, as shown in Example 8-7.

Example 8-7 Linking a minidisk without proper privileges

```
CP LINK LNXSU1 201 201 RR
RPIMGR032E YOU ARE NOT AUTHORIZED TO LINK TO LNXSU1.201
HCPLNM298E LNXSU1 0201 not linked; request denied
```

Assuming you want another Linux guest, LNXSU2, to access disk address 201 on LNXSU1 with only permission to read and copy files, use the command shown in Example 8-8.

Example 8-8 Granting permission to access a minidisk

```
RAC PERMIT LNXSU1.201 CLASS(VMMDISK) ID(LNXSU2) ACCESS(READ)
```

Note: You can use the same command to grant privileges to groups.

If the user LNXSU2 no longer needs access to user LNXSU1's minidisk, you must remove the privilege by updating the RACF profile. This can be accomplished by running the PERMIT command with the DELETE operand, as shown in Example 8-9.

Example 8-9 Revoking privileges to access a minidisk

```
RAC PERMIT LNXSU1.201 CLASS(VMMDISK) ID(LNXSU2) DELETE
```

With only a few precautions, you can have your minidisks secured against unauthorized access at the VM level. Using RACF, you can control who can or cannot access your disks, and you can also determine several of their access levels. In addition, RACF can also capture logs for audit purposes.

8.4.2 Reducing the intrusion points with shared disks

One golden rule of information management is that information should only exist in one location. This is easy to accomplish when you deal with applications that retrieve data from a database. But how can one handle a requirement to have files available across several servers and still maintain data integrity and keep data safe from unauthorized access?

On a distributed server farm, this task is accomplished by using a network service such as Network File System (NFS). However, because the data, confidential or not, would flow through the network, such a solution requires additional security tools to eliminate or mitigate risks. Linux on System z can take advantage of z/VM features for this purpose. One such feature is the ability to virtually connect devices among guests within the same system. A possibility is to have a disk or a set of disks (an LVM group for example) shared among multiple servers without the need for network services of any kind. Your data will not flow across the network and consequently cannot be sniffed, reinforcing the overall security of your environment and reducing the number of intrusion points.

The ability to share disks among multiple servers is one of the values added by the platform, but you still must determine which file system will go on the top of those disks. z/VM can have the disks shared in read-write mode to all guests within the system, but you are not responsible for determining how the Linux systems will control the I/O lock to avoid corruption of the file system, eventually leading to loss of data integrity.

If the ability to have multiple nodes accessing the same disks in read-write mode is a requirement, you might want to evaluate the usage of a clustered file system. Several are available to be used with Linux, but they are not discussed in this book. If you want to provide access to static content, such as the installation binaries for a specific software product, a static file system that is using z/VM minidisks linked in read-only mode is definitely worth implementing. It will reduce the overall complexity and risk exposure of serving multiple copies of the same files across your servers.

If you do not have RACF enabled on your system, and you need to share disks between virtual servers, you must determine which disk contains the data and then connect it to the target nodes.

As described in “Protecting access to minidisks” on page 29, z/VM can grant or deny access to the minidisks, depending on the password that is defined for that specific resource in the user directory. If you try to link a minidisk protected by a password without the appropriate password, an error message is issued, as shown in Figure 8-1.

```
[root@lnxrh1 /]# vmcp link lnxsu1 300 2300 rr write
HCPLNM114E LNXSU1 0300 not linked; mode or password incorrect
Error: non-zero CP response for command 'LINK LNXSU1 300 2300 RR WRITE': #114
```

Figure 8-1 Linking a minidisk with the wrong password

When ESM is not used, manageability is seriously compromised. Although the access to the user directory is protected and restricted to users with privileged access, it is still a clear text file, so the use of an ESM is strongly suggested. The efforts to micro-manage a large number of devices makes the whole process prone to human errors and, as a consequence, a lot more subject to security exposures.

The adoption of an ESM such as RACF, which was selected for this book, is considered a best practice for securing Linux servers on the System z platform. Granting access to a minidisk with RACF can be accomplished by issuing the command shown in Example 8-10.

Example 8-10 Allowing read access to a minidisk

```
RAC PERMIT LNXSU1.300 CLASS(VMMDISK) ID(LNXRH1) ACCESS(READ)
```

After you have granted permission for LNXRH1 to link to LNXSU1’s 300 disk, you can proceed with the command to link the minidisk. No password is needed. RACF controls the access to the resource and guarantees that only authorized users access the resource in a permitted mode.

After completing this process, you can have as many virtual guests existing on the same z/VM as you need, sharing the same minidisk. As seen in Figure 8-2 on page 272 and

Figure 8-3 on page 272, you only have to bring the disk online in Read Only mode and make it available.

Figure 8-2 shows an example on LNXSU1.

```
lnxsu1:/ # df -h /mnt/
Filesystem            Size  Used Avail Use% Mounted on
/dev/dasdc1            69M   40K   65M   1% /mnt
```

Figure 8-2 Minidisk mounted R/W on the source node

Figure 8-3 shows an example on LNXRH1.

```
[root@lnxrh1 ~]# vmcp link lnxsu1 300 300 rr
DASD 0300 LINKED R/O; R/W BY LNXSU1
[root@lnxrh1 ~]# vmcp q 300
DASD 0300 3390 LXC40C R/O       100 CYL ON DASD  C40C SUBCHANNEL = 0011
[root@lnxrh1 ~]# chccwdev -e 0.0.0300
Setting device 0.0.0300 online
Done
[root@lnxrh1 ~]# lsdasd | grep 0.0.0300
0.0.0300   active      dasdd     94:12   ECKD  4096   70MB      18000
[root@lnxrh1 ~]# mount -r /dev/dasdd1 /mnt/
[root@lnxrh1 ~]# df -h /mnt/
Filesystem            Size  Used Avail Use% Mounted on
/dev/dasdd1            69M   40K   65M   1% /mnt
```

Figure 8-3 Minidisk mounted R/O at the target node

Note: You will not be able to perform the process that we have described by using Journalling file systems because these must be updated on every mount. This process, in turn, would generate errors, preventing the successful mount because of journal inconsistencies. Consequently, we used the `ext2` file system when running the tests for this chapter.

8.4.3 Protecting the data with encrypted file systems

Businesses today have to deal with an increasing number of security challenges. Organizations need to protect themselves against known and unknown security threats to one of their most valuable assets: information. In addition, in many regions, the adoption of security mechanisms to protect data has been made obligatory by legislation.

The use of data encryption has been pointed out as one way to protect information. In general, the definition for the term *encryption* is the transformation of plain text data into encrypted data by using a key. Without the key, data cannot be converted back to plain text. Consequently, the proper handling and implementation of encryption is extremely important. Failing to properly implement key management can result in an encryption deadlock, leading to permanent loss of all encrypted data.

One of the most challenging decisions to make is on what to protect through encryption and what are the trade-offs in terms of performance, legislation and regulatory compliance, and security requirements. From our experience, encrypt everything that is required from a business perspective regarding data availability, performance, and disaster recovery.

An important highlight is that the System z platform leverages advantages through the use of cryptographic hardware and cryptographic functions that are built-in to the central processor (CP). Through the use of Central Processor Assist for Cryptographic Function (CPACF), it is possible to offload cryptographic cipher calculations from the central processor, thus considerably reducing the processor cycles of such operations compared to the cost of having them done through software emulation. This has to be considered when you size the hardware requirements for data encryption. The possibility to have this function enabled in all the servers on your virtual farm is a unique feature to help in evaluating which data to encrypt and how much of the resources are needed to do so.

For specific information about how to encrypt data on disks, using the Linux functions available on all the major distributions, see "File system encryption" on page 155.

The dm-crypt subsystem in the Linux kernel is implemented as a device mapper that can be stacked on top of other devices that are managed through the device mapper framework. This means that it is possible to encrypt everything from entire disks to software RAID volumes and LVM logical volumes, adding a high level of flexibility to your encryption strategy, compared to cryptoloop, the predecessor subsystem of dm-crypt.

We suggest the use of the dm-crypt disk encrypt subsystem for flexibility and resilience. The dm-crypt is broadly supported on most Linux distributions, which makes it a reliable solution when the continuous development of the technology is considered. Further instructions of how to set up and enable encryption through dm-crypt is in "dm-crypt" on page 158.

You could consider having all your disks encrypted at a lower level in your storage array subsystem. This is not a subject of discussion in this book, but additional information about the disk-based encryption provided with the IBM DS8000® subsystem is in *IBM System Storage DS8000 Disk Encryption Implementation and Usage Guidelines*, REDP-4500.

8.5 Securing the network

The System z platform provides a great security advantage when it comes to virtual networking capabilities. Using industry standards and providing a variety of tools to control who can or cannot couple to the virtual devices, System z also provides auditing and troubleshooting functionality. Using Linux on System z guests with correct connectivity can dramatically reduce physical intrusion points, making the physical infrastructure easier to maintain.

z/VM offers several possible network configurations, such as HiperSockets adapters, guest LAN, and virtual switches. The configuration of these resources is not detailed in this chapter; for specific configuration details and system requirements, see *z/VM V6R2.0 Connectivity*, SC24-6174-02. Although guest LANs might be considered as a viable alternative to provide connectivity among virtual servers, we have chosen to use virtual switches instead of any other z/VM networking options. The features that this technology offers are part of the best practices we discuss in this section. Information about how to use z/VM native functions to protect guest LANs are in "Protecting z/VM guest LANs" on page 28.

To illustrate how to create your virtual network infrastructure in a secure way, using both the native security and an external security manager (ESM) such as RACF, see the scenario in Figure 8-4 on page 274. It shows that four servers are configured to support an application: two act as web servers facing the Internet, and two act as application servers with tighter security requirements. Observe that for this first scenario, the only way to guarantee that the data flowing from a server on an Internet-facing exposed network to a more secure network zone is by separating them on two different subnets. We also assume that the VSWITCH is operating in the data layer.

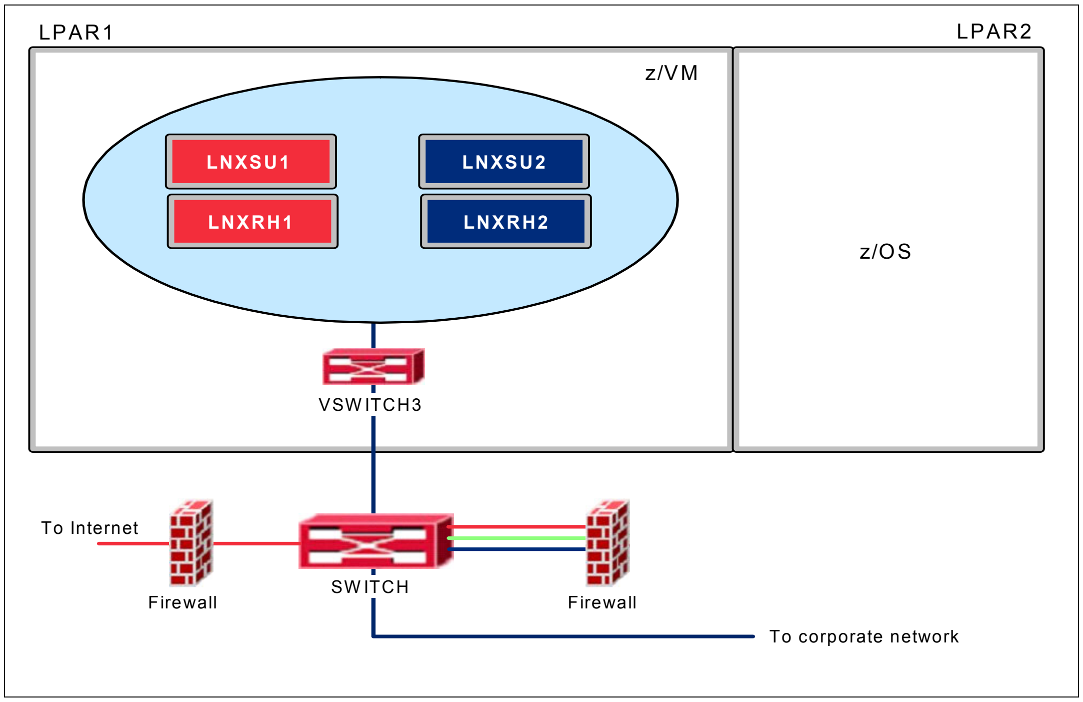


Figure 8-4 VSWITCH infrastructure

8.5.1 Securing the virtual switches

z/VM default configuration automatically prevents users from coupling to a VSWITCH if not explicitly allowed. To create a layer 2 virtual switch, you may use the command syntax shown in Example 8-11.

Example 8-11 Creating a Layer2 VSWITCH

```
DEFINE VSWITCH VSWITCH3 RDEV 3083 30A3 ETHERNET CONTROLLER *
```

This command produces the output shown in Figure 8-5.

```
VSWITCH SYSTEM VSWITCH3 Type: VSWITCH Connected: 0    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    ETHERNET                  Accounting: OFF
  VLAN Unaware
  MAC address: 02-00-00-00-00-0A
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  Isolation Status: OFF
  RDEV: 3083.P00 VDEV: 3083 Controller: DTCVSW1
  RDEV: 30A3.P00 VDEV: 30A3 Controller: DTCVSW2  BACKUP
```

Figure 8-5 VSWITCH configuration output

If, upon creation, no VLAN ID is specified, a VLAN-Unaware switch is created and all virtual guests coupled to it will be on the same segment (Figure 8-6 on page 275).

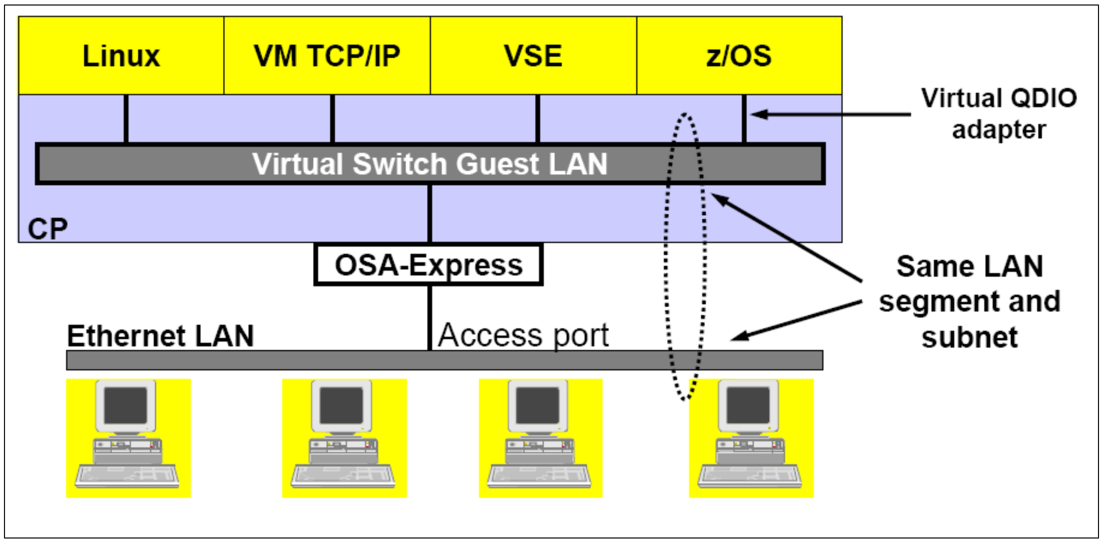


Figure 8-6 VLAN-Unaware VSWITCH infrastructure

Without any previous permission, if you try to couple to the VSWITCH from your Linux server, an error occurs, as shown in Figure 8-7. The error indicates that you do not have the necessary privileges to couple to the device.

```
lnxsu1:~ # vmcp couple 700 system VSWITCH3
HCPNDF6011E You are not authorized to COUPLE to SYSTEM VSWITCH3
Error: non-zero CP response for command 'COUPLE 700 SYSTEM VSWITCH3': #6011
```

Figure 8-7 Attempt to couple to a VSWITCH without privileges

The command you can use to grant the user LNXSU1 privileges to couple to the VSWITCH, if you are not using an external systems manager such as RACF, is shown in Example 8-12.

Example 8-12 Granting privileges for a user ID to couple to the VSWITCH

```
SET VSWITCH VSWITCH3 GRANT LNXSU1
```

Similarly, if you have to revoke, from any user ID, a permission that was previously granted, use the syntax in Example 8-13.

Example 8-13 Revoking privileges to couple to a VSWITCH

```
SET VSWITCH VSWITCH3 REVOKE LNXSU1
```

If you have to assess the list of users with permission to couple to the VSWITCH, Example 8-14 shows the syntax to use.

Example 8-14 Listing a VSWITCH's access list

```
QUERY VSWITCH VSWITCH3 ACCESSLIST
```

This command produces output shown in Figure 8-8 on page 276. Observe that SYSTEM is automatically granted access to the resource.

```
VSWITCH SYSTEM VSWITCH3 Type: VSWITCH Connected: 0    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    ETHERNET                  Accounting: OFF
  VLAN Unaware
  MAC address: 02-00-00-00-00-0A
  State: Ready
  IPTimeout: 5        QueueStorage: 8
  Isolation Status: OFF
    Authorized userids:
      LNXSU1   SYSTEM
  RDEV: 3083.P00 VDEV: 3083 Controller: DTCVSW1
  RDEV: 30A3.P00 VDEV: 30A3 Controller: DTCVSW2  BACKUP
```

Figure 8-8 VSWITCH access list

After the correct permission is granted to the user ID, in this example LNXSU1, you can run the command to dynamically couple the network interface card (NIC) previously defined to the virtual switch. See Figure 8-9.

```
lnxsu1:~ # vmcp couple 700 system VSWITCH3
NIC 0700 is connected to VSWITCH SYSTEM VSWITCH3
```

Figure 8-9 Coupling to a VSWITCH with the correct privileges granted

Note: For specific information about how to define the virtual switches and NICs, see *z/VM V6R2.0 Connectivity*, SC24-6174-02.

RACF can improve the security requirements, managing policies to grant appropriate access to your system resources, and can also reduce the risks of security breaches by reducing the number of interactions needed to maintain system integrity.

RACF can be used to protect guest LANs and virtual switches by using profiles in the VMLAN class. From an access control perspective, the two are treated the same way. This class contains two sets of profiles to secure access to the network devices:

- A base profile that controls the ability of a user ID to connect to a virtual switch
- An IEEE VLAN-aware virtual switch profile that can be used to assign a user to one or more VLANs

If the VMLAN class is not enabled on your system, you must enable it, for the sake of security and integrity of your network resources. Example 8-15 shows the command to activate the class.

Example 8-15 Activating the VMLAN RACF class

```
RAC SETROPTS CLASSACT(VMLAN)
```

Note: If you are a Linux administrator with no previous RACF experience, be sure to have your RACF or z/VM administrator performing this task for you.

The base profiles are named `userid.name`, where `userid` is the name of the owner of the resource and `name` is the name of the LAN. Because the scenarios we are evaluating contain a virtual switch, not a guest LAN, an important note is that, in the case of a VSWITCH, the `userid` will always be SYSTEM.

The profile for VSWITCH3 is shown in Figure 8-10.

```
RAC SEARCH CLASS(VMLAN) MASK(SYSTEM.VSWITCH3)
SYSTEM.VSWITCH3
```

Figure 8-10 Listing the base profile for a VSWITCH

If a RACF profile is not created for your VSWITCH, you can create it by using the command in Example 8-16.

Example 8-16 Defining the RACF base profile for a VSWITCH

```
RAC RDEFINE VMLAN SYSTEM.VSWITCH3 UACC(NONE)
```

Note: For the same reasons we did this for the minidisks, the universal access authority (UACC) was defined as NONE.

When RACF controls the access to the system resources, having all the users you want to couple to the VSWITCH individually listed in the VSWITCH access list is unnecessary, because the ESM overrides the CP access list. Only SYSTEM remains in the VSWITCH access list; it is automatically added to the list upon creation of the resource.

Using RACF, the list of users must exist on the ESM database, which for obvious reasons reduces the chances of a leakage of specific information about your network topology, because the information is not kept in a plain text file.

Example 8-17 shows the command that should be executed to grant access to our LNXSU1 user to the VSWITCH3 VSWITCH, currently managed by RACF.

Example 8-17 Granting update access to a base vmlan RACF profile

```
RAC PERMIT SYSTEM.VSWITCH3 ID(LNXSU1) CLASS(VMLAN) ACCESS(UPDATE)
```

Note: The UPDATE access is required so that the user can couple to a VSWITCH that is managed by RACF.

With the process we have described, no matter how much better and less error prone the process is, a necessary step is still to update the device base profile for each of the servers that we want to grant access to the resource.

As a best practice for the profile maintenance, grant the desired privileges to the virtual guests based on RACF groups. The use of such groups can also help to perform a security audit on your systems. Determining the privileges a given user has is possible simply by determining the user’s privilege class and the groups that the user is a member of.

Be sure that RACF group list checking is active by issuing the command in Example 8-18.

Example 8-18 Activating RACF group list checking

```
RAC SETROPTS GRPLIST
```

A good strategy to manage who can or who cannot couple to a specific virtual switch is as follows:

1. Create a RACF group to hold the list of virtual guests that will be allowed to couple to your VSWITCH. In Example 8-19, we created a RACF group and named it VSW3GRP.

   Example 8-19 Creating a RACF group

   ```
   RAC ADDGROUP VSW3GRP
   ```

2. Update your VSWITCH base profile with the privileges that you want granted to this group, as shown in Example 8-20. The syntax is the same used in Example 8-17 on page 277, using the group VSW3GRP in the ID field.

   Example 8-20 Granting a RACF group access to a VSWITCH

   ```
   RAC PERMIT SYSTEM.VSWITCH3 ID(VSW3GRP) CLASS(VMLAN) ACCESS(UPDATE)
   ```

3. As a final step, connect the user, requiring access to this virtual switch, to the group, as shown in Example 8-21.

   Example 8-21 Connecting a user to a RACF group

   ```
   RAC CONNECT LNXSU1 GROUP(VSW3GRP)
   ```

All new virtual guests that are to be allowed to couple to the VSWITCH need to be connected only to this group. This step greatly reduces the maintenance efforts and makes the overall access lists audit-ready.

8.5.2 Virtual switch using VLAN tagging

The use of VLANs increases traffic throughput and reduces overhead by allowing the network to be organized by traffic patterns and not based on the physical locations of the servers. Because z/VM virtual networking capabilities are compatible with the standard specifications for virtual LAN tagging, the Linux virtual servers coupled to a VSWITCH can belong to VLANs that extend beyond z/VM's virtual network. See Figure 8-11.

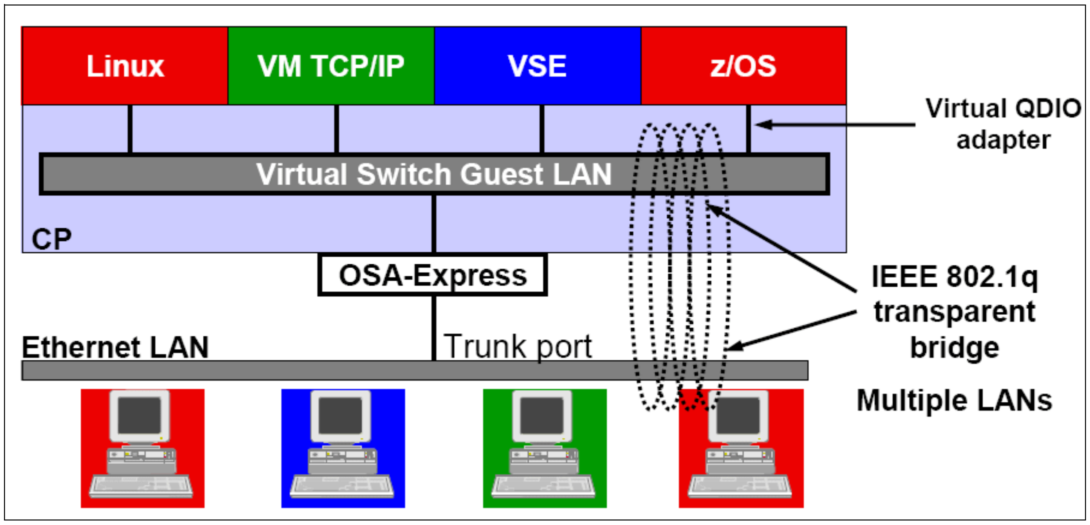


Figure 8-11 VLAN-aware VSWITCH infrastructure

From a security perspective this infrastructure makes the whole environment safer, reducing the broadcast domain, and the servers become less exposed to network sniffers and port scanners. It also allows the data networks to be separated from management networks, which is desirable and should be mandatory to reinforce the access control at the network level. The servers should only be able to reach exactly what they need, nothing else. With this, the exposure and the security requirements for other access and authentication controls are greatly reduced. Having a VLAN-aware VSWITCH, if it can be accommodated by your network topology, is strongly advised from a best practices perspective.

Enabling the Linux guests to couple to a VLAN-aware VSWITCH does not differ from what we have discussed so far. When no ESM is in place, and the native CP is being used to manage the virtual device access list, a few options are available to be used with the SET or the MODIFY commands. Table 8-1 lists options to be used with VLAN-aware virtual switches.

Table 8-1 VLAN-specific attributes for defining a VSWITCH

Attribute	Options
PORTType	**ACCESS | TRUNK** Defines the type of connections that are established by the user ID to be either ACCESS or TRUNK. For VLAN-unaware guests, use the ACCESS port type; for VLAN-aware virtual guests, use the TRUNK port type.
VLAN	**VLAN ID** Identifies the LAN ID to be used to restrict traffic across the adapter. The user ID is connected to the VSWITCH.
NATive	Defines which VLAN ID will be used to identify all non-tagged traffic. For most network equipment, VLAN 1 is used for this purpose.

Note: For all available options for the VSWITCH definition, refer to *z/VM V6R2.0 Connectivity*, SC24-6174-02.

The use of a VLAN-aware VSWITCH enables the network to be segmented, not at the IP layer, but at the data-link layer, which provides flexibility to network administrators and security administrators. The scenario is shown in Figure 8-12 on page 280.

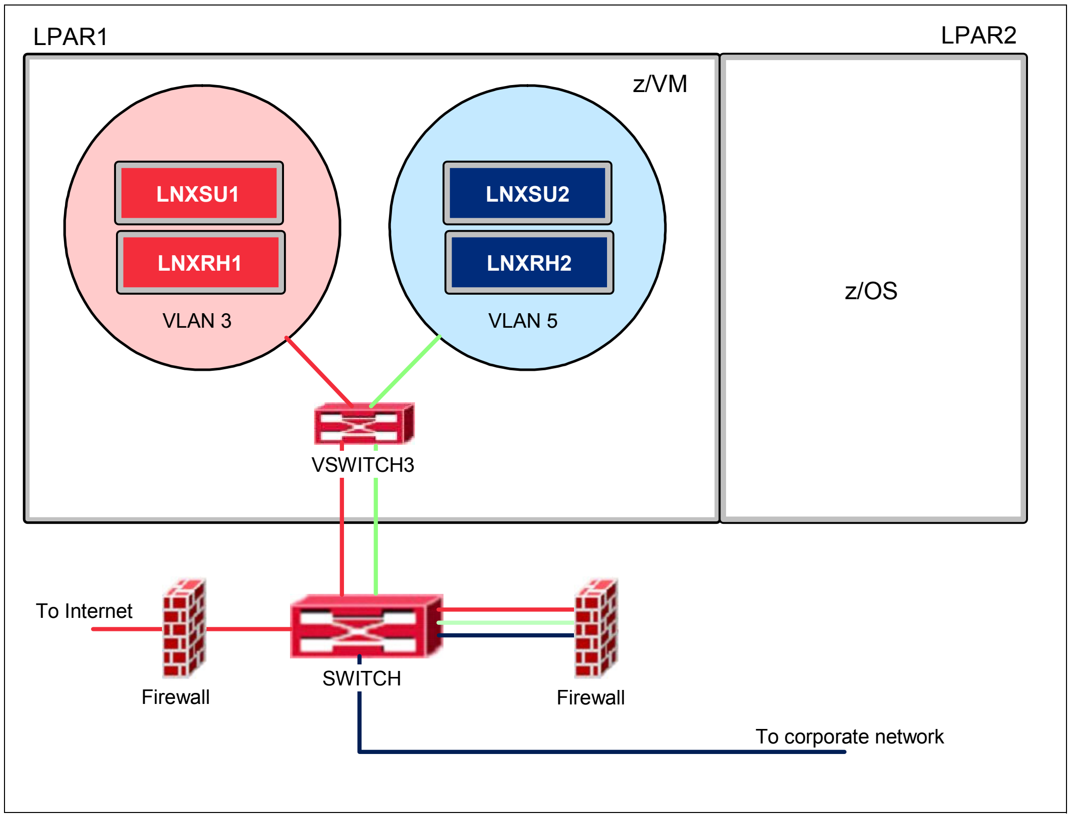


Figure 8-12 VLAN-aware VSWITCH infrastructure

Example 8-22 shows how to create the VLAN-aware VSWITCH for the scenario we have discussed.

Example 8-22 Creating a VLAN-aware VSWITCH

```
DEFINE VSWITCH VSWITCH3 RDEV 3083 30A3 VLAN 3 NATIVE 1 ETHERNET CONTROLLER *
PORTTYPE ACCESS
```

Figure 8-13 shows the configuration that was created as a result of this command.

```
VSWITCH SYSTEM VSWITCH3 Type: VSWITCH Connected: 2    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    ETHERNET                  Accounting: OFF
  VLAN Aware  Default VLAN: 0003    Default Porttype: Access   GVRP: Enabled
              Native  VLAN: 0001 VLAN Counters: OFF
  MAC address: 02-00-00-00-00-0A
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  Isolation Status: OFF
  RDEV: 3083.P00 VDEV: 3083 Controller: DTCVSW1
  RDEV: 30A3.P00 VDEV: 30A3 Controller: DTCVSW2  BACKUP
```

Figure 8-13 VLAN-aware configuration

A best practice is to have all non-tagged traffic flowing across a VLAN-aware VSWITCH confined on a specific VLAN. That is why the native VLAN is configured to 1 on VSWITCH3; all non-tagged traffic will be tagged as 1 and can be easily found from a network administrator station. If the NATive attribute is not specified, the native VLAN ID is set to the default VLAN ID, and all loose traffic will be on the same logical segment as the genuine data traffic.

Upon the VSWITCH creation, the CP access list is empty, which means that nobody, except for the SYSTEM itself, will be able to couple to it. When granting access to the switch is required, use the SET or MODIFY command to grant the virtual guest privileges to couple to the device that is using the VLAN, and the port type, depending on whether the guest is stripping VLAN IDs from the frames that flow through its interfaces. In Example 8-23, we grant privileges to LNXSU1 to couple to VSWITCH3 using VLAN ID 0003 and the ACCESS port type.

Example 8-23 Updating CP access list granting access to a virtual guest to a VLAN-aware VSWITCH

```
SET VSWITCH VSWITCH3 GRANT LNXSU1 VLAN 3 PORTTYPE ACCESS
```

Note: If not specified, the user ID for which the access is being granted will automatically get the default VLAN ID and the default port type.

When working with RACF, a VLAN ID-qualified profile, named for the VSWITCH name and the VLAN ID, must exist. This profile defines who can have access to a specific VLAN on that VSWITCH. The best way to handle this configuration is by managing RACF groups. For the VSWITCH3 (the VSWITCH that is using VLAN 0003), the profile is shown in Figure 8-14.

```
RAC SEARCH CLASS(VMLAN) MASK(SYSTEM.VSWITCH3.0003)
SYSTEM.VSWITCH3.0003
```

Figure 8-14 Listing the IEEE VLAN-aware VSWITCH profile

If the profile does not exist, you can create it with the command in Example 8-24. Remember to always have the universal access authority (UACC) defined to NONE to avoid unwanted access to the device.

Example 8-24 Defining the IEEE VLAN-aware VSWITCH profile

```
RAC RDEFINE VMLAN SYSTEM.VSWITCH3.0003 UACC(NONE)
```

If a base profile grants a user access to a VLAN-aware virtual switch but the user is not permitted to any VLAN ID-qualified profiles for that virtual switch, RACF directs z/VM to authorize the user only to the default VLAN ID that is configured for the virtual switch. The administrator is responsible for guaranteeing consistency between the base profiles and the VLAN ID-qualified specific profiles.

Using the same strategy we used with the VLAN-unaware VSWITCH, we can create a group to hold all users that you need to authorize access to VLAN 0003. You can name the groups the way you find appropriate; they do not have to follow a predefined convention. For the sake of consistency and also to more easily identify what the group is being used for, we name it VLAN0003.

Create the new group using the same command syntax we used in Example 8-20 on page 278, replacing the group VSW3GRP with VLAN0003, and then giving this new group the same UPDATE privilege on the RACF profile for VLAN0003 (`SYSTEM.VSWITCH3.0003`). Use the same command as in Example 8-21 on page 278.

When completed, coupling as many virtual guests as you need to the virtual switch is possible simply by connecting them to the RACF group with the command that is used in Example 8-22 on page 280.

All new guests coupling to VSWITCH3 using VLAN0003 must be on both groups. Having the users only connected to the VLAN0003 group does not imply that the user will have privileges to couple to the virtual switch VSWITCH3.

The IEEE VLAN-aware VSWITCH profile uses whatever is defined for the default port type. If you have to mix port types within the same VSWITCH, you must update the CP access list also. Although this configuration is not common, we mention it because the RACF profile may override the CP access list, so you should take extra care if a requirement exists to have the access lists mixed.

Figure 8-15 shows the summary of commands that were used, assuming that RACF is performing the access control of the devices, to build the virtual network infrastructure shown in Figure 8-15.

```
DEFINE VSWITCH VSWITCH3 RDEV 3083 30A3 VLAN 3 NATIVE 1 ETHERNET CONTROLLER *
PORTTYPE ACCESS
RAC ADDGROUP VSW3GRP
RAC ADDGROUP VLAN0003
RAC ADDGROUP VLAN0005
RAC RDEFINE VMLAN SYSTEM.VSWITCH3 UACC(NONE)
RAC RDEFINE VMLAN SYSTEM.VSWITCH3.0003 UACC(NONE)
RAC RDEFINE VMLAN SYSTEM.VSWITCH3.0005 UACC(NONE)
RAC PERMIT SYSTEM.VSWITCH3 CLASS(VMLAN) ID(VSW3GRP) ACCESS(UPDATE)
RAC PERMIT SYSTEM.VSWITCH3.0003 CLASS(VMLAN) ID(VLAN0003) ACCESS(UPDATE)
RAC PERMIT SYSTEM.VSWITCH3.0005 CLASS(VMLAN) ID(VLAN0005) ACCESS(UPDATE)
RAC CONNECT LNXSU1 GROUP(VSW3GRP)
RAC CONNECT LNXSU2 GROUP(VSW3GRP)
RAC CONNECT LNXRH1 GROUP(VSW3GRP)
RAC CONNECT LNXRH2 GROUP(VSW3GRP)
RAC CONNECT LNXSU1 GROUP(VLAN0003)
RAC CONNECT LNXSU2 GROUP(VLAN0005)
RAC CONNECT LNXRH1 GROUP(VLAN0003)
RAC CONNECT LNXRH2 GROUP(VLAN0005)
```

Figure 8-15 Summary of commands

With this example we assume that the virtual guests already have a network interface ready to couple to the VSWITCH VSWITCH3 and properly configured as far as IP addresses and VLANs are concerned.

Note: Using RACF, when you permit the same virtual guest on two different VLAN profiles, the port type that this one profile will use to couple to the VSWITCH will automatically be assumed as TRUNK no matter what you have configured as the default port type. This statement means that you must have your virtual server stripping the frames for the VLANs it is supposed to communicate with.

8.5.3 Virtual switch port isolation

A virtual switch and an OSA-Express card port can be shared by multiple TCP/IP stacks. This means that two servers hosted on the same hardware box can reach each other and the packets would be directly routed to the sharing TCP/IP stack without transversing the external network. See Figure 8-16 on page 283.

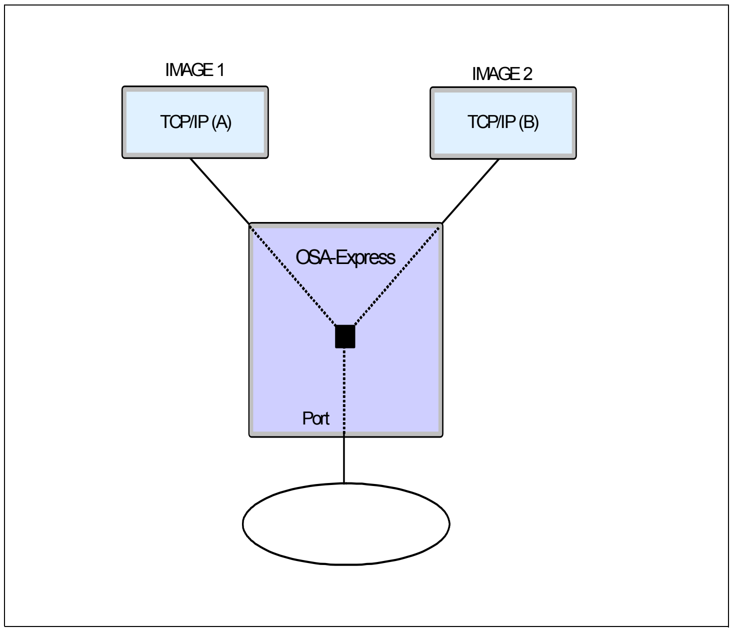


Figure 8-16 OSA-Express port sharing

The default configuration allows full connectivity to all sharing hosts and LPARs that are connected to the same OSA-Express port and between all guest ports on a virtual switch. The communications take place within the boundaries of the central processing complex (CPC).

Although the ability to do image-to-imagine communication can represent a great advantage from a performance standpoint, it might also represent a problem from the security perspective. Some organizations might have a security strategy that all communications must go through an external device.

A feature available on z/VM addresses this situation by completely isolating local traffic, preventing one virtual guest from being able to reach others without going outside the OSA-Express port. From a security perspective, this isolation feature can help to completely isolate servers that are located on exposed network zones, and still maintain a uniform network addressing schema. This process can potentially reduce the control check points, because there would be no need to rely on a layer-3 device (router) to block the traffic using access control lists (ACLs); the guests coupling to a specific virtual switch with isolation turned on would not be able to see each other through the OSA-Express port. The same subnets could be completely isolated from each other at the layer-2 level, reducing the load on firewalls and core network devices.

This function provides an extra assurance against a misconfiguration, which might otherwise allow image-to-image traffic where not desirable, and also ensures that traffic flowing through the OSA-Express does not bypass any security features implemented in the overall network.

To turn on the isolation mode on a virtual switch, run the command shown in Example 8-25.

Example 8-25 Turning on VSWITCH isolation

```
SET VSWITCH VSWITCH3 ISOLATION ON
```

Installations at sites that want the virtual switch ports to be isolated only from local traffic can simply turn the isolation mode on. If the traffic control must be done by an external ACL or firewall, the necessary routes must exist on the virtual guests to be able to forward the traffic to the appropriate gateways.

With the isolation mode turned on, if you want to allow traffic from one virtual guest to another if they are sharing the same subnet, configure each TCP/IP stack on a different VLAN.

An important aspect to note is that if you need to turn on *promiscuous mode* to perform debugging or analysis, you must ensure that it will not affect the guest for whom the promiscuous mode is activated. This specific guest will receive a copy of the guest-to-guest internal traffic when the virtual switch is operating in either of the two isolation modes.

8.5.4 Network diagnostics

Another feature of z/VM is the ability to decide who can put their interfaces in promiscuous mode, thus adding the flexibility to allow Linux users to take advantage of native network capture and troubleshooting tools when necessary.

One method of troubleshooting a network problem is by intercepting all data flowing over the network regardless of destination MAC or IP address. To be able to do this, the network interface card must be configured in promiscuous mode, which means that this specific virtual machine would receive a copy of all data flowing on its configured LAN or VLANs.

As stated previously, to avoid data leakage, the default configuration for both z/VM guest LAN and VSWITCHs prevents users without previous permission to put their interfaces in promiscuous mode.

If you want to take advantage of the Linux native network diagnostic tools available, such as **`tcpdump`** or **`ethereal`**, you have to allow virtual guests to put their interfaces in promiscuous mode. To do so, grant access to couple to the VSWITCH using this mode. In Example 8-26, we allow the virtual guest LNXSU1 to couple to the virtual switch VSWITCH3 using promiscuous mode.

Example 8-26 Allowing promiscuous mode

```
SET VSWITCH VSWITCH3 GRANT LNXSU1 PRO
```

A suggested practice is not to allow any guests to couple to a virtual device in promiscuous mode on a permanent basis, unless the device is a monitoring station where other security mechanisms are in place to provide security against unauthorized access.

After finishing the troubleshooting, to revoke the privileges granted you can run the command in Example 8-27.

Example 8-27 Revoking promiscuous mode

```
SET VSWITCH VSWITCH3 GRANT LNXSU1 NOPRO
```

A command is available to activate the network interface in promiscuous mode if the operating system device driver does not support z/VM promiscuous mode. In Example 8-28, we activate the network interface card by using virtual address 700 from the Linux console, using the vmcp tool.

Example 8-28 Setting the interface in promiscuous mode

```
vmcp set nic 0702 pro
```

Note: Promiscuous mode must be used on a data device.

Recent Linux distribution device drivers support z/VM promiscuous mode. Therefore, putting the interface in this state should be possible. Your preferred diagnostic tool is likely able to do this for you.

After you start network data capturing, you can check the condition of your network card with the command shown in Example 8-29, where we have assumed that your network card virtual address is 700.

Example 8-29 Querying network interface details

```
vmcp query nic 700 detail
```

This example produces the output shown in Figure 8-17.

```
lnxsu1:~ # vmcp query nic 700 detail
Adapter 0700.P00 Type: QDIO      Name: UNASSIGNED  Devices: 3
  MAC: 02-00-00-00-00-09         VSWITCH: SYSTEM VSWITCH3
      RX Packets: 1         Discarded: 0         Errors: 0
      TX Packets: 0         Discarded: 13        Errors: 0
      RX Bytes: 42                  TX Bytes: 0
  Connection Name: HALLOLE   State: Session Established
      Device: 0700  Unit: 000   Role: CTL-READ
      Device: 0701  Unit: 001   Role: CTL-WRITE
      Device: 0702  Unit: 002   Role: DATA      vPort: 0079  Index: 0079
      Options: Ethernet Broadcast Promiscuous
        Unicast MAC Addresses:
          02-00-00-00-00-09
        Multicast MAC Addresses:
          01-00-5E-00-00-01
          33-33-00-00-00-01
          33-33-00-00-02-02
          33-33-FF-00-00-09
```

Figure 8-17 Details for a NIC in promiscuous mode

If the virtual guest has no privileges to turn on promiscuous mode, the same command produces different output, shown in Figure 8-18 on page 286.

```
lnxsu1:~ # vmcp query nic 700 detail
Adapter 0700.P00 Type: QDIO      Name: UNASSIGNED  Devices: 3
  MAC: 02-00-00-00-00-09         VSWITCH: SYSTEM VSWITCH3
      RX Packets: 1484       Discarded: 0         Errors: 0
      TX Packets: 0          Discarded: 11        Errors: 0
      RX Bytes: 86400                TX Bytes: 0
  Connection Name: HALLOLE   State: Session Established
      Device: 0700  Unit: 000   Role: CTL-READ
      Device: 0701  Unit: 001   Role: CTL-WRITE
      Device: 0702  Unit: 002   Role: DATA       vPort: 0077  Index: 0077
      Options: Ethernet Broadcast Promiscuous_Denied
        Unicast MAC Addresses:
          02-00-00-00-00-09
        Multicast MAC Addresses:
          01-00-5E-00-00-01
          33-33-00-00-00-01
          33-33-00-00-02-02
          33-33-FF-00-00-09
```

Figure 8-18 Details for a NIC not authorized to turn promiscuous mode on

As you can see, if the appropriate authority had not been granted, the Linux device driver tries to put the interface in promiscuous mode, but the request will be denied by CP. You are likely to receive a deceiving message from your diagnostics tool saying that the interface is in promiscuous mode. The only really effective way to check the status of the interface is by querying either the status of the virtual interface or the detailed status of the virtual switch.

Note: Promiscuous mode can only be set for a QDIO NIC and is not supported on HiperSockets. For assistance debugging HiperSockets, check how to use TRSOURCE in *z/VM V6R2.0 Connectivity*, SC24-6174-02.

When RACF is actively controlling the network resources, as said before, it can override the privileges previously granted on the CP access list. So, to grant privileges to turn promiscuous mode on for a specific user, update the virtual switch base profile, granting the user ID control access to the resource.

To keep consistency with the best practices described until now, and also to make your systems easier to audit and to perform user privilege revalidation, we suggest the use of groups when possible.

You can create the RACF group by using the command shown in Example 8-20 on page 278, and then connect the user that you want to this new group. In Example 8-30, we assume that a new group named SNIFF was created and the virtual user ID LNXSU1 was connected to this group.

Example 8-30 Allowing a RACF group to turn promiscuous mode on

```
RAC PERMIT SYSTEM.VSWITCH3 CLASS(VMLAN) ID(SNIFF) ACCESS(CONTROL)
```

As said previously, allowing users to be able to listen to the network on a permanent basis is not a good practice. Therefore, after the diagnostics are finished, you can simply revoke the privilege you had granted to the user LNXSU1 by removing the user from the special SNIFF group that was created for this purpose. See Example 8-31 on page 287.

Example 8-31 Removing a user from a RACF group

```
RAC REMOVE LNXSU1 GROUP(SNIFF)
```

Note: To be able to turn promiscuous mode on when using RACF, first uncouple and then couple the NIC back to the VSWITCH, after you update the virtual switch base profile.

8.5.5 Switch off backchannel communication

Other measures must be considered when you secure your virtual network. Those measures are intended to prevent users with high security clearance within a specific Linux server from being able to bypass the security rules and policies that apply to the whole network infrastructure. The measures to switch off backchannel communication are as follows:

- Do not allow the creation of user-defined guest LANs, as shown in Example 8-32.

  Example 8-32 Disabling the creation of transient guest LANs

  ```
  SET VMLAN LIMIT TRANSIENT 0
  ```

- Always use explicit inter-user communication vehicle (IUCV) authorization in the directory; do not use IUCV ALLOW or IUCV ANY.
- Because we suggested that Linux virtual servers have class G privileges, remove the DEFINE command from class G users. This way users will not be able to define CTC interfaces from within their servers. To do this, run the command shown in Example 8-33.

  Example 8-33 Revoking class G users the right to run CP DEFINE commands

  ```
  CP MODIFY COMMAND DEFINE IBMCLASS G PRIVCLASS M
  ```

  Note: If you use DirMaint, make sure it is capable of using the DEFINE command now assigned for privilege class (PRIVCLASS) M. A good idea might be to create a new class with diminished authority for the virtual guests.

- Remove the ability from class G users to set secondary consoles. Use the command shown in Example 8-34.

  Example 8-34 Revoking class G users the right to run SET SECUSER commands

  ```
  CP MODIFY COMMAND SET SECUSER IBMCLASS G PRIVCLASS M
  ```

Note: Except for the IUCV suggestion that should be implemented on each virtual user directory entry, all configurations must be updated in the system configuration file, or they will not be in operation after a subsequent IPL.

As discussed here, with a few steps, using either native CP commands or an external security management tool like RACF, you can take advantage of the built-in features the platform leverages to reinforce the security of your virtual network. In addition, you can make it a completely functional member of your existing physical network, and also make it compliant with the existing policies (firewall), unless it is a monitoring station where other security mechanisms are in place to provide security against authorized access, and where traffic controls are concerned.

8.5.6 Implementing mandatory access control

So far, several ways to provide access control to system resources have been presented, either using z/VM native functions or an external security management tool, such as RACF. Although they would be efficient and considered essential to maintain system security, they are all discretionary rules that must be carefully maintained to avoid security breaches. Imagine what might happen in a hyphotetical scenario, where hundreds of servers need to be supported, if, because of a simple mistake in the security configurations, one of the virtual guests gets access to otherwise protected resources. The company's entire IT infrastructure would be at risk and subject to unplanned outages, data leakage, and information theft.

To address this issue, the implementation of mandatory access controls, supported by an ESM, is a step in the correct direction. The establishment of system-wide security policies can make the entire virtual infrastructure more resilient and also protect the security administrators from their own mistakes when updating the discrete profiles. It would not be the security panacea, but another security layer that would, among other things, help to keep things in balance.

Mandatory access control (MAC) is a security policy that governs which subject or users can access which resources, and in what way. MAC can restrict access to an object, depending on the following:

- The security label of the subject
- The security label of the object
- The type of access that the subject requires for the task being performed

If the MAC criteria are met, z/VM proceeds with the discretionary access control when appropriate.

With MAC implemented, it would be possible to separate all the virtual resources, or objects in their security zones, and give system administrators more flexibility managing their devices. MAC guarantees that only subjects previously authorized to use a resource within a security zone have the privilege to do so, as shown in Figure 8-19.

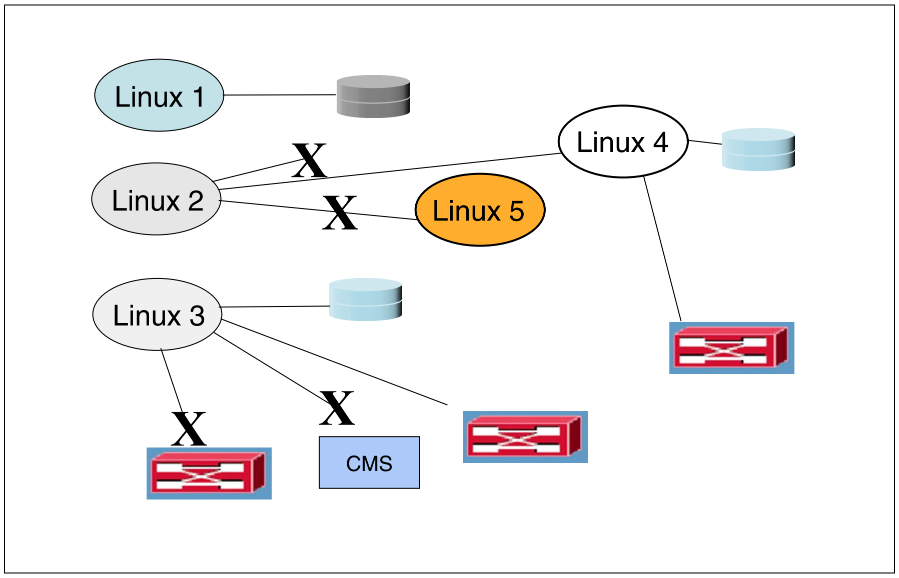


Figure 8-19 Separating resources in security zones

In this section, we briefly describe the process to define a default security level and a security label to allow only users with a specific security label to be able to couple to the virtual switches—in our examples, the VSWITCH3.

Labeled Security is part of the z/VM Common Criteria (CC) certification. RACF is the only ESM that has implemented Security Labels.

First, for our example, we create the security level and data partition shown in Example 8-35.

Example 8-35 Creating a default security level and data partition

```
RAC RDEFINE SECDATA SECLEVEL ADDMEM(DEFAULT/100)
RAC RDEFINE SECDATA CATEGORY ADDMEM(EXTRANET INTRANET)
```

In this example we create a security level and name it DEFAULT, with 100 as the level of security clearance. The number of security levels and the hierarchy you use depend on how your infrastructure is set up and what your security requirements are. For this example, we created two categories, EXTRANET and INTRANET, that will be used to represent the data separation between the resources accessible by the external network and by the internal network.

Example 8-36 shows how to group the security levels and categories.

Example 8-36 Defining a security label

```
RAC RDEFINE SECLABEL RED SECLEVEL(DEFAULT) ADDCATEGORY(EXTERNAL) UACC(NONE)
```

With the example, all resources and users that have the RED security label active must meet the access control entailed by the DEFAULT security level and EXTERNAL category. As a reminder of a best practice, also note that the universal access authority (UACC) was defined to NONE.

Resources can have one, and only one, security label associated with their profiles. Subjects or users can have multiple security labels associated, but only one active at any given time. Because of that, the scenario in Figure 8-12 on page 280 would have to be adapted to what is shown in Figure 8-20 on page 290. Two virtual switches are required, because the same VSWITCH cannot be on different security zones.

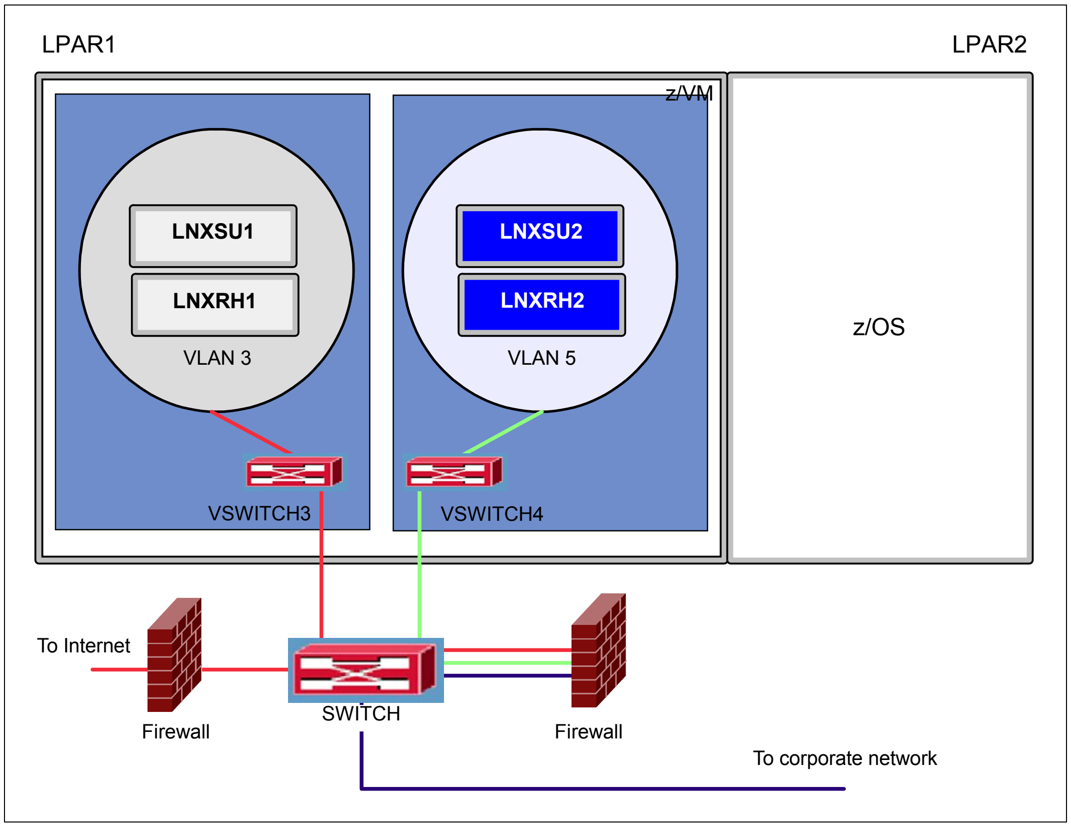


Figure 8-20 VLAN-aware infrastructure with SECLABEL active

With the security levels, categories, and security labels defined, you can update the security label-granting access to the subjects (users) your security policies mandate.

Example 8-37 Granting access permission to a user on a security label

```
RAC PERMIT RED CLASS(SECLABEL) ID(LNXSU1) ACCESS(READ)
```

We have granted read access to the virtual Linux server that we have been using for our examples, LNXSU1 to the security label RED. But, this does not mean that this specific label is associated with the user profile; if not explicitly specified during the logon, the user would not be able to use it. Therefore, assigning a default security label for the user ID, as shown in Example 8-38, is important.

Example 8-38 Assigning a security label to a user ID

```
RAC ALTUSER LNXSU1 SECLABEL(RED)
```

Next, assign a label to the resources you want to secure, which can be minidisks, vswiches, readers, and other resources. In our example, we want to secure the virtual switch VSWITCH3, so we want to assign the security label RED to the `SYSTEM.VSWITCH3` profile. Because this, as described before, is a VLAN-aware virtual switch, the same security label must be assigned to all VLAN ID-specific profiles. This task must be done manually, and it is the system administrator's responsibility to keep consistency between the several profiles for any virtual device. To update the virtual switch profile, and the two VLANs in use, 3 and 5, we used the commands shown in Example 8-39 on page 291.

Example 8-39 Assigning a security label to a virtual switch

```
RAC RALTER VMLAN SYSTEM.VSWITCH3 SECLABEL(RED)
RAC RALTER VMLAN SYSTEM.VSWITCH3.0003 SECLABEL(RED)
RAC RALTER VMLAN SYSTEM.VSWITCH3.0005 SECLABEL(RED)
```

Until now, the RACF profiles were updated, but the protection we want it to provide is not yet active. Assuming that the VMLAN class is already active, follow Example 8-40 to activate the SECLABEL and VMMAC classes.

Example 8-40 Activating the SECLABEL and VMMAC RACF classes

```
RAC SETROPTS CLASSACT(SECLABEL VMMAC)
RAC SETROPTS RACLIST(SECLABEL)
```

Then, you can activate the RACF Multi Level Security feature. Now you have to decide how the system will behave on a resource without a security label. If you want RACF to deny access to any resources where it cannot find a security label, run the command shown in Example 8-41.

Example 8-41 Setting MLACTIVE to fail on the absence of a security label

```
RAC SETROPTS MLACTIVE(FAILURES)
```

Note: Do not activate this mode if you are following this example to try security labels for the first time.

This approach provides a higher level of security, but it also requires a lot more preliminary preparation and careful definition of all your system resources' security levels, categories, and security labels.

Another configuration option is available, where the mandatory access control is evaluated only if the security label is present. If it is not present, discretionary access control will be processed where appropriate. This is the recommended method if you want to start separating your virtual infrastructure into zones, but cannot afford to have all virtual resources updated at once. To implement this operating method, use the option in Example 8-42.

Example 8-42 Activating MLACTIVE in warning mode of operation

```
RAC SETROPTS MLACTIVE(WARNING)
```

The rules of MAC on z/VM are determined by RACF, and are subject to the domination rules. Therefore, if the security level of a user is higher than the one that is applied to a resource, and all groups contained on a user are found on the object, this user dominates the resource. In other words, the security clearance of the user is higher than the object's, so it would grant access, and the discretionary access control would take place after that. The opposite is true from the object's perspective also. The possible combinations between the user's security labels and an object's security labels determine the type of access that will be granted.

In our example, the virtual switch VSWITCH3 and all the configured VLANs have the RED security label assigned to their profiles. Because we have only configured the virtual Linux server LNXSU1 with the appropriate label, if we try to log on again using another virtual server, requiring to couple to VSWITCH3, the error shown in Figure 8-21 on page 292 occurs.

```
l lnxrh1 lnxrh1
 ICH70001I LNXRH1   LAST ACCESS AT 14:52:41 ON WEDNESDAY, SEPTEMBER 23, 2009
NIC C200 is created; devices C200-C202 defined
NIC 0700 is created; devices 0700-0702 defined
RPIMGR058A Security label authorization failed for resource SYSTEM.VSWITCH3 in
t
he VMLAN class.
HCPCPL6011E You are not authorized to COUPLE to SYSTEM VSWITCH3
z/VM Version 5 Release 4.0, Service Level 0902 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0005 RDR,   NO PRT,   NO PUN
LOGON AT 14:53:05 EDT WEDNESDAY 09/23/09
z/VM V5.4.0    2009-09-09 09:47
Ready; T=0.01/0.01 14:53:05
```

Figure 8-21 Failure to couple because of the lack of a security label

Although access is permitted on the virtual switch discrete profile for LNXRH1, LNXRH1 failed in its attempt to couple to VSWITCH3, because it does not have enough security clearance. To grant access to LNXRH1, we can simply repeat Example 8-37 on page 290 and Example 8-38 on page 290 by using the new user ID you want to update.

When you have completed this, you should be able to couple to the virtual switch with no errors. See Figure 8-22.

```
NIC C200 is created; devices C200-C202 defined
NIC 0700 is created; devices 0700-0702 defined
z/VM Version 5 Release 4.0, Service Level 0902 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0005 RDR,   NO PRT,   NO PUN
LOGON AT 15:24:07 EDT WEDNESDAY 09/23/09
Ready; T=0.01/0.01 15:29:52
couple 700 system VSWITCH3
NIC 0700 is connected to VSWITCH SYSTEM VSWITCH3
Ready; T=0.01/0.01 15:30:03
```

Figure 8-22 Successful logon with a security label

Note: The appropriate configuration for a specific environment is subject to a detailed assessment of the security requirements and policies. The activation of the MAC and the security labels might affect other functions of a running z/VM installation. There, we strongly suggest a detailed study of *z/VM: Secure Configuration Guide* (for V5R4.0), SC24-6158.

8.6 Access control

In computer security, the ability of a certain subject or initiator to access a specific object or system resource is generally controlled by some sort of mechanism. Such mechanisms can either grant or deny access of a subject to an object, and also determine what kind of privilege should be granted.

Based on that, two methods to control access are possible:

- The mandatory access control (MAC)
- The discretionary access control (DAC)

We have briefly discussed both methods when securing networks. We further discuss the methods and their associated tools and procedures in this section.

Although DAC enables owners of objects to grant access to other users, MAC has the *policy* as the center of all decisions. The security policies would have to be available and respected system-wide. The compliance of your organization's security policies would not depend on the discretion of each individual user. Because DAC is not able to override what was determined by the MAC, only the function responsible for the overall security policy would be able to change what is mandatory in your security policy.

DAC has been the usual method to control access to resources on most operating systems. It is built-in, and system administrators feel comfortable using it. An example of a DAC is the well-known file system object permission on UNIX-like operating systems. Users are able to manipulate the bits that define what kind of privilege the owner, the group, and others have on the system resources they own or have authority to control.

MAC, however, has been available for several years and for several platforms. Although various approaches can be used to implement this type of access control, it is closely related to multilevel security, which describes the process to grant access to an object, based on its classification, or, in other words, based on its sensitivity and the security clearance that the initiator or the subject of the request has. That is exactly what we did with the implementation of the security labels while securing our network devices.

With the advent of SELinux and AppArmor, MAC for Linux as become more mainstream. The increasing number of security threats that organizations face today has pushed more IT organizations into the implementation of system-wide security policies as an effective way to protect their systems. A well-implemented MAC will protect the resources from internal and from unknown external exposures. The use of MAC to strenghten your organization's security policies, although initially laborious, is considered, by the authors, a best practice for access control.

Additional information about how to enable SELinux and AppArmor, as well as the main configuration files and the most common administration tasks to deal with can be found in Chapter 4, "Authentication and access control" on page 103.

8.7 Authentication

Although access control and authentication are usually closely connected, for best practices we wanted to address them in separate sections. Access control defines who can access what and how this access can be made, but authentication involves determining whether someone really is who he or she claims to be. The users attempting to access a system or a resource must first give sufficient proof of their identity.

Linux offers a very flexible interface to plug and unplug authentication mechanisms to meet the various security requirements your organization might have. The Pluggable Authentication Modules (PAM) can be used as a powerful instrument to reinforce compliance with your security policies by only authenticating users who meet specific characteristics.

PAM separates the task of authentication into four independent management groups: account management, authentication management, password management, and session management. The groups can be stacked. Several PAM modules are available to meet a wide variety of security requirements. See 4.3, “Pluggable Authentication Modules” on page 116 for more details about PAM authentication.

Password challenge has been, for a long time, the most common method to verify somebody’s identity during the authentication process. We recognize its use and understand that the authentication requirements vary from one installation to another. To avoid having passwords travelling over communication lines, the use of either RSA or DSA key pairs might be considered as an alternative. Although SSH provides an encrypted communication channel between the user and the server, the fact that the password will not go through the network is an additional factor that must be considered when strengthening overall security.

To implement the RSA key pair method, you can simply do the following:

1. Create a key pair on the server you are connecting from, as shown in Figure 8-23.

```
[root@lnxrh1 ~]# ssh-keygen -b 1024 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
2c:59:b3:aa:03:ed:16:07:1e:85:45:78:e5:5b:d3:d4 root@lnxrh1.itso.ibm.com
```

Figure 8-23 Creating an RSA key pair

2. Copy the public key you had just created. See Figure 8-24.

```
[root@lnxrh1 ~]# cat /root/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAssERl/xik1gvSGLiEvvYSWR9DCnYFcChOLnGb5ETVAzqQ/R3
LKDSvqrpO5QU2phT8Ggq2ROmkg2i5g8ygJHWsoDlcpoKxD4zuSJG7//kz6q3xzv3dGObzyoC2s1u
I9BBznThZzncKmujKcHZWMh1QJwzdKl99Ugf19LfIEc2/ok= root@lnxrh1.itso.ibm.com
```

Figure 8-24 Capturing the RSA public key

3. Copy your public key on the server you want to connect to, under the user’s `~/.ssh/authorized_keys`. In Figure 8-25, the user ID is `linuxuser`.

```
[root@lnxrh2 ~]# echo "ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAssERl/xik1gvSGLiEvvYSWR9DCnYFcChOLnGb5ETVAzqQ/R3
LKDSvqrpO5QU2phT8Ggq2ROmkg2i5g8ygJHWsoDlcpoKxD4zuSJG7//kz6q3xzv3dGObzyoC2s1u
I9BBznThZzncKmujKcHZWMh1QJwzdKl99Ugf19LfIEc2/ok= root@lnxrh1.itso.ibm.com"
>> ~linuxuser/.ssh/authorized_keys
```

Figure 8-25 Copying the public key into authorized_keys

4. Then, log on to the target system by using your RSA key pair, as shown in Figure 8-26.

```
[root@lnxrh1 .ssh]# ssh linuxuser@lnxrh2.itso.ibm.com
Enter passphrase for key '/root/.ssh/id_rsa':
```

Figure 8-26 Logging on using the key pair

Note: The passphrase that is provided does not flow through the network. Upon creation, we protected the private key with this passphrase. So the user is required to provide it to be able to open the key.

To make the process automatic, you might consider ssh-agent as an option only to provide the passphrase to open the key once, as shown in Figure 8-27.

```
[root@lnxrh1 .ssh]# eval `ssh-agent`
Agent pid 20229
[root@lnxrh1 .ssh]# ssh-add id_rsa
Enter passphrase for id_rsa:
Identity added: id_rsa (id_rsa)
[root@lnxrh1 .ssh]# ssh linuxuser@lnxrh2.itso.ibm.com
Last login: Tue Sep 29 13:47:34 2009 from 9.12.5.93
```

Figure 8-27 Using ssh-agent

We understand all the alternative methods listed here have advantages and disadvantages for the various scenarios and security requirements that the organizations have. We did not try to set a guideline or establish any type of comparison between the alternatives. Regarding authentication, there is no difference between a Linux server running on a System z server or another platform. The best practice is to take advantage of the Pluggable Authentication Modules (PAM) and build the rules that suit your needs.

8.7.1 Improving security for SSH key pair

SSH is being used as a secure alternative for Telnet and FTP services. Access to Linux guests can be facilitated with the use of SSH and the public key infrastructure (PKI). Each user has a public and private key pair that ensures the user ID.

Many users like to use this combination of SSH key pairs to access Linux guests. However, in complex environments, where you need to handle a high number of servers, the management of SSH keys is becoming a challenge for security administrators.

The process starts when an employee joins the organization. An SSH key pair is generated and the public key added into the `authorized_keys` file located in a user's home directory. Additionally, this file is synchronized to all Linux guests where the new employee requires access. On the other hand, when an employee leaves the organization, you need to remember to remove the public key from all `authorized_keys` files on all hosts. If you forget to perform this important task, you are putting your environment at risk for malicious access. You must take security very seriously and accomplish it either by individually editing the `authorized_key` file to remove any no longer used keys regularly, at least once a year or by creating an automatic process using shell scripts and so on to force key rollover.

8.8 User management

Depending on the number of servers that your organization must maintain, supporting distributed user IDs can be an effort. Even more difficult is to provide security to all of them.

8.8.1 Centralized user repository

The adoption of a centralized repository to handle and maintain user information for multiple Linux servers is considered a best practice for the maintenance and consistency of the security policies that are applied to user management.

Being able to perform user administration tasks (such as adding, deleting, and changing account information), resetting passwords from a single and centralized point, such as an LDAP server, can help keep security requirements and policies consistent throughout the infrastructure, and avoid having users' sensitive information, such as passwords, spread on hundreds of servers.

As mentioned in 8.7, "Authentication" on page 293, the traditional and most common method to authenticate users on Linux systems is through a password challenge. Unless a centralized repository is in use, the passwords must be maintained in the `/etc/shadow` file on each of the servers, multiplying the number of intrusion points to your systems. A single vulnerability on a single server can be exploited, and when the intruder is in, your entire infrastructure is at potential risk.

By using Linux native tools and functions such as PAM and the Name Service Switch (NSS), you can get the flexibility you need to easily make your distributed Linux servers authenticate on a single, central LDAP server.

Chapter 3, "Configuring and using the System z LDAP servers" on page 51 has instructions for how to set up and correctly configure the z/VM LDAP server, and with it, use an SDBM as the backend database and make use of existing RACF user definitions for Linux users. In addition, you can extend your LDAP repository with the scalability and availability you already have for your Linux servers on the System z platform.

If you have z/OS in your environment, another option is to use the z/OS LDAP server and RACF on z/OS. Additional information about how to do so, and how to configure Linux is in *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221.

Note: Although we suggest that your Linux user IDs be on a centralized repository, keeping the superuser (root) available locally is also wise. This approach enables you to log on to the servers during a planned or unplanned outage of the LDAP server, and to have each root user assigned a different password. See "Never put root in LDAP." on page 124 for a further discussion of this topic.

Implementing security is a matter of trade-offs. We explained how to turn on the virtual switch isolation, and make your virtual Linux servers fully compliant with your security policies by directing their traffic to the firewalls and routers. However, if your security policies allow you to establish a backchannel communication, like Guest LANs and HiperSockets, having a central user repository can help you reduce the intensive network traffic that the use of an LDAP server can cause on your external (outside the System z server) network segment.

8.8.2 Securing connections to the user information repository

As a default security recommendation, all the connections to the LDAP server should be through an encrypted channel, either using LDAPs on port 636, or TLS over port 389. Not all information about your central user repository will be sensitive and probably your security policies do not necessarily require this information to flow through an encrypted channel. However, we suggest to use a secure, encrypted channel, which is much easier from the operational perspective than trying to classify all the information exchanged between your Linux clients and the LDAP server.

Chapter 3, “Configuring and using the System z LDAP servers” on page 51 explains how to set up the LDAP server and the clients to establish secure connections, even to query information where anonymous binds are allowed.

We also understand that some applications might not offer support to connections through SSL or StartTLS commands. However, from a security best-practice perspective, we suggest that those applications be handled as exceptions, their usage risks assessed as the business requires, and the appropriate actions taken to restrict the scope of information they can get (as determined by your organization’s security policies).

8.9 Audit

The first steps to make your entire organization less exposed to the threats and vulnerabilities of a business environment are to identify the most adequate security requirements for your business and, based on that, create the policies that will guide the overall configuration of your IT infrastructure. The information is considered one of the main assets of any company. A constant challenge is to make it always available and at the same time always secure.

The availability of strategic information can make the difference between a successful and a failed business. And today, security must deal with the information that the organization classifies as important, and also with the legal requirements within several lines of businesses around the world.

A well-defined security policy is worthless if you do not have a way to assess whether the policies are really effective, or, in another words, whether all the parties in the organization adhere to the policy and are playing the roles they are expected to.

Section 2.6.5, “Centralized audit” on page 33 has information about how to enable auditing on several RACF classes you might be using, how to grant auditing privileges to specific user IDs, and how to access the audit records that are created by RACF.

The ability to track changes in the profiles of resources that are handled by RACF extends the same ability to the virtual environment. In a virtual environment, you still have to control the security of your physical resources, with the advantage of being able to control everything from a central location. Using the methods described in 2.6.5, “Centralized audit” on page 33, you can track changes to your minidisks, virtual switches, groups, security labels, and everything you could possibly imagine you might need to assess, if your security policies are implemented the way they were supposed to.

The resources and systems subject to auditing can vary from one organization to another, and the amount of audit records you must retain also depends on your security policies and legal obligations. Besides keeping track of all the changes that could potentially compromise your security, you must establish the appropriate actions to address them in time to avoid a major exposure.

8.10 Separation of duties

Separation of duties, as far as IT security is concerned, although laborious and in some cases expensive, is definitely a best practice. It avoids conflicts of interest and can better detect control failures that would lead to security breaches, information theft, and violations of the corporate security controls.

On z/VM, with RACF enabled, the security administrator, the system administrator, and the auditor are roles that can be easily deployed. Actually, besides the auditor, which requires the user to be a member of a specific group, the MAINT and the SYSADMIN are standard users that are created to perform the system administrator and the security administrator roles, respectively.

In Linux, defining each job role scope is more complicated because everything converges to the root user ID. However, doing so is strongly suggested by defining the policies, and having strong control of who in the organization has access to the superuser account. All other administrators, although able to run a privileged set of commands and with a high security clearance should not be able to override the mandatory access controls and should not be able to manipulate the controls that are under the jurisdiction of another job role. As an example, the auditors should not be able to change the way the resources are configured, but should be able to set up the audit logging requirements; such configurations should not be overrated by the owner of the resources or the system administrator. With the separation of duties, the functionality of your systems and the integrity of your audit logs will not be compromised.

In numerous organizations, the same person might have various roles to play. We strongly suggest that each role have its own profile and user ID. With this approach, the individual with several roles is forced to change to the environment in which each role is performed. Such implementation can help protect the system against non-intentional changes to the security environment.

A

Using phpLDAPadmin to manage the z/VM and z/OS LDAP servers

A variety of tools are available to manage LDAP databases. One very popular utility from the Open Source world is phpLDAPadmin, a web-based tool that provides a comprehensive interface to managing data in LDAP databases. It also provides a viewer for the database schema.

As a way to illustrate that the z/VM and z/OS LDAP servers can be viewed and managed with any available LDAP management utility, we configured phpLDAPadmin to manage our installed z/VM LDAP server. Once that was working, we also configured it to address the z/OS LDAP server.

Note: This material is not intended to be an exhaustive introduction to phpLDAPadmin, on the z/VM LDAP server or any other LDAP server. If you want more information about phpLDAPadmin, refer to the project website at:

`http://phpldapadmin.sourceforge.net`

Installing phpLDAPadmin

We used a SLES 11 SP1 virtual machine as the host system for phpLDAPadmin. The phpLDAPadmin software is not part of the SLES distribution, so we downloaded the software from the project website and installed it into our server's web root directory. We then had to create and edit the phpLDAPadmin configuration file to add our LDAP servers.

Note: Full instructions for configuring phpLDAPadmin are found on the project's documentation site, and also in the comments of the sample configuration file.

When the configuration was complete, we simply used a web browser to access the system. Figure A-1 shows the phpLDAPadmin main panel.

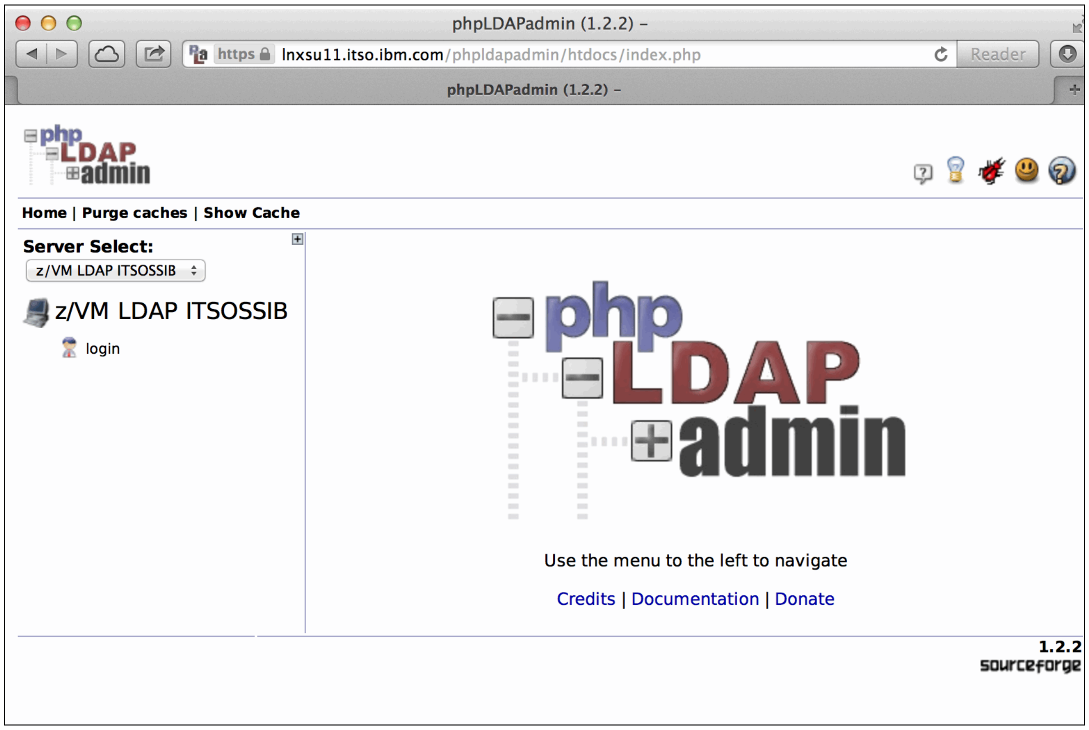

phpLDAPadmin (1.2.2) –
https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/index.php
Reader
phpLDAPadmin (1.2.2) –
Home | Purge caches | Show Cache
Server Select:
z/VM LDAP ITSOSSIB
z/VM LDAP ITSOSSIB
login
Use the menu to the left to navigate
Credits | Documentation | Donate
1.2.2
sourceforge

Figure A-1 The phpLDAPadmin welcome panel

Logging on to phpLDAPadmin

Access to phpLDAPadmin is managed by the LDAP server that you are trying to administer. The most common configuration for phpLDAPadmin is to use the credentials entered at the login panel to attempt to bind to the LDAP database; if this is successful, you have logged on to phpLDAPadmin. The level of access you have to phpLDAPadmin is then determined by the level of access you have to the LDAP database you were bound to.

Note: For our testing configuration, we used the value that was entered in the z/VM LDAP server DS CONF file as the adminDN. This is more authority than is necessary for normal operations, and we do not suggest that you do this in a real LDAP environment.

To log on to manage our z/VM LDAP server, we selected the server in the Server Select list box and pressed the login link. This displayed the login panel, shown in Figure A-2.

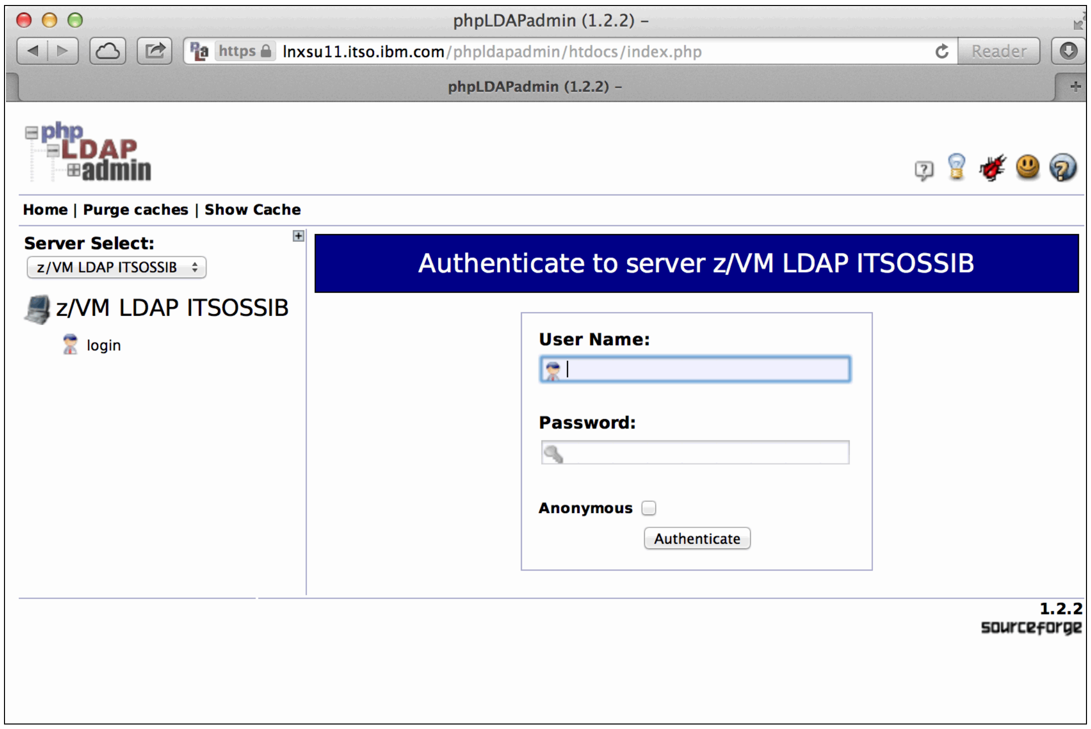


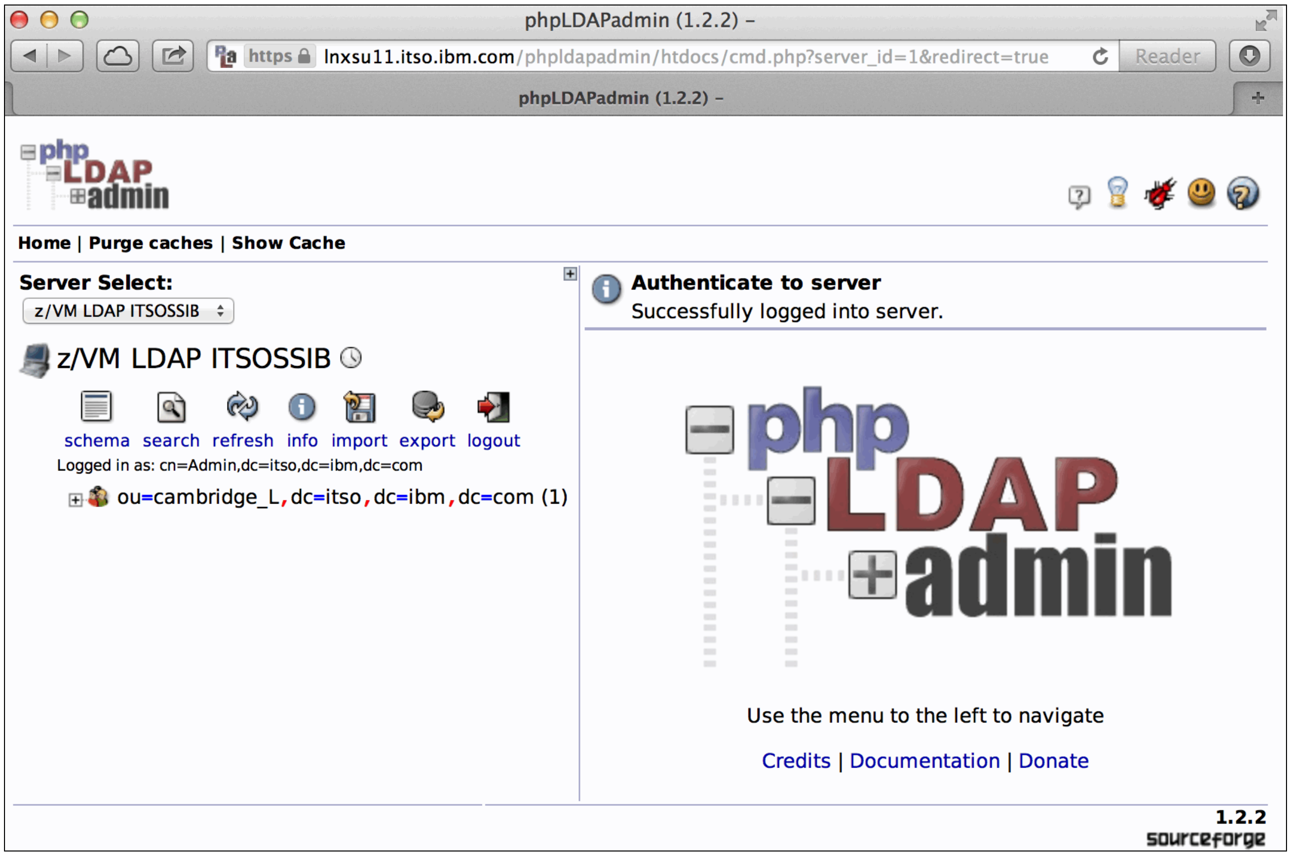
Figure A-2 The phpLDAPadmin login panel

When we first logged on, we received a message on the left side of the panel indicating that no objects were in our LDAP database. We had to create at least the base DN so that we could proceed with phpLDAPadmin functions. We used a simple LDIF file, shown in Example A-1, to define the root objects of our database.

Example A-1 LDIF to create root database objects in LDBM

```
dn: ou=cambridge_L,dc=itso,dc=ibm,dc=com
objectClass: top
objectClass: organizationalUnit
ou: cambridge_L

dn: ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com
objectClass: top
objectClass: organizationalUnit
ou: People
```

We logged in again after defining the database objects, and saw the panel shown in Figure A-3 on page 302.



Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

ou=cambridge_L,dc=itso,dc=ibm,dc=com (1)

Authenticate to server

Successfully logged into server.

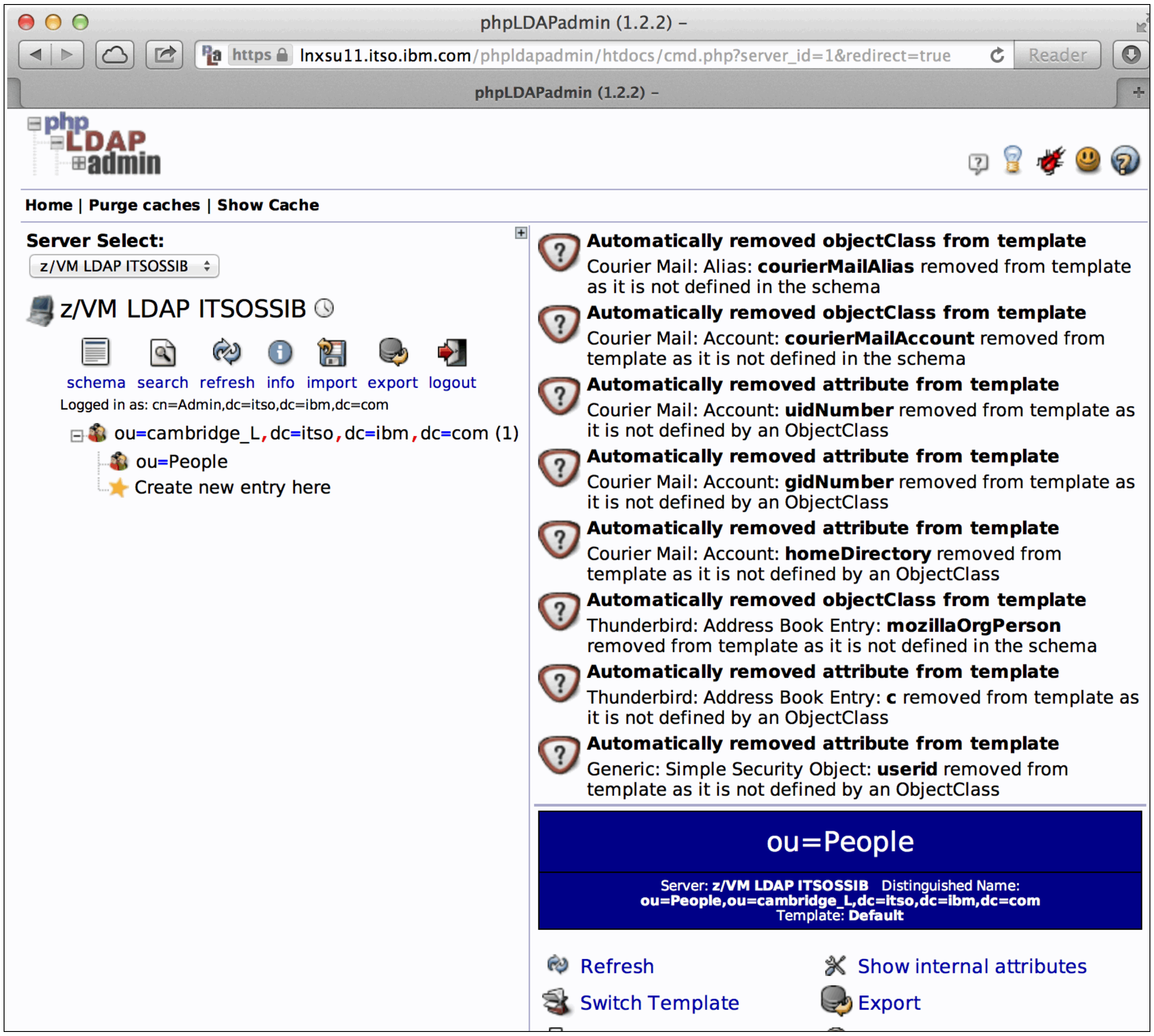
Use the menu to the left to navigate

Figure A-3 Logged in to phpLDAPadmin using our z/VM LDAP server

Having successfully logged on to phpLDAPadmin, we checked to see what functions we were able to perform.

Common schema supporting phpLDAPadmin

When we selected the `ou=People` record, we received a number of errors in the phpLDAPadmin panel, as shown in Figure A-4 on page 303.


phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?server_id=1&redirect=true

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

ou=cambridge_L,dc=itso,dc=ibm,dc=com (1)
ou=People
Create new entry here

Automatically removed objectClass from template
Courier Mail: Alias: **courierMailAlias** removed from template as it is not defined in the schema

Automatically removed objectClass from template
Courier Mail: Account: **courierMailAccount** removed from template as it is not defined in the schema

Automatically removed attribute from template
Courier Mail: Account: **uidNumber** removed from template as it is not defined by an ObjectClass

Automatically removed attribute from template
Courier Mail: Account: **gidNumber** removed from template as it is not defined by an ObjectClass

Automatically removed attribute from template
Courier Mail: Account: **homeDirectory** removed from template as it is not defined by an ObjectClass

Automatically removed objectClass from template
Thunderbird: Address Book Entry: **mozillaOrgPerson** removed from template as it is not defined in the schema

Automatically removed attribute from template
Thunderbird: Address Book Entry: **c** removed from template as it is not defined by an ObjectClass

Automatically removed attribute from template
Generic: Simple Security Object: **userid** removed from template as it is not defined by an ObjectClass

ou=People

Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Refresh Show internal attributes
Switch Template Export

Figure A-4 Template errors when selecting an object in phpLDAPadmin

The phpLDAPadmin uses a template model, using predefined definitions of various common objects found in LDAP databases (such as people, Samba servers and clients, phone book entries, email users, and so on). For these templates to work, the LDAP server must have support for the objects and attributes on which these templates are based. The messages that appeared in Figure A-4 told us that the schema in place on our z/VM LDAP server did not have many of the expected definitions. Although the schemas supplied by IBM provide a basic LDAP directory capability, extending these schemas is necessary so that they operate fully for authenticating Linux systems.

Note: These warning messages about missing schema definitions are not unique to the z/VM LDAP server. When we set up an OpenLDAP instance on Novell SLES 11 for another part of this book, we also saw similar warning messages regarding definitions that were not present in that server's schema.

As discussed in 3.4, "Extending the LDBM schema" on page 59, we extended the schema to support many of the object types required for the templates that are provided with phpLDAPadmin. Adding the appropriate schemas was done by obtaining the schema files from our Linux system, with the OpenLDAP server installed. OpenLDAP supplies the schemas that implement the object and attribute types described in Internet RFCs (such as RFC 2307 and RFC 2798), and other schemas such as those for Samba.

Including all the schemas for every feature that is supported by phpLDAPadmin is not necessary. You can remove templates for definitions you do not require, or you can turn off the warnings about invalid object classes and attribute types. When the majority of the desired objects and attributes exist in the schema, you can update the configuration of phpLDAPadmin to no longer warn about the template errors. Example A-2 shows the change in the phpLDAPadmin's `config.php` file to turn off the warning.

Example A-2 Hiding warnings about templates in phpLDAPadmin

```
/* Hide the warnings for invalid objectClasses/attributes in templates. */
$config->custom->appearance['hide_template_warning'] = true;
```

Hint: After you turn off the warnings, you might find that phpLDAPadmin does not allow you to use some of its defined templates; or, if you add a template later on, you might find that it is not working correctly. You can turn on the warnings again by changing this setting to `false` and clicking **Purge caches** on the phpLDAPadmin panel. This step enables you to see what object or attribute types your schema is missing.

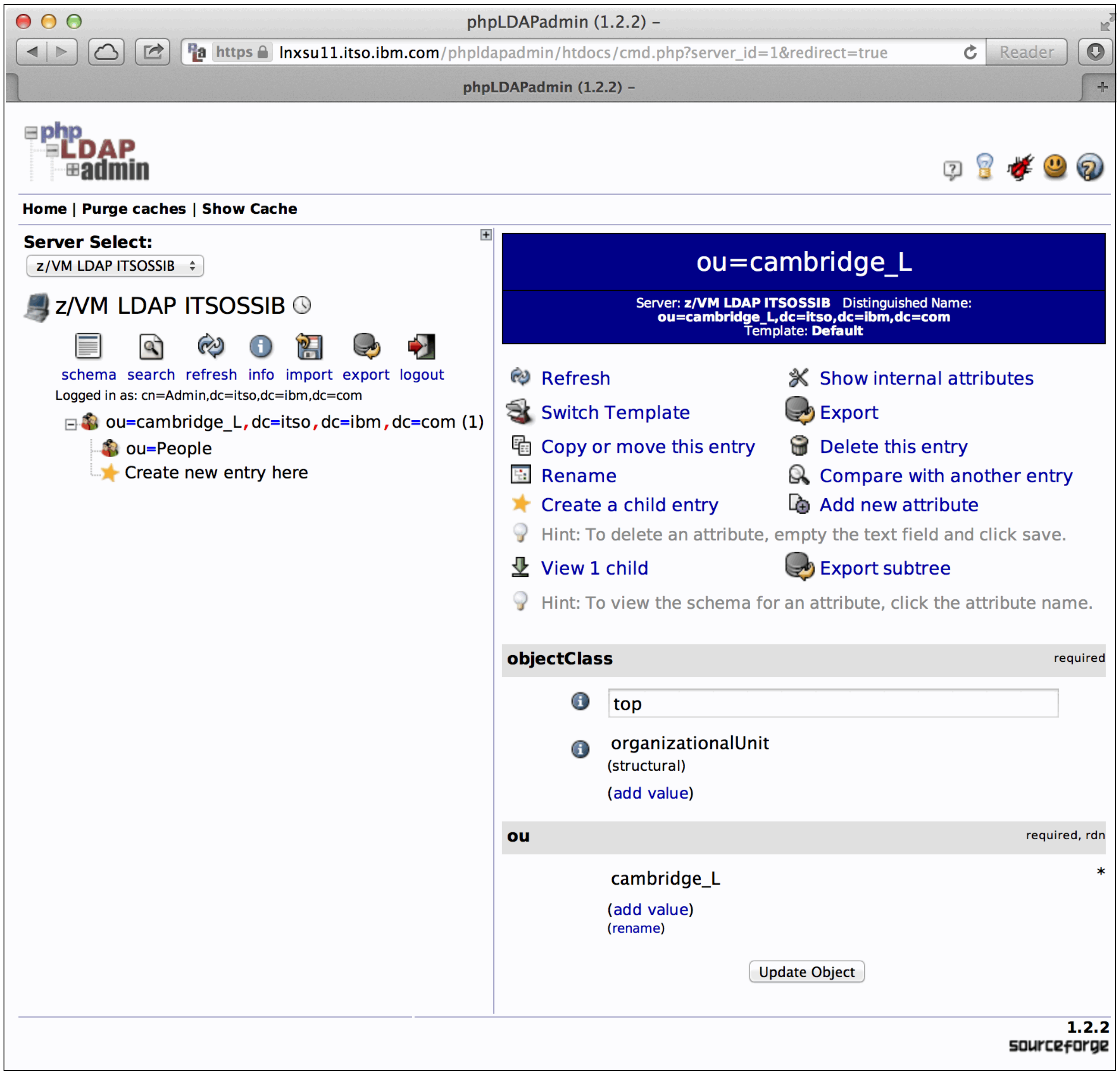
Updating LDBM using phpLDAPadmin

When our phpLDAPadmin installation seemed functional, we used it to add objects to our database. We had to add the Groups organizationalUnit object, as a prerequisite to creating user objects later.

Adding an LDAP object

We selected the top-level object for the LDBM database by clicking it in the tree view of the database on the left side of the panel. When phpLDAPadmin asked us to confirm the template to be used to edit the object, we selected **Default**.

The panel, shown in Figure A-5 on page 305, opened.



phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?server_id=1&redirect=true

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

- ou=cambridge_L,dc=itso,dc=ibm,dc=com (1)
  - ou=People
  - Create new entry here

ou=cambridge_L

Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Refresh | Show internal attributes
Switch Template | Export
Copy or move this entry | Delete this entry
Rename | Compare with another entry
Create a child entry | Add new attribute

Hint: To delete an attribute, empty the text field and click save.

View 1 child | Export subtree

Hint: To view the schema for an attribute, click the attribute name.

objectClass required

top

organizationalUnit (structural)

(add value)

ou required, rdn

cambridge_L *

(add value)
(rename)

Update Object

1.2.2

Figure A-5 Viewing LDAP object attributes with phpLDAPadmin

We clicked **Create a child entry**.

Figure A-6 shows the next stage of the process, where phpLDAPadmin prompts us for the template to use for creating the new LDAP object.

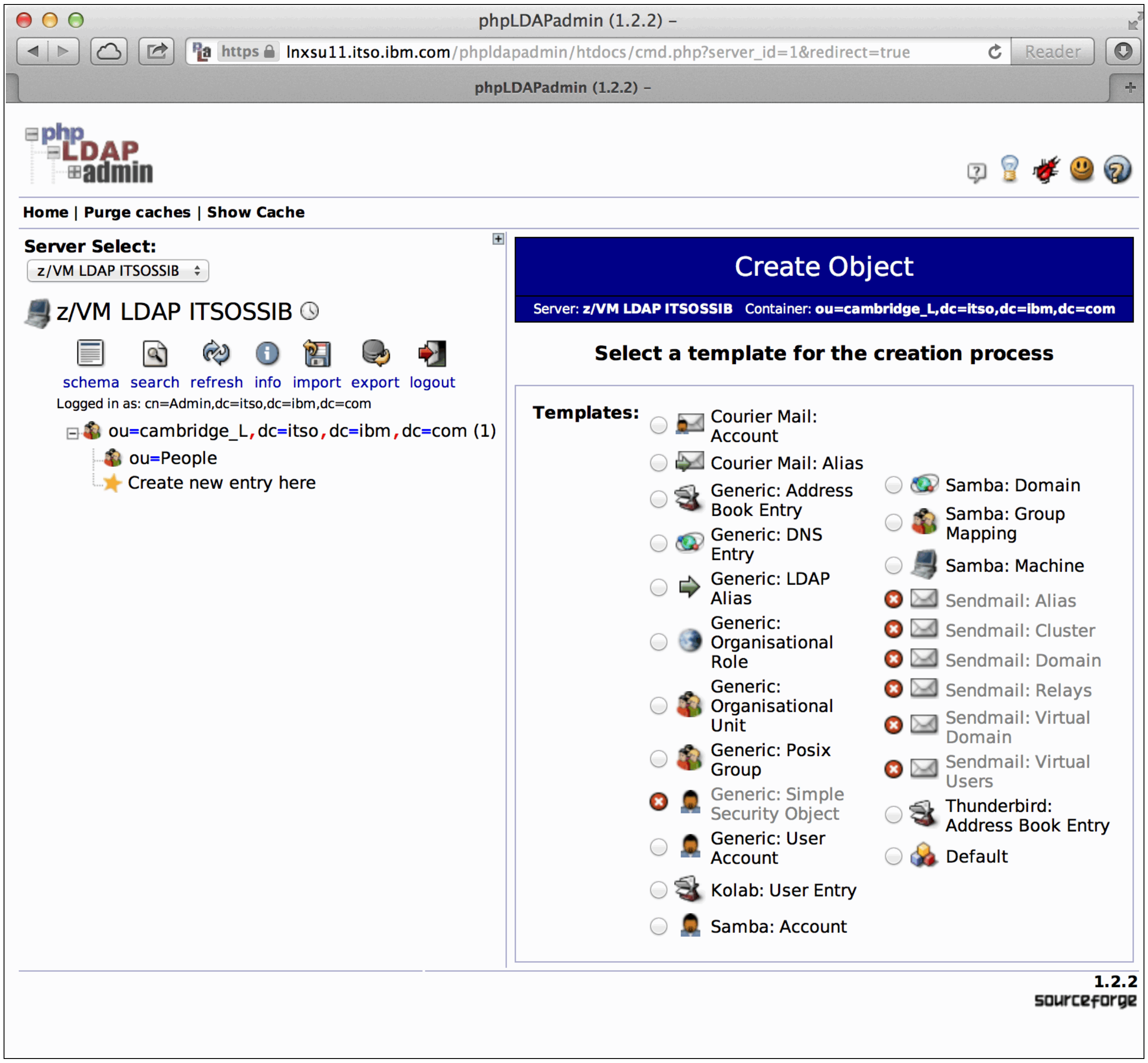
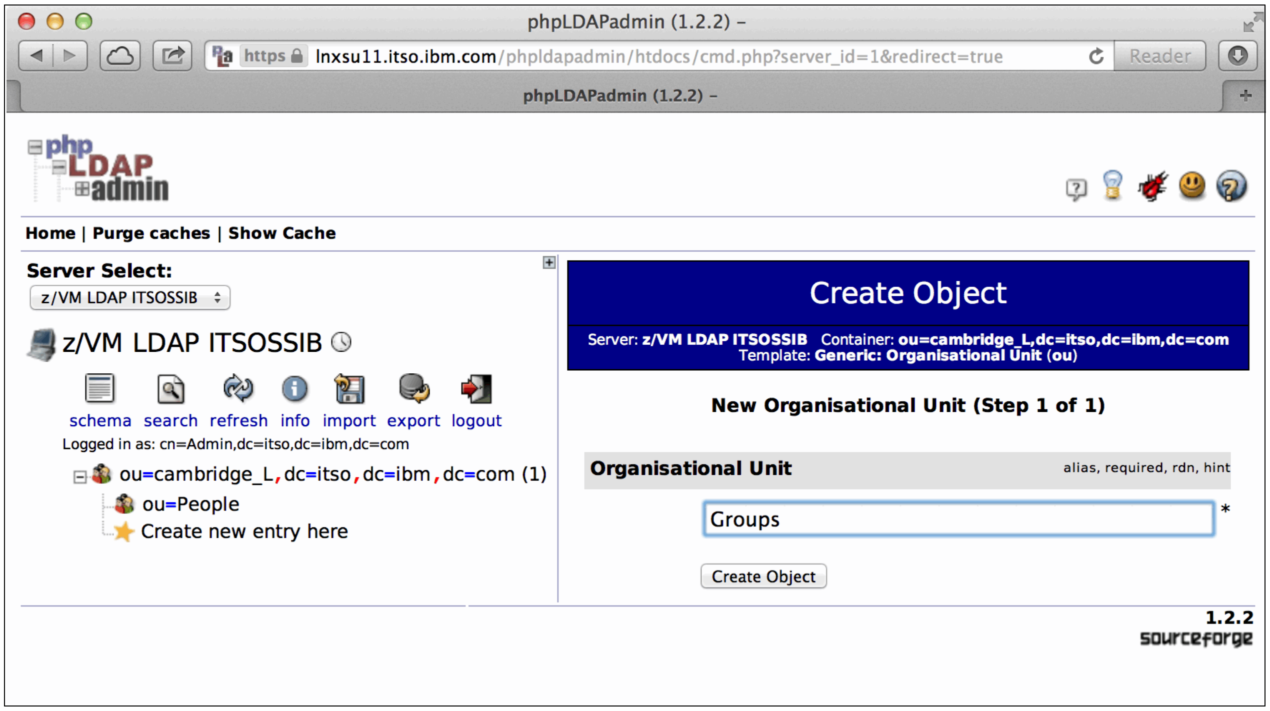

phpLDAPadmin (1.2.2) –
https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?server_id=1&redirect=true
Reader
phpLDAPadmin (1.2.2) –
php LDAP admin
Home | Purge caches | Show Cache
Server Select:
z/VM LDAP ITSOSSIB
z/VM LDAP ITSOSSIB
schema search refresh info import export logout
Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com
ou=cambridge_L,dc=itso,dc=ibm,dc=com (1)
ou=People
Create new entry here
Create Object
Server: z/VM LDAP ITSOSSIB Container: ou=cambridge_L,dc=itso,dc=ibm,dc=com
Select a template for the creation process
Templates:
Courier Mail: Account
Courier Mail: Alias
Generic: Address Book Entry
Generic: DNS Entry
Generic: LDAP Alias
Generic: Organisational Role
Generic: Organisational Unit
Generic: Posix Group
Generic: Simple Security Object
Generic: User Account
Kolab: User Entry
Samba: Account
Samba: Domain
Samba: Group Mapping
Samba: Machine
Sendmail: Alias
Sendmail: Cluster
Sendmail: Domain
Sendmail: Relays
Sendmail: Virtual Domain
Sendmail: Virtual Users
Thunderbird: Address Book Entry
Default
1.2.2
sourceforge

Figure A-6 Selecting the template to use to create an object

Note: In Figure A-6 we can see some templates that are unavailable (greyed-out, with a red-and-white “stop” symbol). Unavailable templates are caused by the LDAP server’s schema missing object or attribute definitions required by those templates, as discussed in “Common schema supporting phpLDAPadmin” on page 302.

To create the Groups organizationalUnit object, we clicked **Generic: Organisational Unit**.

The panel shown in Figure A-7 opened and we entered the name of the organizationalUnit object to be created.



phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?server_id=1&redirect=true

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

ou=cambridge_L,dc=itso,dc=ibm,dc=com (1)

ou=People

Create new entry here

Create Object

Server: z/VM LDAP ITSOSSIB Container: ou=cambridge_L,dc=itso,dc=ibm,dc=com
Template: Generic: Organisational Unit (ou)

New Organisational Unit (Step 1 of 1)

Organisational Unit — alias, required, rdn, hint

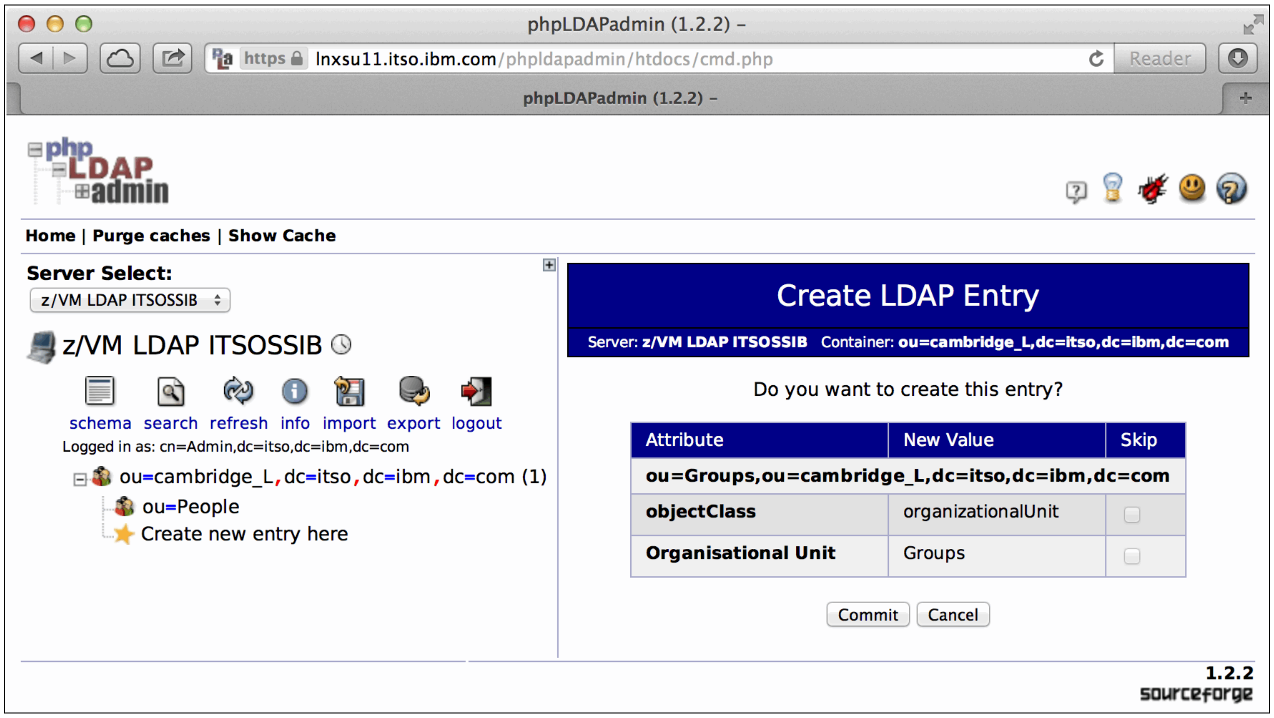
Groups *

Create Object

1.2.2

Figure A-7 Specifying the name of the organisationalUnit object

We clicked **Create Object** and then phpLDAPadmin asked us to confirm the operation we were about to perform. This is shown in Figure A-8.



phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

ou=cambridge_L,dc=itso,dc=ibm,dc=com (1)

ou=People

Create new entry here

Create LDAP Entry

Server: z/VM LDAP ITSOSSIB Container: ou=cambridge_L,dc=itso,dc=ibm,dc=com

Do you want to create this entry?

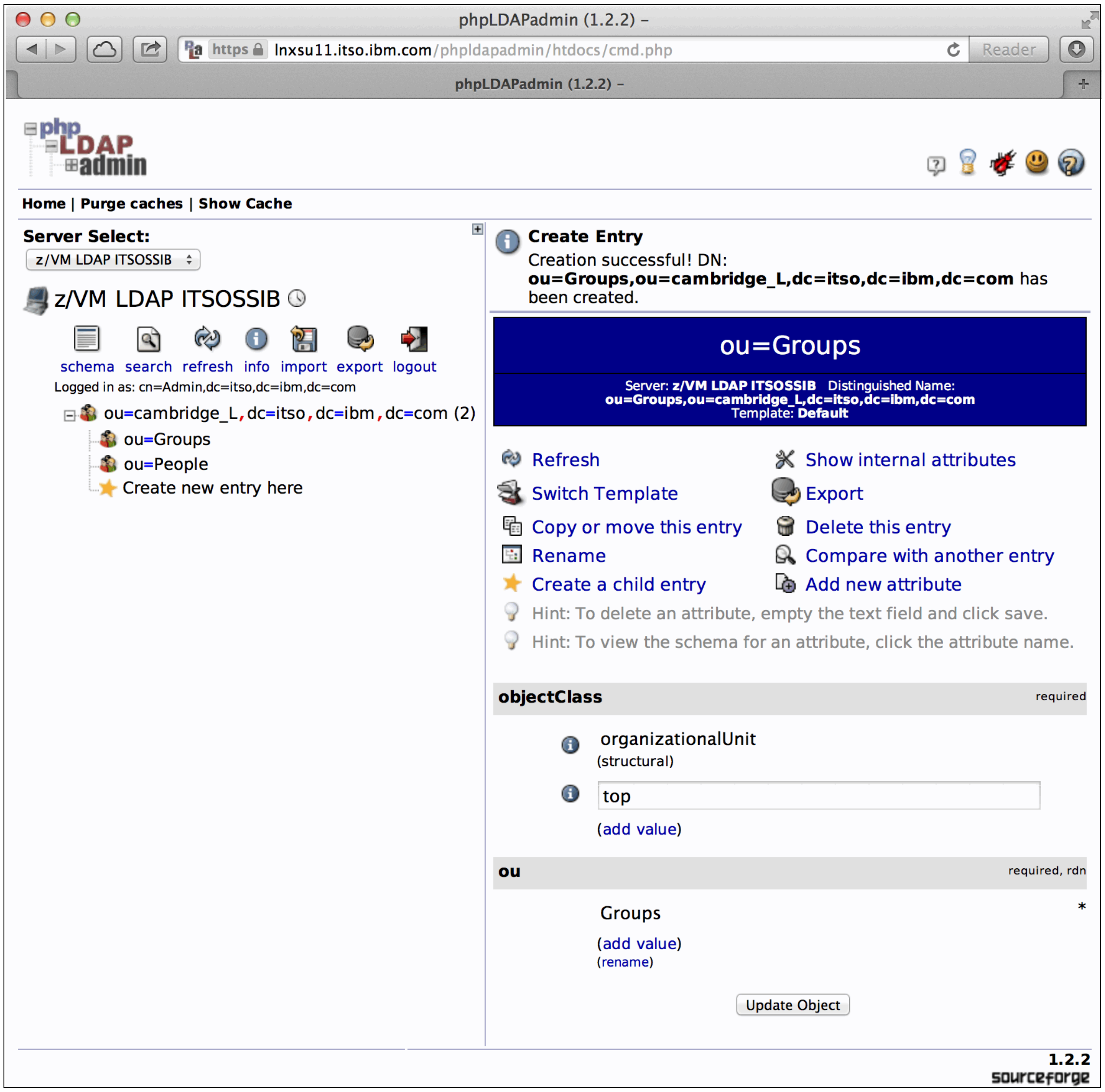
Attribute	New Value	Skip
ou=Groups,ou=cambridge_L,dc=itso,dc=ibm,dc=com		
objectClass	organizationalUnit	☐
Organisational Unit	Groups	☐

Commit Cancel

1.2.2

Figure A-8 Confirmation of object creation

We clicked **Commit**, and then phpLDAPadmin created the database object for us. The panel, shown in Figure A-9 opened, confirming the object creation and allowing us to edit it if necessary.


phpLDAPadmin (1.2.2) –
https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php
Reader
phpLDAPadmin (1.2.2) –
php LDAP admin
Home | Purge caches | Show Cache
Server Select:
z/VM LDAP ITSOSSIB
z/VM LDAP ITSOSSIB
schema search refresh info import export logout
Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com
ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
ou=Groups
ou=People
Create new entry here

Create Entry
Creation successful! DN: **ou=Groups,ou=cambridge_L,dc=itso,dc=ibm,dc=com** has been created.

ou=Groups
Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **ou=Groups,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Hint: To delete an attribute, empty the text field and click save.
Hint: To view the schema for an attribute, click the attribute name.

objectClass required
organizationalUnit (structural)
top
(add value)

ou required, rdn
Groups *
(add value)
(rename)

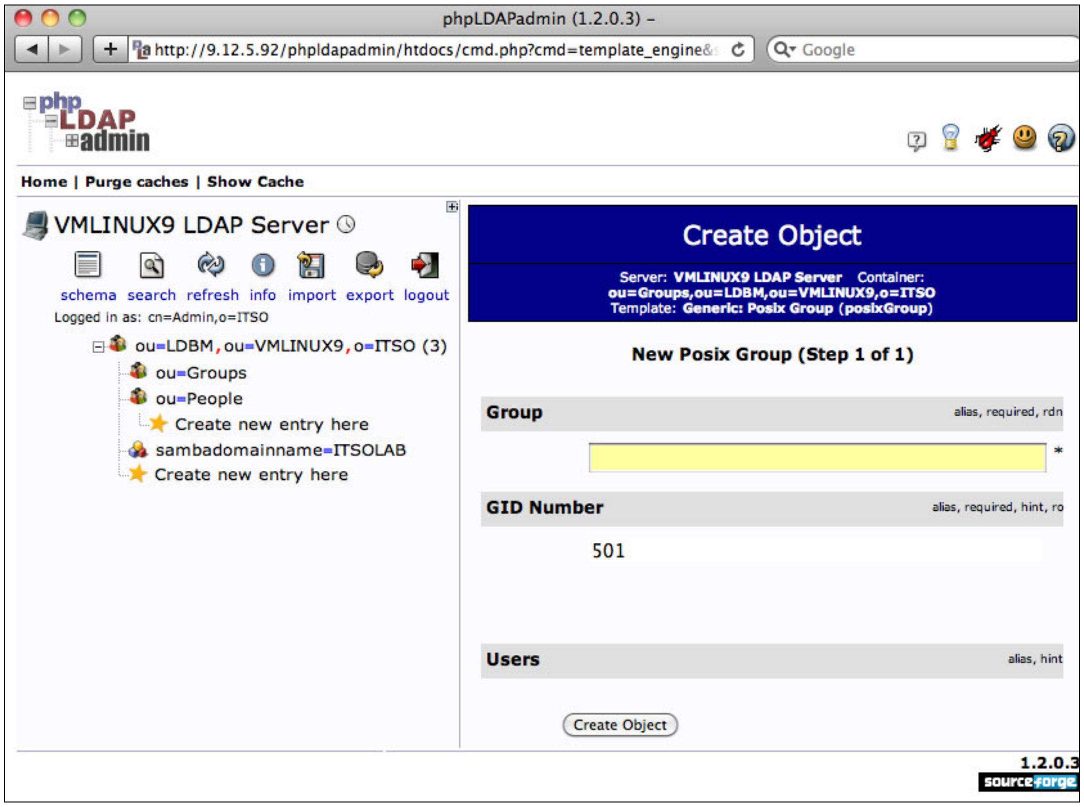
Figure A-9 Object details, with confirmation of object creation

Next, we created a group under the new organizationalUnit object that we just created. We selected **Create a child entry**, and in the panel that followed, we selected **Generic: Posix Group**. The panel, shown in Figure A-10 on page 309, opened. phpLDAPadmin had filled in the gidNumber field based on configuration defaults.

Note: In previous versions of phpLDAPadmin, there were no default values for automatically filling the uidNumber and gidNumber fields in the POSIX templates. Before automatic UID and GID generation for the POSIX templates could be used, it was necessary to create records in LDAP that phpLDAPadmin could use to "seed" the automatic generation. Now, all that is required is to modify the following line from `config.php` to provide the starting values for UID and GID generation:

```
$servers->setValue('auto_number','min',array('uidNumber'=>10001,'gidNumber'=>500));
```

Instead of 10001 and 500 as shown here, you would set the numbers to appropriate starting values for your installation.


phpLDAPadmin (1.2.0.3) –

http://9.12.5.92/phpldapadmin/htdocs/cmd.php?cmd=template_engine&

Home | Purge caches | Show Cache

VMLINUX9 LDAP Server

schema search refresh info import export logout

Logged in as: cn=Admin,o=ITSO

ou=LDBM,ou=VMLINUX9,o=ITSO (3)
- ou=Groups
- ou=People
  - Create new entry here
- sambadomainname=ITSOLAB
- Create new entry here

Create Object

Server: **VMLINUX9 LDAP Server** Container: **ou=Groups,ou=LDBM,ou=VMLINUX9,o=ITSO**
Template: **Generic: Posix Group (posixGroup)**

New Posix Group (Step 1 of 1)

Group alias, required, rdn

GID Number alias, required, hint, ro
501

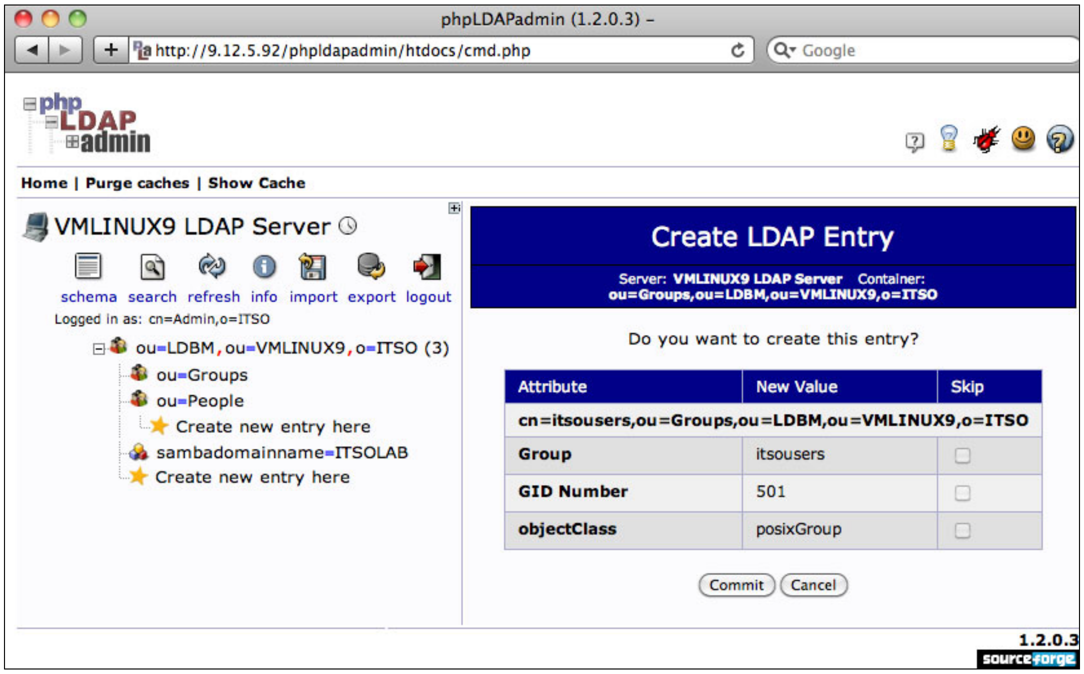
Users alias, hint

Create Object

1.2.0.3

Figure A-10 Creating a new LDAP object using the Generic: Posix Group template

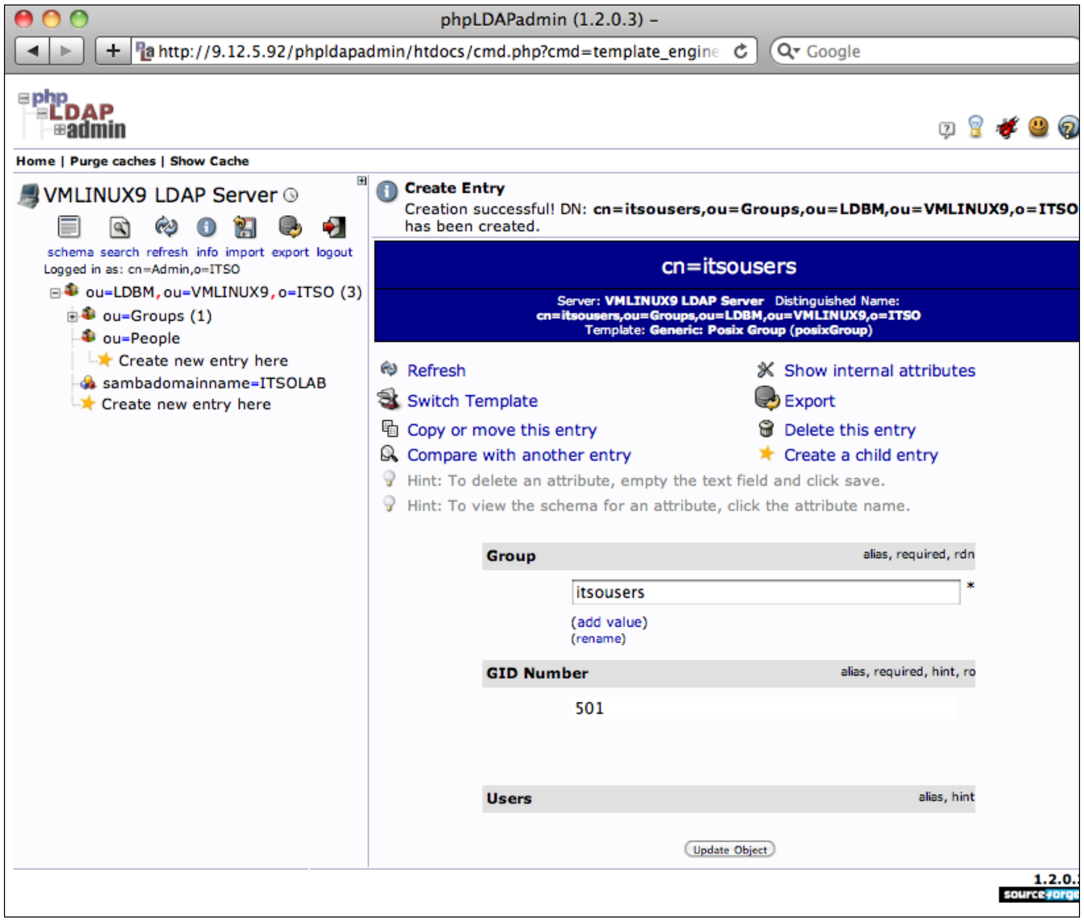
We filled in the group name as `itsousers` and selected **Create Object**. The confirmation panel shown in Figure A-11 on page 310 opened.


phpLDAPadmin (1.2.0.3) –
http://9.12.5.92/phpldapadmin/htdocs/cmd.php
Google
Home | Purge caches | Show Cache
VMLINUX9 LDAP Server
schema
search
refresh
info
import
export
logout
Logged in as: cn=Admin,o=ITSO
ou=LDBM,ou=VMLINUX9,o=ITSO (3)
ou=Groups
ou=People
Create new entry here
sambadomainname=ITSOLAB
Create new entry here
Create LDAP Entry
Server: VMLINUX9 LDAP Server Container: ou=Groups,ou=LDBM,ou=VMLINUX9,o=ITSO
Do you want to create this entry?
Commit
Cancel
1.2.0.3

Attribute	New Value	Skip
cn=itsousers,ou=Groups,ou=LDBM,ou=VMLINUX9,o=ITSO		
Group	itsousers	☐
GID Number	501	☐
objectClass	posixGroup	☐

Figure A-11 Confirming the details of the object to be added

We clicked **Commit**, and phpLDAPadmin created the new posixGroup object. This is shown in Figure A-12 on page 311.


phpLDAPadmin (1.2.0.3) –

http://9.12.5.92/phpldapadmin/htdocs/cmd.php?cmd=template_engine

Home | Purge caches | Show Cache

VMLINUX9 LDAP Server

Logged in as: cn=Admin,o=ITSO

ou=LDBM,ou=VMLINUX9,o=ITSO (3)
ou=Groups (1)
ou=People
Create new entry here
sambadomainname=ITSOLAB
Create new entry here

Create Entry
Creation successful! DN: **cn=itsousers,ou=Groups,ou=LDBM,ou=VMLINUX9,o=ITSO** has been created.

cn=itsousers

Server: **VMLINUX9 LDAP Server** Distinguished Name: **cn=itsousers,ou=Groups,ou=LDBM,ou=VMLINUX9,o=ITSO**
Template: **Generic: Posix Group (posixGroup)**

Hint: To delete an attribute, empty the text field and click save.
Hint: To view the schema for an attribute, click the attribute name.

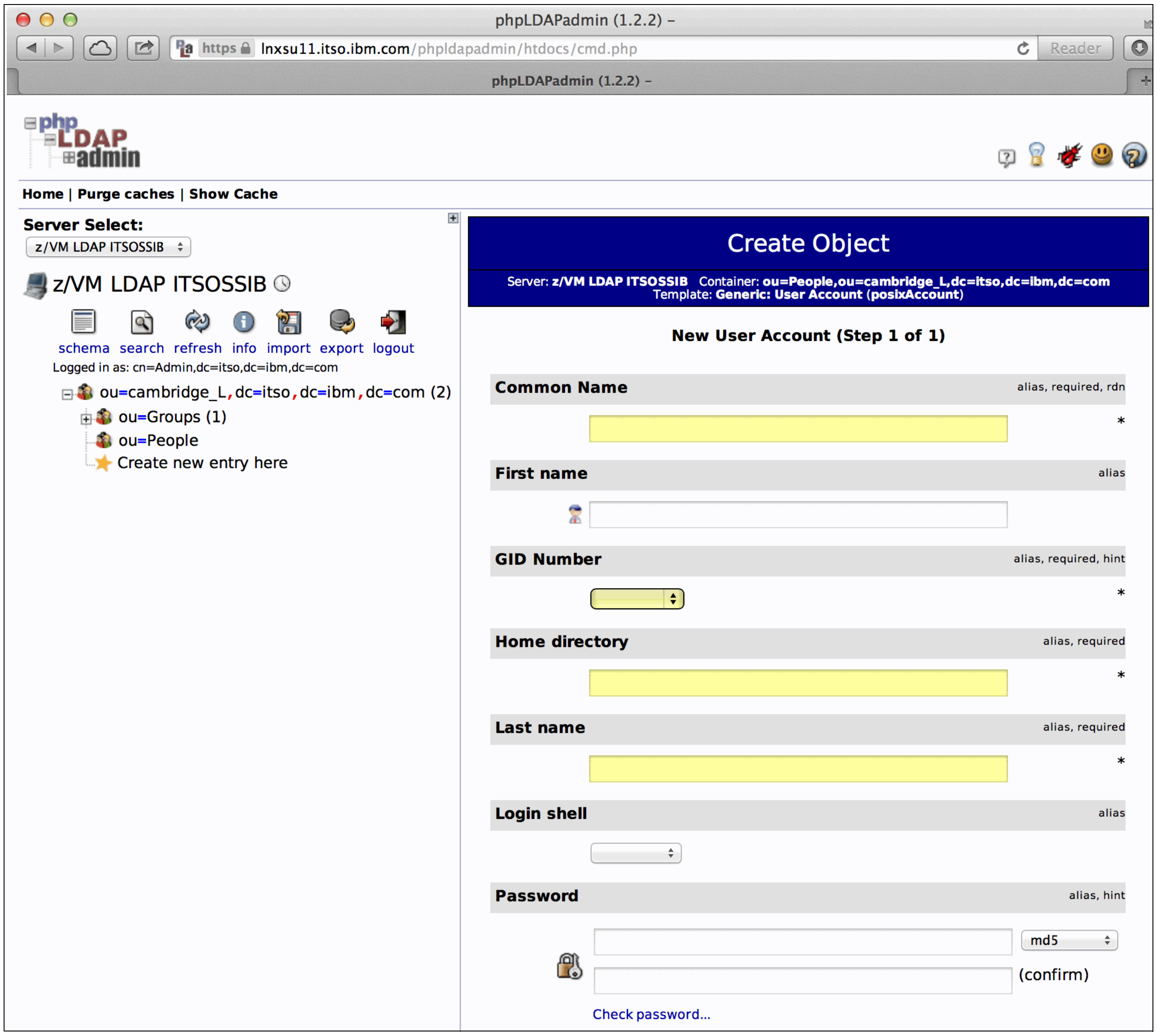
Group alias, required, rdn
itsousers *
(add value)
(rename)

GID Number alias, required, hint, ro
501

Users alias, hint

Figure A-12 Object edit panel, with confirmation of the newly created object

We then added a user object. While displaying the ou=People object, we clicked **Create a child entry**. This again opened the Create Object selection panel, where we chose **Generic: User Account**. This gave us the panel shown in Figure A-13 on page 312, where we filled in the details of the user account to be created. The figure shows that the uidNumber field was automatically generated by phpLDAPadmin. In addition, the GID Number field is a list box filled by phpLDAPadmin based on the objects found in the database of type posixGroup.


phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
- ou=Groups (1)
- ou=People
- Create new entry here

Create Object

Server: **z/VM LDAP ITSOSSIB** Container: **ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Generic: User Account (posixAccount)**

New User Account (Step 1 of 1)

Common Name	alias, required, rdn
First name	alias
GID Number	alias, required, hint
Home directory	alias, required
Last name	alias, required
Login shell	alias
Password	alias, hint

md5
(confirm)
Check password...

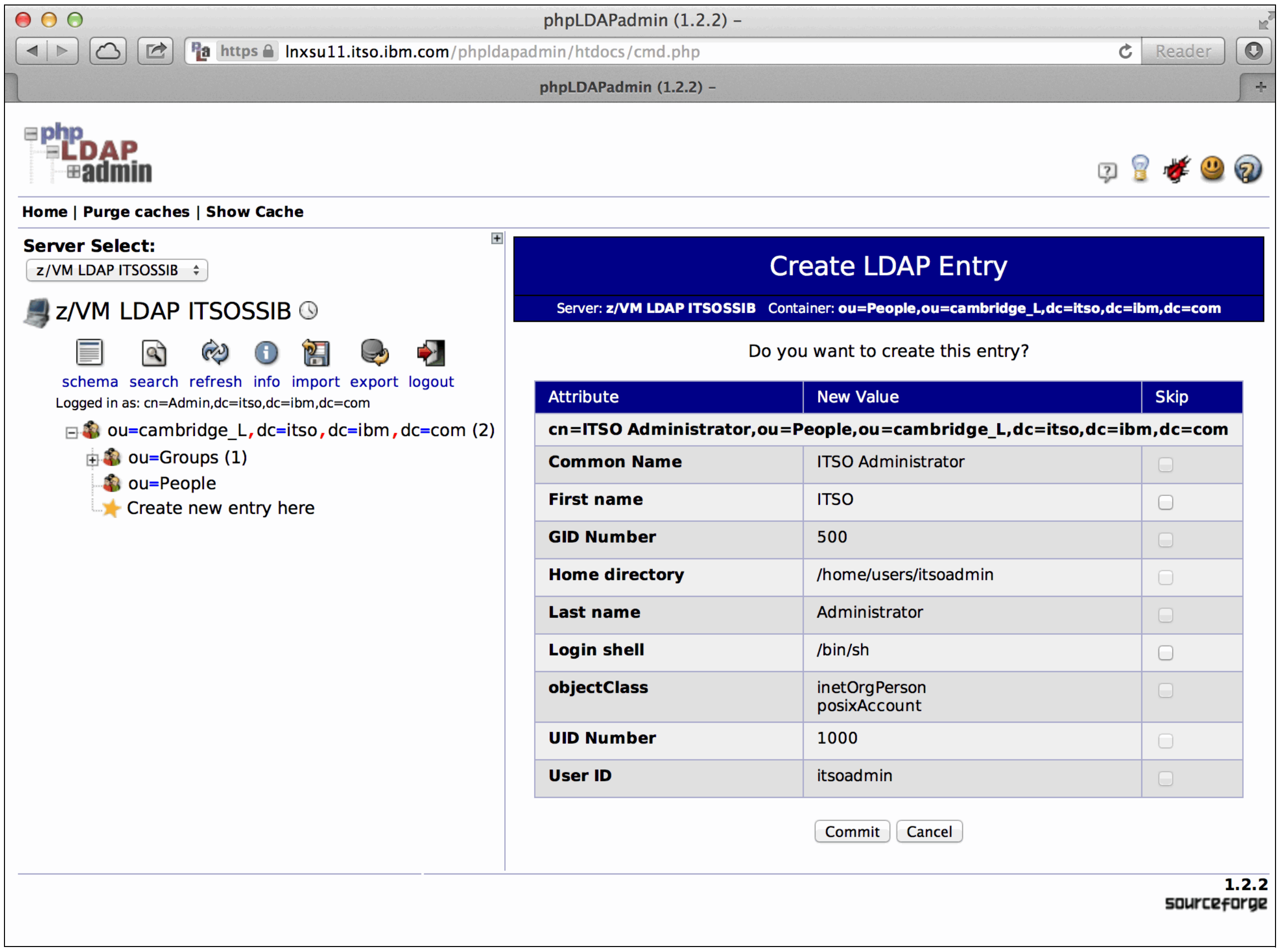
Figure A-13 Adding a new LDAP object using the Generic: Posix Account template

The Login shell field is also a list box, but this one is populated by phpLDAPadmin, based on the template file.

We filled in values for the user object to be created. The phpLDAPadmin application fills in the Common Name field as you type in the First name and Last name fields. It also derives a User ID value based on the first initial and surname entered. However, all of these values can be modified if the generated values are not suitable.

Note: Like the Login shell values, the algorithm for the auto-generation of these values is contained in the template. phpLDAPadmin allows for changes to be made to the supplied templates, or you can make your own templates by using your installation's policies.

After filling in the values, we clicked **Create Object**. The confirmation panel shown in Figure A-14 on page 313 opened.

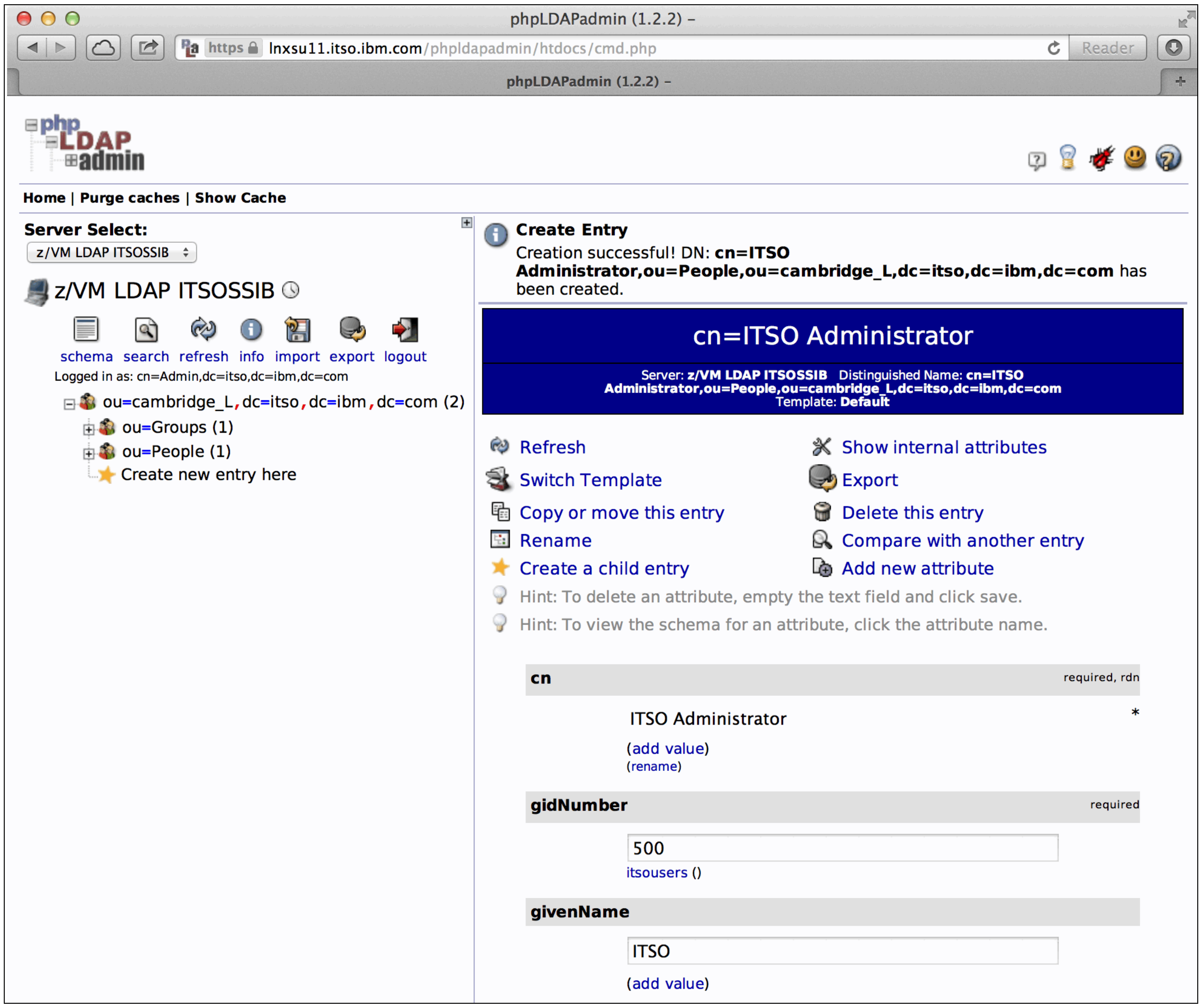

phpLDAPadmin (1.2.2) –
https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php
Reader
phpLDAPadmin (1.2.2) –
php LDAP admin
Home | Purge caches | Show Cache
Server Select:
z/VM LDAP ITSOSSIB
z/VM LDAP ITSOSSIB
schema
search
refresh
info
import
export
logout
Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com
ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
ou=Groups (1)
ou=People
Create new entry here
Create LDAP Entry
Server: z/VM LDAP ITSOSSIB Container: ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com
Commit
Cancel
1.2.2
sourceforge

Do you want to create this entry?

Attribute	New Value	Skip
cn=ITSO Administrator,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com		
Common Name	ITSO Administrator	☐
First name	ITSO	☐
GID Number	500	☐
Home directory	/home/users/itsoadmin	☐
Last name	Administrator	☐
Login shell	/bin/sh	☐
objectClass	inetOrgPerson posixAccount	☐
UID Number	1000	☐
User ID	itsoadmin	☐

Figure A-14 Confirming the attributes of the new Posix Account object

We clicked **Commit** and the update was successful, as shown in Figure A-15 on page 314.



phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php

Home | Purge caches | Show Cache

Server Select: z/VM LDAP ITSOSSIB

z/VM LDAP ITSOSSIB

schema search refresh info import export logout

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

ou=cambridge_L,dc=itso,dc=ibm,dc=com (2)
- ou=Groups (1)
- ou=People (1)
- Create new entry here

Create Entry
Creation successful! DN: **cn=ITSO Administrator,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com** has been created.

cn=ITSO Administrator

Server: **z/VM LDAP ITSOSSIB** Distinguished Name: **cn=ITSO Administrator,ou=People,ou=cambridge_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Refresh | Show internal attributes
Switch Template | Export
Copy or move this entry | Delete this entry
Rename | Compare with another entry
Create a child entry | Add new attribute

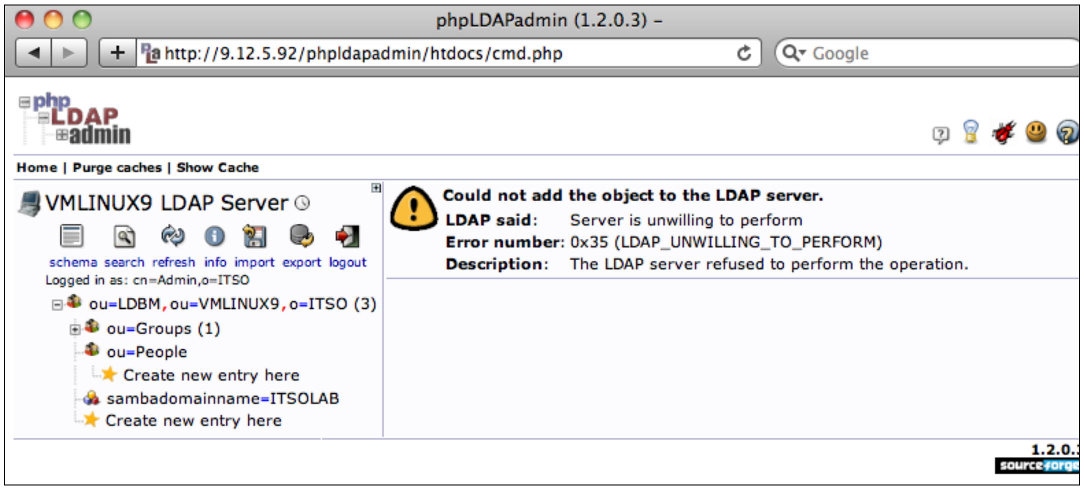
Hint: To delete an attribute, empty the text field and click save.
Hint: To view the schema for an attribute, click the attribute name.

cn	required, rdn
ITSO Administrator (add value) (rename)	*
gidNumber	required
500 itsousers ()	
givenName	
ITSO (add value)	

Figure A-15 Object edit panel, with confirmation of the newly created Posix Account object

Updating the password with native authentication

In previous versions of phpLDAPadmin, the userPassword field was flagged for update as part of the user object creation, regardless of whether it was filled in on the panel or not. This resulted in an error when phpLDAPadmin tried to create the record in LDAP, as shown in the message in Figure A-16 on page 315.


phpLDAPadmin (1.2.0.3) –
http://9.12.5.92/phpldapadmin/htdocs/cmd.php
Google
Home | Purge caches | Show Cache
VMLINUX9 LDAP Server
schema search refresh info import export logout
Logged in as: cn=Admin,o=ITSO
ou=LDBM,ou=VMLINUX9,o=ITSO (3)
ou=Groups (1)
ou=People
Create new entry here
sambadomainname=ITSOLAB
Create new entry here

Could not add the object to the LDAP server.

LDAP said: Server is unwilling to perform

Error number: 0x35 (LDAP_UNWILLING_TO_PERFORM)

Description: The LDAP server refused to perform the operation.

Figure A-16 A phpLDAPadmin error message on attempt to create the Posix Account object

To resolve this problem we had to look at the log of the LDAP server, and to see the error log we had to turn on the debug function of the LDAP server. We used the Dynamic Server Operation capability to enable debugging, and the ERROR debug level. Example A-3 shows the enabling of ERROR level debug logging on our z/VM LDAP server using Dynamic Server Operation.

Example A-3 Enabling error logging in the z/VM LDAP server

```
smsg ldapsrv debug error
Ready; T=0.01/0.01 15:05:14
LDAPSRV : 090928 15:05:14.010559 GLD1022I Debug option processed: ERROR.
 GLD1022I Debug option processed: ERROR.
```

Note: Dynamic Server Operation and Dynamic Debugging (including an explanation of the debug levels available in the z/VM LDAP server) are explained in *z/VM: TCP/IP Planning and Customization* (for V5R4.0), SC24-6125 .

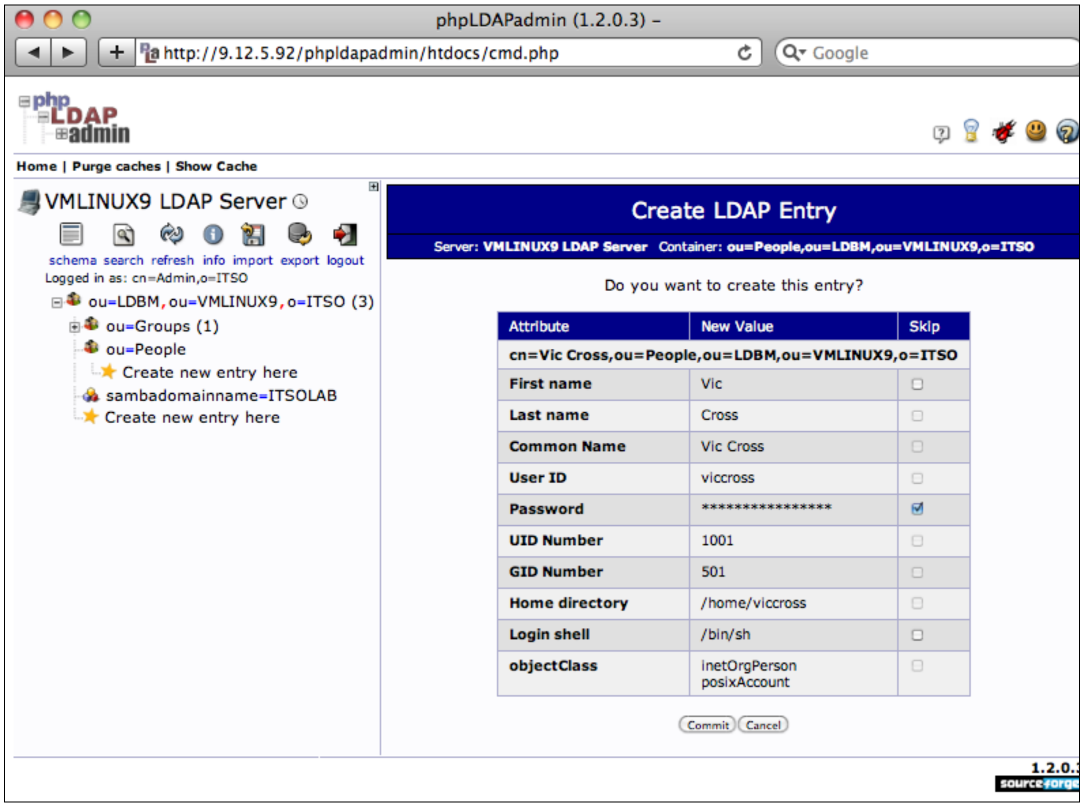
After enabling error logging, we retried the operation and saw the messages shown in Example A-4.

Example A-4 Error log messages for the attempt to create a Posix Account object

```
LDAPSRV : 090928 15:06:36.798425 (7AF4F740/001B) ERROR process_backend_request()
: Request failed OP code=8, bindDN='cn=Admin,o=ITSO', entryDN='cn=Vic Cross,ou=P
eople,ou=LDBM,ou=VMLINUX9,o=ITSO', requestElem=0x6143840, rc=53
LDAPSRV : 090928 15:06:36.798853 (7AF4F740/001B) ERROR process_backend_request()
: Request failed OP code=8 msg=R004120 The userPassword attribute cannot be adde
d because the entry uses native authentication (ldbm_add_entry)
```

We had provided a password for the user object in the **userPassword** attribute, but our LDBM backend on the LDAP server was configured for native authentication. The z/vM LDAP server does not allow the add operation on the **userPassword** attribute when native authentication is enabled.

To remove the error, we removed the userPassword attribute from the update phpLDAPadmin sent to the LDAP server. We clicked **Back** on the browser, and selected the **Skip** check box next to the userPassword attribute, as shown in Figure A-17.


phpLDAPadmin (1.2.0.3) –
http://9.12.5.92/phpldapadmin/htdocs/cmd.php
Google
Home | Purge caches | Show Cache
VMLINUX9 LDAP Server
schema search refresh info import export logout
Logged in as: cn=Admin,o=ITSO
ou=LDBM,ou=VMLINUX9,o=ITSO (3)
ou=Groups (1)
ou=People
Create new entry here
sambadomainname=ITSOLAB
Create new entry here
Create LDAP Entry
Server: VMLINUX9 LDAP Server Container: ou=People,ou=LDBM,ou=VMLINUX9,o=ITSO
Do you want to create this entry?
Commit
Cancel

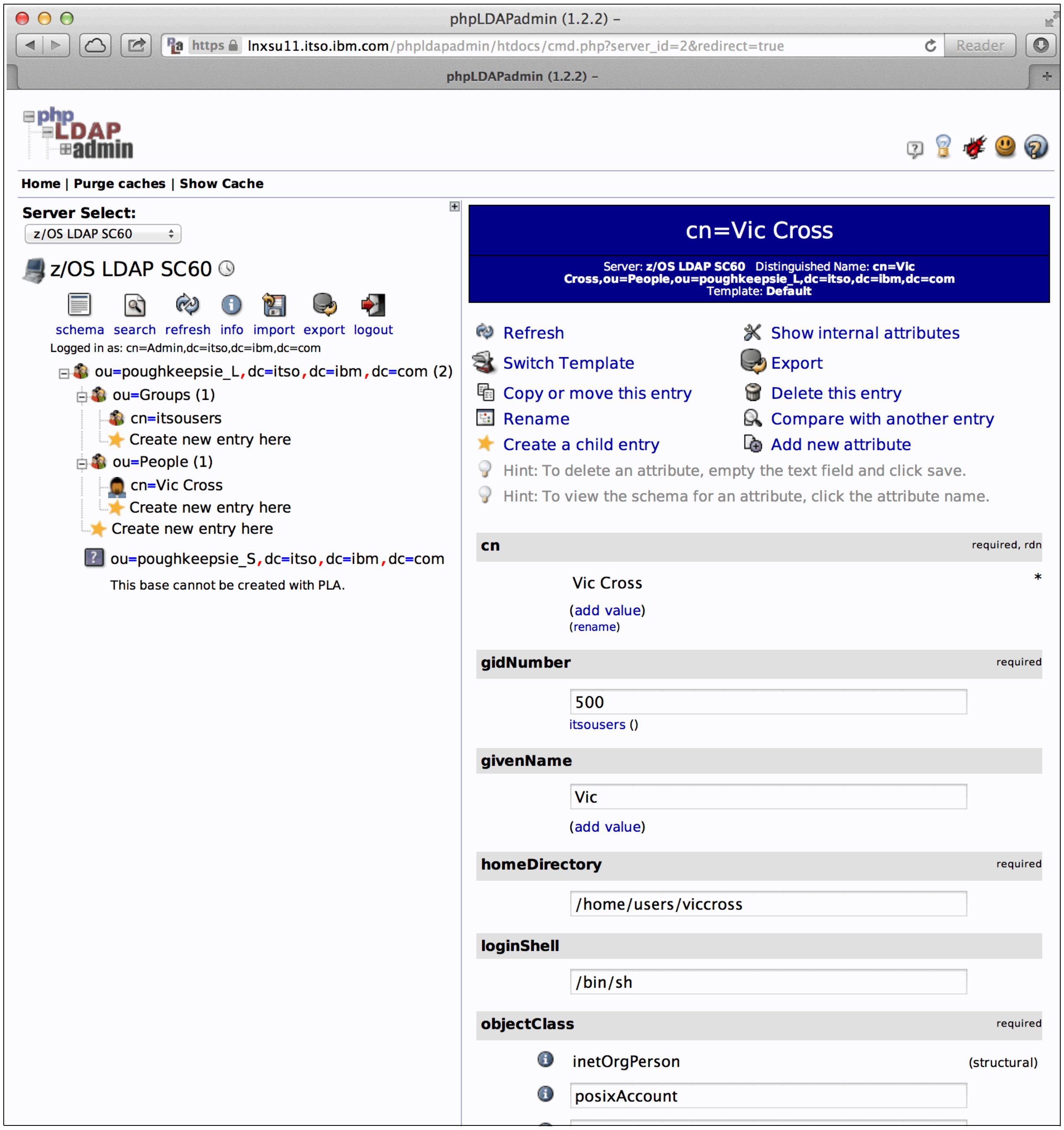
Attribute	New Value	Skip
cn=Vic Cross,ou=People,ou=LDBM,ou=VMLINUX9,o=ITSO		
First name	Vic	☐
Last name	Cross	☐
Common Name	Vic Cross	☐
User ID	viccross	☐
Password	****************	☑
UID Number	1001	☐
GID Number	501	☐
Home directory	/home/viccross	☐
Login shell	/bin/sh	☐
objectClass	inetOrgPerson posixAccount	☐

Figure A-17 Confirming the LDAP update, with the userPassword attribute skipped

The LDAP server now allowed the update to occur.

phpLDAPadmin and the z/OS LDAP server

When we set up phpLDAPadmin to manage our z/OS LDAP server, we had exactly the same experiences as we had with the z/VM LDAP server. Schema definitions, object creation, everything behaved in exactly the same way on the z/OS LDAP server. The panel showing the tree of the z/OS LDAP directory, as well as a defined user object, are shown in Figure A-18 on page 317.


phpLDAPadmin (1.2.2) –

https lnxsu11.itso.ibm.com/phpldapadmin/htdocs/cmd.php?server_id=2&redirect=true

Home | Purge caches | Show Cache

Server Select: z/OS LDAP SC60

z/OS LDAP SC60

Logged in as: cn=Admin,dc=itso,dc=ibm,dc=com

- ou=poughkeepsie_L,dc=itso,dc=ibm,dc=com (2)
  - ou=Groups (1)
    - cn=itsousers
    - Create new entry here
  - ou=People (1)
    - cn=Vic Cross
    - Create new entry here
  - Create new entry here
- ou=poughkeepsie_S,dc=itso,dc=ibm,dc=com

This base cannot be created with PLA.

cn=Vic Cross

Server: **z/OS LDAP SC60** Distinguished Name: **cn=Vic Cross,ou=People,ou=poughkeepsie_L,dc=itso,dc=ibm,dc=com**
Template: **Default**

Hint: To delete an attribute, empty the text field and click save.
Hint: To view the schema for an attribute, click the attribute name.

Attribute	Value	Notes
cn	Vic Cross * (add value) (rename)	required, rdn
gidNumber	500 itsousers ()	required
givenName	Vic (add value)	
homeDirectory	/home/users/viccross	required
loginShell	/bin/sh	
objectClass	inetOrgPerson (structural); posixAccount	required

Figure A-18 User object panel, accessing the z/OS LDAP server

Managing an SDBM backend using phpLDAPadmin

Looking at the panel in Figure A-18, you can see a second directory base listed in the section to the left of the panel: `ou=poughkeepsie_S,dc=itso,dc=ibm,dc=com`. For the z/OS LDAP server, we defined the base DN of the SDBM backend to phpLDAPadmin as well as the DN of the LDBM backend.

The schema used by the SDBM backend appears to be not recognized by phpLDAPadmin, which is why the base DN for the SDBM appears the way it does. Also, phpLDAPadmin tries to automatically create the base DN object if it believes it to be not present or invalid, but it cannot do this because the object does actually already exist.

Using phpLDAPadmin to manage the SDBM backend, in either the z/VM or z/OS LDAP servers, appears to be not possible at this time. With some development effort, it might be possible to support SDBM using this utility in the future.

B

Additional material

This book refers to additional material that can be downloaded from the Internet as described here.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

`ftp://www.redbooks.ibm.com/redbooks/SG247728`

Alternatively, you can go to the IBM Redbooks website at:

`ibm.com``/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247728.

Using the web material

The additional web material that accompanies this book includes the following files:

File name	**Description**
`ZSW03123-USEN-00_Intro.pdf`	Introduction
`ZSW03124-USEN-00_Use Cases.pdf`	Use cases

System requirements for downloading the web material

A PDF reader is required to view this material. There are no other system requirements.

How to use the web material

You can either open these files in a web browser by clicking on their links or you can download these files by saving the link locally to your computer.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Enterprise Multiplatform Auditing*, SG24-7472
- *Linux Performance and Tuning Guidelines*, REDP-4285
- *Advanced LDAP User Authentication: Limiting Access to Linux Systems Using the Host Attribute*, REDP-3863
- *Security on z/VM*, SG24-7471
- *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221
- *IBM System z10 Enterprise Class Configuration Setup*, SG24-7571

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

This publication is also relevant as further information source:

- *z/VM: CP Planning and Administration*, SC24-6178-03

Online resources

These websites are also relevant as further information sources:

- Security Management & Threat Mitigation solutions

  http://www.ibm.com/software/tivoli/solutions/security/products.html
- IBM Security Solutions: Threat Mitigration

  http://www.ibm.com/software/tivoli/solutions/threat-mitigation/
- IBM Tivoli Endpoint Manager for Security and Compliance

  http://www-01.ibm.com/software/tivoli/products/endpoint-security-compliance/
- IBM Tivoli Endpoint Manager for Patch Management

  http://www-01.ibm.com/software/tivoli/products/endpoint-patch-mgmt/
- Comprehensive security for a dynamic infrastructure

http://www.ibm.com/software/tivoli/solutions/security/

- IBM Security zSecure Manager for RACF z/VM

  http://www.ibm.com/software/tivoli/products/zsecure-mgr-zvm-racf/
- Restricting MAC address spoofing

  http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaat/liaatkvmsecnetmac.htm
- ebtables

  http://ebtables.sourceforge.net
- First experiences with hardware cryptographic support for OpenSSH with Linux for System z

  http://www.vm.ibm.com/devpages/spera/MG_OpenSSH.pdf
- *Linux on System z Performance Cryptographic Support*, available at:

  http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/perf/crypto.pdf
- *Cryptographic Hardware Use Cases for Web Servers on Linux on IBM System z*, available at:

  http://www.ibm.com/systems/resources/systems_services_platformtest_z_crypto_intro.pdf

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

Security for Linux on System z

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Security for Linux on System z

Learn about the new cryptography functions in the CEX3C

Deploy security-related technologies in Linux on System z

Understand protected key cryptography

No IT server platform is 100% secure and useful at the same time. If your server is installed in a secure vault, three floors underground in a double-locked room, not connected to any network and switched off, one would say it was reasonably secure, but it would be a stretch to call it useful.

This IBM Redbooks publication is about switching on the power to your Linux on System z server, connecting it to the data and to the network, and letting users have access to this formidable resource space in a secure, controlled, and auditable fashion to make sure the System z server and Linux are useful to your business. As the quotation illustrates, the book is also about ensuring that, before you start designing a security solution, you understand what the solution has to achieve.

The base for a secure system is tightly related to the way the architecture and virtualization has been implemented on IBM System z. Since its inception 45 years ago, the architecture has been continuously developed to meet the increasing demands for a more secure and stable platform.

This book is intended for system engineers and security administrators who want to customize a Linux on System z environment to meet strict security, audit, and control regulations.

SG24-7728-01 ISBN# 0738437549