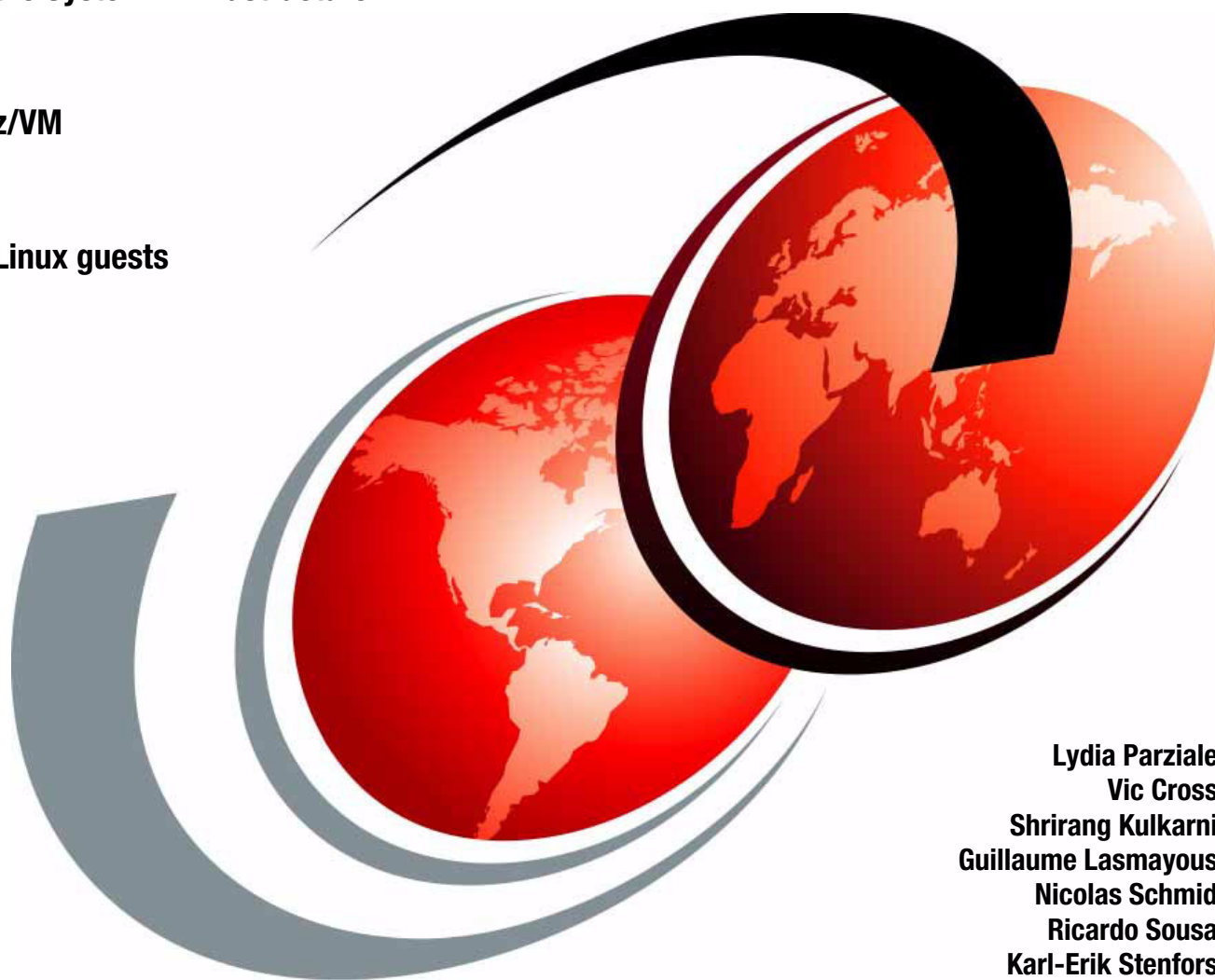


Security for Linux on System z

Securing the System z infrastructure

Securing z/VM

Securing Linux guests



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

Redbooks



International Technical Support Organization

Security for Linux on System z

January 2010

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

First Edition (January 2010)

This edition applies to z/VM version 5.4. Novell SUSE Linux Enterprise Server version 11 and Red Hat Enterprise Linux version 5.4.

© Copyright International Business Machines Corporation 2010. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team who wrote this book	x
Become a published author	xiii
Comments welcome	xiv
Chapter 1. Introduction	1
1.1 Hardware configuration	2
1.2 z/VM configuration	2
1.3 Linux distributions	2
1.4 Other software used	2
1.5 Disk storage configurations	2
Chapter 2. The z/VM security management support utilities	3
2.1 The need for security management in z/VM	4
2.1.1 Scaling-up the proof-of-concept	4
2.2 External security management	4
2.2.1 z/VM internal security	4
2.2.2 Reasons to use an ESM	5
2.2.3 Selective enablement of an ESM	6
2.3 User directory management	6
2.3.1 User management	7
2.3.2 Disk management	7
2.4 ESM and directory manager security observations	8
2.5 Securing console access to z/VM virtual machines	8
2.5.1 The role of console management in securing your environment	9
2.5.2 The z/VM LOGONBY function	9
2.5.3 Using a console management utility	10
2.6 Securing network access to z/VM	14
2.6.1 z/VM Telnet server	14
2.6.2 z/VM FTP server	21
2.7 Securing z/VM resources	24
2.7.1 Built-in security features	24
2.7.2 Securing z/VM resources with RACF	27
2.7.3 Securing TCP/IP service machines with RACF	28
2.7.4 Centralized authentication	29
2.7.5 Centralized audit	29
2.8 z/VM Directory Maintenance Facility (DirMaint)	38
2.8.1 DirMaint features	38
2.8.2 Customizing DirMaint	39
2.8.3 Using DirMaint	45
Chapter 3. Configuring and using the z/VM LDAP server	49
3.1 The z/VM LDAP server	50
3.1.1 LDAP server back ends	50
3.1.2 The relationship between z/VM LDAP server and RACF	51
3.2 Setting up the z/VM LDAP server	51

3.2.1	Activating the z/VM LDAP server	51
3.2.2	Adding schemas supplied by IBM to LDBM	54
3.3	Extending the LDBM schema	54
3.3.1	LDAP schema dependencies for Linux	54
3.3.2	Adding schemas to the z/VM LDAP server	56
3.4	Using phpLDAPadmin to manage the z/VM LDAP server	59
3.4.1	Installing phpLDAPadmin	60
3.4.2	Common schemas supporting phpLDAPadmin	62
3.4.3	Updating LDBM using phpLDAPadmin	63
3.5	LDBM and Native Authentication	81
3.5.1	LDBM record with the userPassword attribute	81
3.5.2	Creating a RACF account for an LDAP user	82
3.5.3	Identifying the RACF account corresponding to the LDAP object	82
3.6	Linux authentication using the z/VM LDAP server	83
3.6.1	Using YaST to enable LDAP on SLES 11	83
3.7	Centralizing Linux audit information with z/VM RACF	90
3.7.1	Enabling extended operations support in z/VM LDAP server	91
3.7.2	RACF configuration	91
3.7.3	Adding the @LINUX class to RACF	92
3.7.4	Linux configuration	94
3.8	Using an OpenLDAP server with the z/VM LDAP server	96
3.8.1	The OpenLDAP rewrite/remap overlay	96
3.8.2	Configuring OpenLDAP	96
Chapter 4. Authentication and access control		97
4.1	SELinux	98
4.1.1	Important files and directories for SELinux	98
4.1.2	Enabling SELinux	99
4.1.3	Disabling SELinux	103
4.1.4	Policies	104
4.1.5	RPMs required for SELinux	105
4.2	AppArmor	105
4.2.1	Important files and directories for AppArmor	105
4.2.2	Enable or disable AppArmor by using YaST	106
4.2.3	RPMs required for AppArmor	109
4.3	Pluggable Authentication Modules	109
4.3.1	Important files and libraries for PAM	110
4.3.2	Enabling PAM	111
4.3.3	PAM and LDAP	115
Chapter 5. Crypto hardware		117
5.1	Clear key	118
5.1.1	Set up of CPACF, Crypto Express2, and Crypto Express3	118
5.1.2	Accelerated Linux kernel functions	126
5.1.3	Key management	137
5.1.4	Securing communication and applications	148
5.1.5	Statistics and performance	155
5.2	Secure Key Crypto	157
5.2.1	Comparing secure key and clear key operations	157
5.2.2	Secure key functions	158
Chapter 6. Physical and infrastructure security on System z		159
6.1	Physical environment	160
6.2	Protecting the Hardware Management Console	160

6.3	Protecting the configuration	160
6.4	Building a secure multizone application environment	161
6.4.1	The multizone concept	161
6.4.2	Controlling the zones with RACF	162
6.4.3	Using HiperSockets as part of your network solution	162
6.5	IBM Proventia products	164
6.5.1	Preemptive protection	165
6.5.2	Buffer overflow exploit prevention	165
6.5.3	Firewall	165
6.6	Linux firewalls	165
6.6.1	Iptables	166
6.6.2	Linux firewall tools	169
6.7	Disk security	177
6.7.1	Traditional mainframe environments	177
6.7.2	Modern environments	177
6.8	Protecting ECKD disk	177
6.8.1	Shared-DASD configurations	178
6.8.2	LPAR configuration for Linux workloads	178
6.9	Protecting Fibre Channel Protocol (FCP) disks	179
6.9.1	Using FBA emulation in z/VM	179
6.9.2	Using N_Port ID Virtualization	183
6.10	Protecting z/VM minidisks	192
6.10.1	Minidisk access security	192
6.10.2	Overlapping minidisks	192
6.10.3	Minidisk-owning user	194
6.10.4	Shared DASD considerations	194
Chapter 7	Best practices	197
7.1	Security checklist	198
7.2	Physical security	198
7.3	Securing the logical access to z/VM	198
7.3.1	z/VM user passwords	198
7.3.2	Choosing the z/VM privilege class	199
7.3.3	z/VM network connection	199
7.4	Securing the data	200
7.4.1	Securing your minidisks	200
7.4.2	Reducing the intrusion points with shared disks	202
7.4.3	Protecting the data with encrypted file systems	204
7.5	Securing the network	205
7.5.1	Securing the virtual switches	206
7.5.2	Virtual switch using VLAN tagging	210
7.5.3	Virtual switch port isolation	214
7.5.4	Network diagnostics	216
7.5.5	Switch off backchannel communication	219
7.5.6	Implementing mandatory access control	220
7.6	Access control	225
7.7	Authentication	225
7.8	User management	228
7.8.1	Centralized user repository	228
7.8.2	Securing connections to the user information repository	229
7.9	Audit	229
7.10	Separation of duties	230

Appendix A. Using z/OS features in a Linux environment	231
Authentication using IBM Tivoli Access Manager	232
Using the LDAP server on z/OS	232
IBM Tivoli Access Manager WebSEAL	233
Appendix B. z/VSE Security and Linux on System z	235
Appendix C. Additional material	237
Locating the Web material	237
Using the Web material	237
System requirements for downloading the Web material	237
How to use the Web material	238
Abbreviations and acronyms	239
Related publications	241
IBM Redbooks	241
Other publications	241
Online resources	241
How to get Redbooks	242
Help from IBM	242
Index	243

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

DB2®	IBM®	Tivoli®
DirMaint™	Proventia®	VTAM®
DS8000®	RACF®	z/Architecture®
ECKD™	Redbooks®	z/OS®
Enterprise Storage Server®	Redpaper™	z/VM®
ESCON®	Redbooks (logo)  ®	z/VSE™
eServer™	System Storage™	z9®
FICON®	System z10™	zSeries®
FlashCopy®	System z9®	
HiperSockets™	System z®	

The following terms are trademarks of other companies:

Carta, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

AppArmor, Novell, SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Interchange, Red Hat, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

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

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

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

“Alright,” said the computer and settled into silence again. The two men fidgeted. The tension was unbearable.

“You’re really not going to like it,” observed Deep Thought.

“Tell us!”

“Alright,” said Deep Thought. “The Answer to the Great Question...”

“Yes...!”

“Of Life, the Universe and Everything...” said Deep Thought.

“Yes...!”

“Is...” said Deep Thought, and paused.

“Yes...!”

“Is...”

“Yes...!!!...?”

“Forty-two,” said Deep Thought, with infinite majesty and calm.

Extracted from: *The Hitchhiker’s Guide to the Galaxy*, by Douglas Adams, published by Pan Books, October 1979

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.

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 EAL5 Common Criteria security branding. The System z10™ Enterprise Class (EC¹) and the z10 Business Class (BC²) have also obtained this branding. To consult the official documentation for this, see:

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

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.

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 24 years in various technology areas.

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.



Figure 1 Back row from left: Nicolas Schmid, Karl-Erik Stenfors, Guillaume Lasmayous, Shrirang Kulkarni.
Front row from left: Ricardo Sousa, Vic Cross

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 *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.

Shrirang Kulkarni is an Advisory IT Specialist with IBM India Software Labs. He has over 8 years of experience in the IT industry as Lead for z/VM & Linux on System z support. Current responsibilities include z/VM System Programming and Linux for System z Administration. He currently supports environments that include half a dozen z/VM LPARs spread over couple of System z hardware, supporting 100+ Linux guests and a z/OS guest, including guest sysplex configuration.

Guillaume Lasmayous is an IT Specialist working in the IBM Products and Solutions Support Centre (PSSC) in Montpellier, France. He has 5 years of expertise in the mainframe area. He is currently working as a Technical Pre-sales Specialist, supporting customers who are implementing new workloads on the mainframe. His areas of expertise include z/VM, Linux, and a wide range of software running in this environment.

Nicolas Schmid is an IT System Specialist for System z at UBS in Zürich, Switzerland. He has 10 years of experience in the general Linux field and 2 years on System z. He holds a degree in Computer Science and Media from the Stuttgart Media University (HdM). His areas of expertise include Linux networking, performance, virtualization and security, IP networking, and z/OS. He has written extensively about Linux clear key cryptography and firewall utilities.

Ricardo Sousa is an IT Specialist for IBM Global Technology and Services in Brazil. He has 13 years of experience in the IT industry, 5 of them working with Linux on IBM system z. He is currently working as a Team Lead for the Linux team in Brazil and also as a Linux Specialist supporting more than 1500 Linux on System z servers for the IBM internal account. He holds a degree in Economics from Universidade Catolica do Salvador with an MBA in Information Management. His areas of expertise include Linux, IP networking, Linux performance and server management. He has written extensively on best practices for securing Linux on the System z platform.

Karl-Erik Stenfors is an IT Specialist in the IBM Products and Solutions Support Centre (PSSC) in Montpellier, France. Karl-Erik has spent 40 years of his career in the mainframe field, as a systems programmer, and as a consultant with IBM customers. Since 1986, he has been an employee with IBM. His areas of expertise include IBM System z hardware and operating systems. He teaches at numerous IBM user groups and IBM internal conferences. As a frequent participant in international projects, he has written several Redbooks. Karl-Erik is a member of the zChampions workgroup. His current responsibility is to execute Early Support Programmes (ESP) for IBM System z hardware and operating systems in the European and Asian geographies. Before joining the PSSC in 2000, he held two international assignments, both in the mainframe field.

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

David J. Bennin
Roy P. Costa
International Technical Support Organization, Poughkeepsie Center

Robert (Bob) Haimowitz
International Technical Support Organization, Raleigh Center

Serge E. Hallyn
IBM Texas

Peter G. Spera
Linux for System z Security Design / System Integrity Competency Center,
IBM Poughkeepsie

Sylvain Carta
Guillaume Hoareau
Josef Klitsch
Gerard Laumay
Michael Schapira
PSSC, IBM Montpellier, France

Alan Altmark
IBM Endicott

Cliff Laking
IBM UK

David Czajkowski
Frederick Gillespie
Integrated Technology Delivery, Server Systems Operations, IBM USA

Thanks to the authors of the following publications:

- ▶ Authors of *Linux Performance and Tuning Guidelines*, REDP-4285, published in April, 2008:
Eduardo Ciliendo, Takechika Kunimasa, Byron Braswell
- ▶ Authors of *Advanced LDAP User Authentication: Limiting Access to Linux Systems Using the Host Attribute*, REDP-3863, published in April, 2004:
Manfred Gnirss, Frank Kirschner
- ▶ Authors of *Security on z/VM*, SG24-7471, published November 2007 and updated December 2007:
Paola Bari, Helio Almeida, Gary Detro, David Druker, Marian Gasparovic, Manfred Gnirss, Jean Francois Jiquet.
- ▶ Authors of *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221, published June 2002 and updated May 2004:
Erich Amrehn, Ulrich Boche, Manfred Gnirss
- ▶ Authors of *Linux on System z Performance Cryptographic Support*, available at:
<http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/perf/crypto.pdf>
Thomas Weber and the IBM Lab in Böblingen
- ▶ *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
- ▶ Authors of *IBM System z10 Enterprise Class Configuration Setup*, SG24-7571:
Franck Injey, Peter Hoyle, Masaya Nakagawa, Frank Packheiser, Karan Singh

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Learn 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 e-mail 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



Introduction

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

1.1 Hardware configuration

We used a logical partition on an IBM System z10 Enterprise Class server with four processors, 4 GB of Central Storage and 2 GB of Expanded Storage.

The System z server that we used was a System z10 Enterprise Class server 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) features were configured on the system.

1.2 z/VM configuration

We used z/VM v5.4 at RSU 0902 level with the Performance Toolkit, RACF, RSCS, and DIRMAINT features installed.

The z/VM v5.4 was executing in the logical partition. For certain experiments, we used a second-level z/VM. Use this method for trying out unfamiliar security settings. Certain settings can lead to situations where you have locked yourself out from the z/VM system. The only solution to such a situation is to rebuild the system. A second-level z/VM image can save you a lot of time and effort.

1.3 Linux distributions

The book's working examples were developed by using two Linux for System z distributions. Our directory entries for the Linux virtual machines had 512 MB of memory assigned.

For our Red Hat testing and examples, we used the Red Hat Enterprise Linux (RHEL) 5.4 distribution. For testing and examples using Novell SUSE Linux Enterprise Server (SLES), we used the SLES 11 distribution. We did not use any other Linux distributions for this book.

1.4 Other software used

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

- ▶ phpLDAPAdmin Version 1.2.0.3
- ▶ IBM Operations Manager for z/VM Version 1 Release 3

1.5 Disk storage configurations

The majority of our disk storage was provided by an IBM 2105-800 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 2105-800 Enterprise Storage System, configured for OpenSystems storage. We used two FCP paths out of our System z server, using FICON® Express4 adapters. The FCP ports were attached to two ports on a BROCADE 2005-32 FCP switch, which in turn had two connections to the 2105 system.



The z/VM security management support utilities

This chapter provides information about the utilities and products that can help you manage the security of 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 certification (or 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 under 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

Many 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 de-prioritized. 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).

Note: Examples and descriptions provided in this book will 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, however:

- ▶ The Secure Database Manager (SDBM) back end 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

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

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

The document states that 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.

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 who 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 about 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. In the case of RACF, the RACF database can be placed on shared DASD, allowing multiple systems to be managed according to the same rules and settings.

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 or not. For example, the installation process for RACF enables both the VMMDISK and VMRDR classes. Additionally, if your installation does not require unit-record device protection, you will need to disable this class when the installation process has completed.

When installing RACF, the 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.

Note: When modifying this command list, be careful to ensure that the system is not made too insecure, or that essential permission commands are not removed (leading to a broken system). We recommend editing the file based on the removal of an entire unrequired class, rather than specific permissions within the class.

Refer to Chapter 7, “Best practices” on page 197 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.

Note: For more information about the user directory, refer to the z/VM manual, *z/VM Planning and Administration*, SC24-5995.

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

This section describes several features of the IBM Directory Maintenance (DirMaint™) facility, which is a priced, optional feature of z/VM. More detail about usage of DirMaint is in 2.8, “z/VM Directory Maintenance Facility (DirMaint)” on page 38.

2.3.1 User management

User definitions can be easily added and changed using a directory maintenance system. Through the use of template definitions, new users can typically be added by using a single command.

Passwords

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 the user ability to set a password:

- ▶ It avoids insecure password management practices, such as setting the password to be the same as the user name.
- ▶ It reduces *deniability*, or the opportunity for users to dispute that they are responsible for an action performed using their user ID.

When a directory management utility is not in use, passwords are easily read in the source file for the directory, and a user is unable to change his or her own password. When a directory management utility is in place, users can be given sufficient access to the directory management facility to be able to change their own password.

2.3.2 Disk management

One of the most useful features provided with directory management systems 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 operating 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 easy writing of an HTML or other interactive interface to perform user and system management tasks.

Scrubbing of released disks

When disk space is released from use by a virtual machine, ensuring that data held on that system is removed before the disks are being reused is often necessary. 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.

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

2.4 ESM and directory manager security observations

Chapter 7, “Best practices” on page 197 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, 6.10, “Protecting z/VM minidisks” on page 192 discusses an aspect of data management and how an ESM and directory management can assist in securing your data.

2.5 Securing console access to z/VM virtual machines

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

Some virtual machines use their 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.

2.5.1 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.
- ▶ 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.
- ▶ 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.

2.5.2 The z/VM LOGONBY function

z/VM provides a logon function that provides 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.

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.

2.5.3 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 2.5.1, "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 11.

Console management considerations

As introduced in 2.5, "Securing console access to z/VM virtual machines" on page 8, 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 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 2-1 shows an overview of the Operations Manager product.

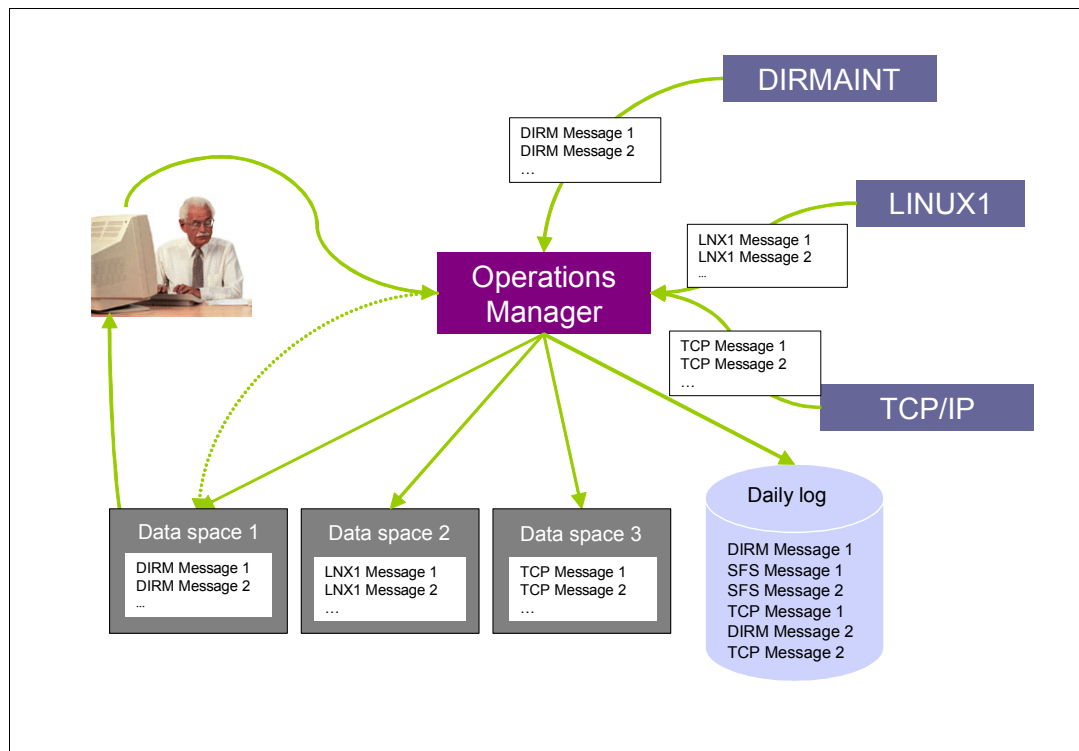


Figure 2-1 Operations Manager overview

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 2-2 shows the resulting VIEWCON session.

```

x3270-4 9.12.4.189
File Options
.rnd          fcpWwpm.csv          zosremotedebug.grc
.ssh/        ftpkey.pem
lnxsu1:~ #
* -- Operations Manager VIEWCON session from VIC          entered the following --
head fcpWwpm.csv
head fcpWwpm.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,chipidId,ssId,deviceNumber,wwpn,npiv mode,current con
A02,00,02,78,00,b800,c05076f77a000dd0,Dn,Yes,0573,5005076401e25fca
A02,00,02,78,00,b801,c05076f77a000dd4,Dn,Yes,0573,5005076401e25fca
A02,00,02,78,00,b802,c05076f77a000dd8,Dn,Yes,0573,5005076401e25fca
A02,00,02,78,00,b803,c05076f77a000ddc,Dn,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 2-2 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 2-3 on page 14 shows the VIEWSPF 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.

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			

Figure 2-3 Operations Manager VIEWSPPL list

2.6 Securing network access to z/VM

Transport Layer Security (TLS), and its ancestor 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. We define a certificate authority (CA) to issue and sign certificate requests, and implement the SSL server to be able to perform SSL or TLS encrypted connections to the system.

2.6.1 z/VM Telnet server

By default, Telnet 3270 sessions flow 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.

To protect the communication between host and clients (IBM Personal Communications, or x3270, for instance), z/VM provides an SSL-capable virtual machine, called SSLSERVE.

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 to be able to provide certificates.

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 or certification authority (CA). When running on System z, customers can choose to implement their CA to run on:

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

In this book, we implement a z/VM certification authority by using the standard z/VM tool called **gskkyman**, which can be found in the GSKADMIN virtual machine.

Note: The **gskkyman** tool is the tool used for keys and certificates management in z/VM, and that can be seen as the command-line version of the IKEYMAN graphical tool found in other IBM products.

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 certification authority 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

To create the CA certificate and key:

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 instance 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 message in Example 2-1.

Example 2-1 Successful key database creation

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 end 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

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 to **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 2-2.

Example 2-2 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

To create a certificate request for the SSL server:

1. Create a key database by following steps 1 on page 15 - 6 on page 15 in “Create the CA certificate and key” on page 15.

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 filename 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 2-3, sign the certificate request.

Example 2-3 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 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 15, 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 2-4.

Example 2-4 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 2-5 shows parts of a TCP/IP configuration file PROFILE TCPIP D relevant to SSL server configuration.

Example 2-5 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 will be able to 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 2-6 shows the SSL-related messages:

Example 2-6 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 before 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 connections
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 when requested
TCPIP : 09:39:35 DTCSTM296I Telnet server: Erase All Unprotected (EAU) data will 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 option 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 prefixed 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 SLSRV
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. 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.

2.6.2 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 2-7.

Example 2-7 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 SLSRV
===== ;
===== ;
===== ; -----
===== ; 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 17 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 2-8.

Example 2-8 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 SLSRV
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 SLSRV 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.

2.7 Securing z/VM resources

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

2.7.1 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.	-
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 2-9 on page 25 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 2-9 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 customers 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 2-10, enables LAN access restriction.

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

Example 2-10 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 2-11 on page 26.

Example 2-11 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 2-12.

Example 2-12 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 the Example 2-13.

Example 2-13 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 2-14.

Example 2-14 MDISK password in z/VM User Directory

```
01804 USER SSLSERV SSLSERV 256M 2G G
01805 INCLUDE TCPCMSU
01806 POSIXINFO UID 7 GNAME security
01807 IUCV ALLOW
01808 OPTION ACCT MAXCONN 1024 QUICKDSP SVMSTAT APPLMON
01809 SHARE RELATIVE 3000
01810 LINK 5VMTCP40 491 491 RR
01811 LINK 5VMTCP40 492 492 RR
01812 LINK TCPMAINT 591 591 RR
01813 LINK TCPMAINT 592 592 RR
01814 LINK TCPMAINT 198 198 RR
01815 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 2-15 on page 27, the password is MSSLSERV. If the password that is provided is incorrect, the link fails.

Example 2-15 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.

2.7.2 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:

- ▶ Identifies the users that connect to the system, and checks their identity.
- ▶ Gives these users access to the 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 attachments to mdisks.
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 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 7, “Best practices” on page 197.

Implementation

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 by no means 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 required exits.
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.

2.7.3 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.

2.7.4 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 z/VM LDAP server” on page 49.
- ▶ 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 96.

2.7.5 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 authentication, 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 2-16.

Example 2-16 Granting a user the AUDITOR attribute

```
rac a lu 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 2-17, RACF is configured to audit the USER, GROUP, VMDISK, VMLAN and SURROGAT classes.

Example 2-17 Querying audit functions

```
rac setropts list
ATTRIBUTES = INITSTATS NOWHEN(PROGRAM) SAUDIT CMDVIOL NOOPERAUDIT
STATISTICS = NONE
AUDIT CLASSES = USER GROUP VMDISK VMLAN SURROGAT
ACTIVE CLASSES = DATASET USER GROUP SECLABEL VMDISK 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 2-18, replacing the VMRDR class with the one required:

Example 2-18 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 VMDISK VMRDR VMLAN SURROGAT
ACTIVE CLASSES = DATASET USER GROUP SECLABEL VMDISK 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 2-19 reflects this.

Example 2-19 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
VSMERVE  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 2-20 shows how the security administrator updated a VSWITCH profile to record every coupling request to VSWITCH1, whether it is a success or failure.

Example 2-20 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 r1 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 2-21.

Example 2-21 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 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 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: 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 2-22.

Example 2-22 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 2-23.

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

Example 2-23 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 2-24.

Example 2-24 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 2-4 on page 37.

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

Figure 2-4 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 2-5 on page 38.

```

<?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 2-5 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.

2.8 z/VM Directory Maintenance Facility (DirMaint)

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.

2.8.1 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 ESM, such as RACF/VM or other vendors' external security manager (ESM).
- ▶ Online information for the DirMaint Release base feature includes multicultural support.

2.8.2 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 2-25.

Example 2-25 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 2-26 shows the EXTENT CONTROL file from our z/VM system.

Example 2-26 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) ...
[...]
```

*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 **SYSMDSG** 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 the *z/VM 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 LXDOF0 and LXDOF1) 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 2-27.

Example 2-27 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 2-28.

Example 2-28 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 2-29.

Example 2-29 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
RLDOF0 LXDOF0 START 10000 3390-9
RADOF0 LXDOF0 10001 END 3390-9
RLDOF1 LXDOF1 START 10000 3390-9
RADOF1 LXDOF1 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 2-30.

Example 2-30 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 2-31.

Example 2-31 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 $PAGE$ OA03
  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 2-32 shows a prototype file.

Example 2-32 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 LX SMB01 NOLOCK
```
2. Receive the file, using the name of the new prototype you want to create and a file type of PROTODIR:

```
RECEIVE number LX SMB 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 clon disk” on page 45.

4. Lodge the file to DirMaint.

```
DIRM FILE LX SMB PROTODIR
```

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

2.8.3 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 2-33 lists some important commands.

Example 2-33 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 2.8.2, “Customizing DirMaint” on page 39, 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 another 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 2-34.

Example 2-34 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 2-35.

Example 2-35 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 2-36. 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 passphrases does not appear in log files maintained by DirMaint.

Example 2-36 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 2-37 shows an example of using SETPW.

Example 2-37 DirMaint SETpw command

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



Configuring and using the z/VM LDAP server

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

3.1 The z/VM LDAP server

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 unmodified in a z/VM virtual machine.

The LDAP server in z/VM 5.4 was updated to the z/OS Security Server V1R10 level.

3.1.1 LDAP server back ends

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

RACF back end (SDBM)

The SDBM back end uses RACF as its backing store. All data presented by SDBM comes from the RACF database. The data in the SDBM back end 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 back end (LDBM)

The LDBM back end 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 3.3, “Extending the LDBM schema” on page 54 for information about extending the LDAP schema used by LDBM.

Logging back end (GDBM)

To provide a similar level of auditing and logging capability as SDBM, a separate back end can be configured to log changes to LDBM. This logging back end 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.

Audit back end (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 back end 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 3.7, “Centralizing Linux audit information with z/VM RACF” on page 90.

3.1.2 The relationship between z/VM LDAP server and RACF

The z/VM LDAP server is a separate component from RACF, and runs in its own service virtual machine. 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.

3.2 Setting up the z/VM LDAP server

The *z/VM: TCP/IP Planning and Customization* (for V5R4.0), SC24-6125 manual, along with the *z/VM: TCP/IP LDAP Administration Guide* (for V5R4.0), SC24-6140, 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:

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

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

3.2.1 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* (for V5R4.0), SC24-6125. 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 3-1 shows the changes to PROFILE TCPIP.

Example 3-1 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.

Example 3-2 shows the statements we provided in DS CONF.

Example 3-2 Statements from DS CONF

```
adminDn "cn=Admin,o=ITS0"
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=LDBM,ou=VMLINUX9,o=ITS0"
nativeAuthSubtree all
nativeUpdateAllowed on
useNativeAuth all

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 back ends (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 3-3.

Example 3-3 Statements from DS ENVVARS

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

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 SYSTEM DTCPARMS file.
- ▶ Added the listener for the RACF Program Call interface to DS CONF.
- ▶ Added the SDBM back end 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 3-4.

Example 3-4 DTCPARMS tag for RACF activation of the LDAP server

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

LDAP server configuration changes

Example 3-5 shows the updates made to DS CONF to enable the RACF functions of the LDAP server.

Example 3-5 DS CONF changes for RACF enablement

```
adminDn "cn=Admin,o=ITS0"
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=LDBM,ou=VMLINUX9,o=ITS0"
nativeAuthSubtree all
nativeUpdateAllowed on
useNativeAuth all

database SDBM GLDBSD31
suffix "ou=RACF,ou=VMLINUX9,o=ITS0"
```

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 (for V5R4.0)*, SC24-6125 :

```
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.

3.2.2 Adding schemas supplied by IBM to LDBM

To use the LDBM back end, 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 from *z/VM: TCP/IP Planning and Customization (for V5R4.0)*, SC24-6125 for using the sample schema files (supplied by IBM) to define the schema for our LDBM back end. The process is also described in *Security on z/VM*, SG24-7471.

3.3 Extending the LDBM schema

One of the interesting capabilities offered by the LDBM back end 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” section of *z/VM: TCP/IP LDAP Administration Guide (for V5R4.0)*, SC24-6140 .

3.3.1 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)

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 3-6 lists some of the default values from `ldap.conf` file on our Linux system.

Example 3-6 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 3-6 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 3-7 shows another portion of `ldap.conf` file, this time showing how it would be set up for the schema used on the Microsoft® Active Directory database.

Example 3-7 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 recommendation, and you should consider your mix of server and application before you decide which approach to use.

3.3.2 Adding schemas to the z/VM LDAP server

If you decide to extend the schema of the z/VM LDAP server, this section discusses how to do it.

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, `s1apd`.

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 to LDIF

The definition of the schema in the z/VM LDAP server LDBM back end 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 the Fedora Directory Server and recent versions of OpenLDAP 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: Because the DIT that is used for the schema in OpenLDAP differs from that used for the z/VM LDAP server, 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 three schemas 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.
- ▶ The schema for the OpenSSH LDAP Public Keys patch.

Schema conversion script

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 3-8.

Example 3-8 The schema-ol2ldif.pl script

```
1  #!/usr/bin/perl -w
2  #
3  # this is a quick perl script to convert OpenLDAP schema files
4  # to ldif (schema) files. it is probably not anywhere near
5  # useful, but it did allow me to convert a few of my .schema
6  # files and have FDS successfully start with them.
7  #
8  # Original by Nathan Benson <tuxtattoo@gmail.com>
9  # Modified for z/VM LDAP by Vic Cross <viccross@au1.ibm.com>
10 #
11
12 use strict;
13
14 die "usage: $0 <openldap.schema>\n" unless my $file = $ARGV[0];
15 die "$! '$file'\n" unless -e $file;
16
17 my $start;
18
19 print "dn: cn=schema\n";
20 print "changetype: modify\n";
21 print "add: attributetypes\n";
22 print "-\n";
23 print "add: objectclasses\n";
```

```

24
25 open SCHEMA, $file;
26 while (<SCHEMA>)
27 {
28     next if /^(#|$)/;
29
30
31     if (/^(objectclass|attributetype)\s/i)
32     {
33         print "\n" if ($start);
34         chomp;
35
36
37         $_      =~ s/^objectclass/objectclasses:/i;
38         $_      =~ s/^attributetype/attributetypes:/i;
39         $_      =~ s/(\t|\s)/ /;
40
41
42         $start = 1;
43         print;
44     }
45     elsif ((/^\s*\w/) && ($start))
46     {
47         chomp;
48         $_      =~ s/^\s*/ /;
49         $_      =~ s/IA5Substring/Substring/i;
50         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-0121dif.pl <schema-file> > <ldif-output-file>
```

Note: For this script to work correctly, the "add: objectclasses" line 23 should be moved to the correct location in the file.

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 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 re-ran the converted script.

The `inetOrgPerson` schema has dependencies that were mostly satisfied by the NIS schema. If you need to use the `inetOrgPerson` object, first add the NIS schema.

The LPK schema is extremely simple, consisting of only one attribute type and one object class. This schema converted and applied with no difficulty. Example 3-9 shows how we created the LDIF file for the LPK schema.

Example 3-9 Converting LPK schema file to LDIF format, and the result

```
lnxsul:~ # schema-0121dif.pl lpk.schema > lpk.ldif
lnxsul:~ # cat lpk.ldif
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.24552.500.1.1.1.13 NAME 'sshPublicKey' DESC
'MANDATORY: OpenSSH Public Key' EQUALITY octetStringMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.40 )
-
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.24552.500.1.1.2.0 NAME 'ldapPublicKey' SUP top
AUXILIARY DESC 'MANDATORY: OpenSSH LPK objectclass' MAY ( sshPublicKey $ uid ) )
lnxsul:~ #
```

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 3-10 shows how we extended the z/VM LDAP server schema for the OpenSSH LDAP Public Keys schema.

Example 3-10 The `ldapmodify` command session

```
lnxsul:~ # ldapmodify -v -a -c -h 9.12.4.189 -Z -x -D "cn=Admin,o=ITS0" -w secret
< lpk.ldif
ldap_initialize( ldap://9.12.4.189 )
add attributetypes:
  ( 1.3.6.1.4.1.24552.500.1.1.1.13 NAME 'sshPublicKey' DESC 'MANDATORY: OpenSSH
Public Key' EQUALITY octetStringMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
add objectclasses:
  ( 1.3.6.1.4.1.24552.500.1.1.2.0 NAME 'ldapPublicKey' SUP top AUXILIARY DESC
'MANDATORY: OpenSSH LPK objectclass' MAY ( sshPublicKey $ uid ) )
modifying entry "cn=schema"
modify complete

lnxsul:~ #
```

3.4 Using `phpLDAPAdmin` to manage the z/VM LDAP server

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 LDAP server can be viewed and managed by using any available LDAP management utility, we configured phpLDAPAdmin to manage our installed z/VM 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 using phpLDAPAdmin, refer to the project Web site at:
<http://phpldapadmin.sourceforge.net>

3.4.1 Installing phpLDAPAdmin

We used a Novell SLES 11 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 Web site 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 server.

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 3-1 shows the phpLDAPAdmin main panel.

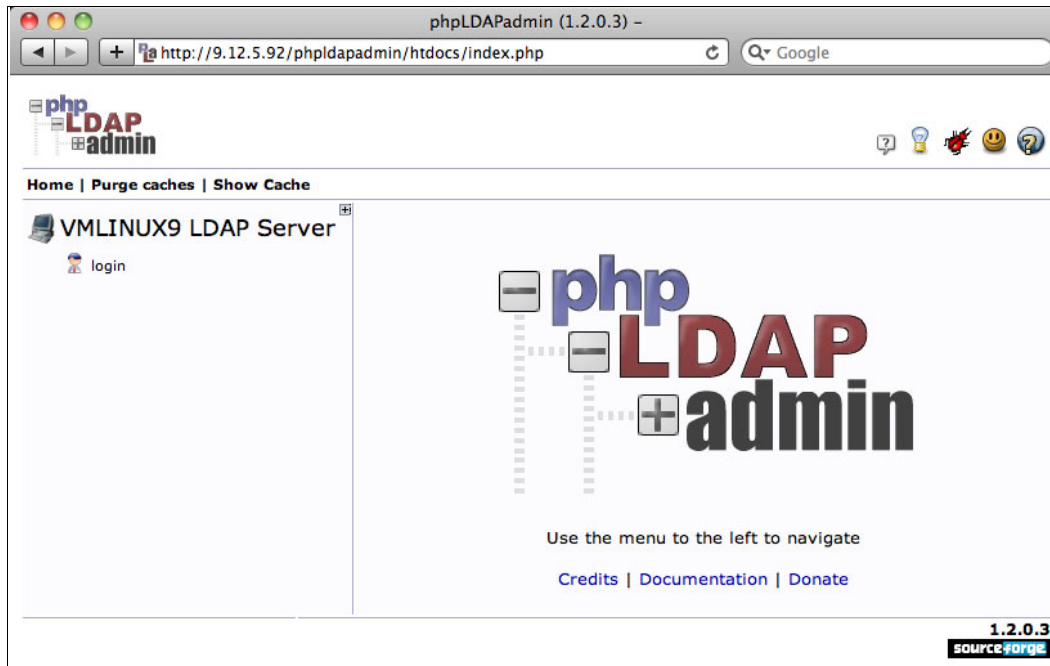


Figure 3-1 The phpLDAPAdmin login panel

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 recommend that you do this in a real LDAP environment.

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 3-11, to define the root objects of our database.

Example 3-11 LDIF to create root database objects in LDBM

```
dn: ou=LDBM,ou=VMLINUX9,o=ITSO
objectClass: top
objectClass: organizationalUnit
ou: LDBM
```

```
dn: ou=People,ou=LDBM,ou=VMLINUX9,o=ITSO
objectClass: top
objectClass: organizationalUnit
ou: People
```

We logged in again after defining the database objects, and saw the panel shown in Figure 3-2.

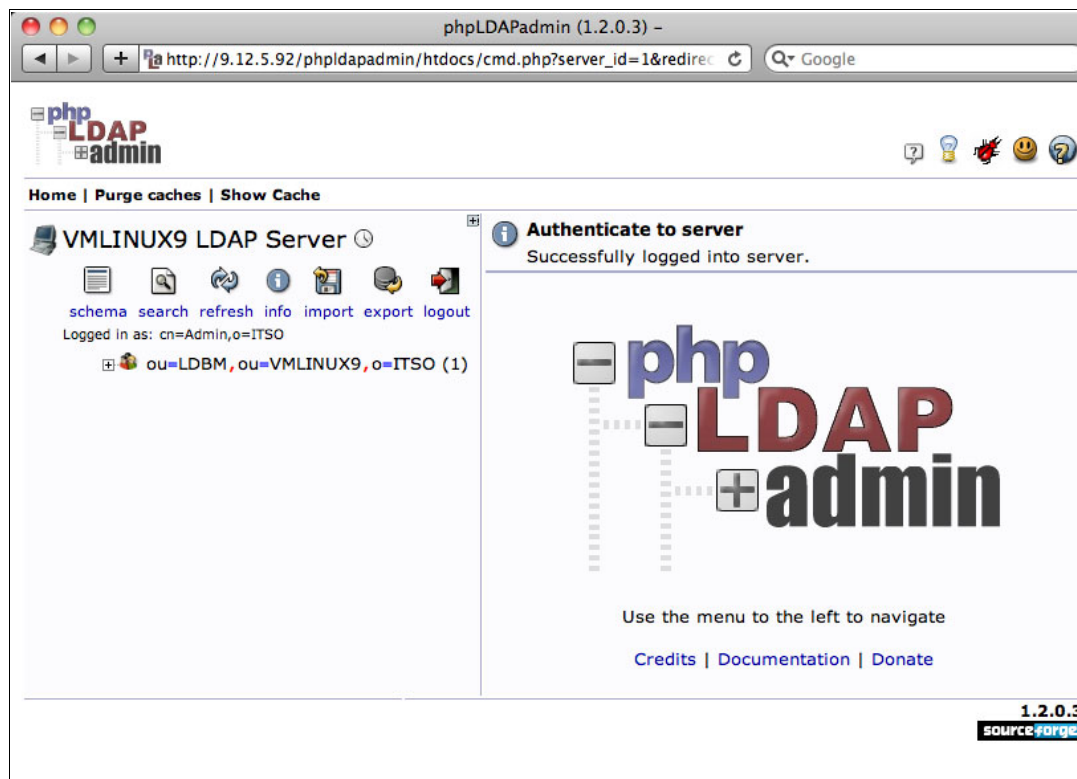


Figure 3-2 Logged in to phpLDAPAdmin using our z/VM LDAP server

Having successfully logged on to phpLDAPAdmin, we looked around at what functions we were able to perform.

3.4.2 Common schemas supporting phpLDAPAdmin

When we selected the ou=People record, we received a number of errors in the phpLDAPAdmin panel, as shown in Figure 3-3.

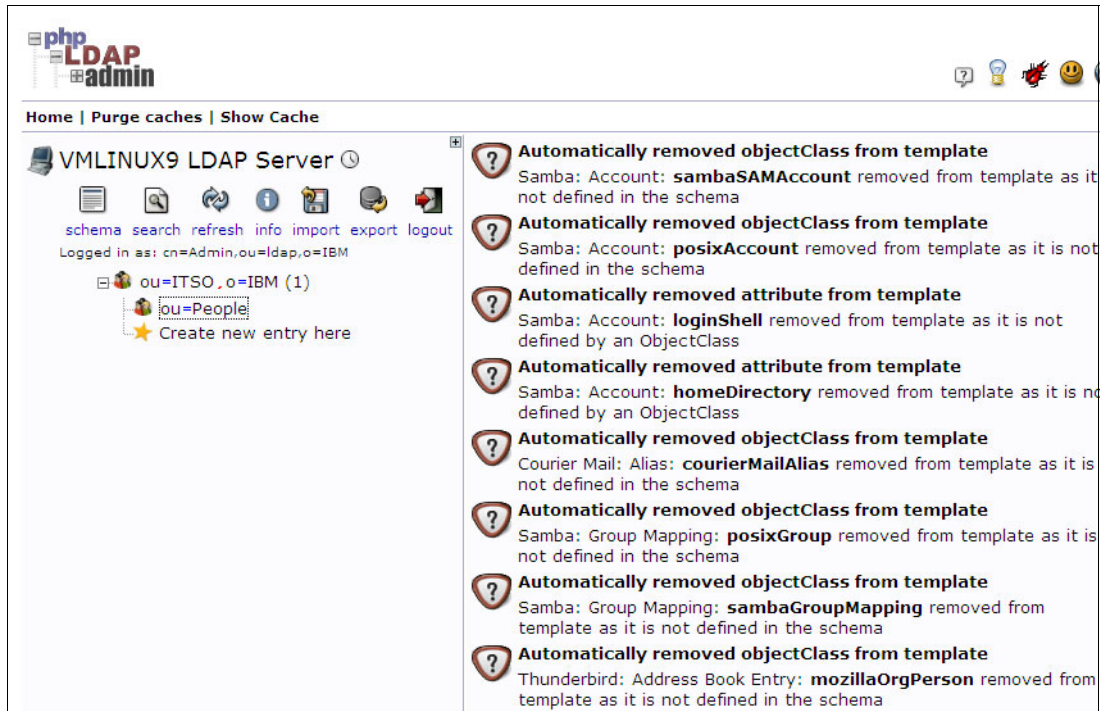


Figure 3-3 Template errors when selecting an object in phpLDAPAdmin

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, e-mail 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 3-3 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 3.3, “Extending the LDBM schema” on page 54, 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 3-12 shows the change in the phpLDAPAdmin's config.php file to turn off the warning.

Example 3-12 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.

3.4.3 Updating LDBM using phpLDAPAdmin

When our phpLDAPAdmin installation seemed functional, we decided to use 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 3-4 on page 64, opened.

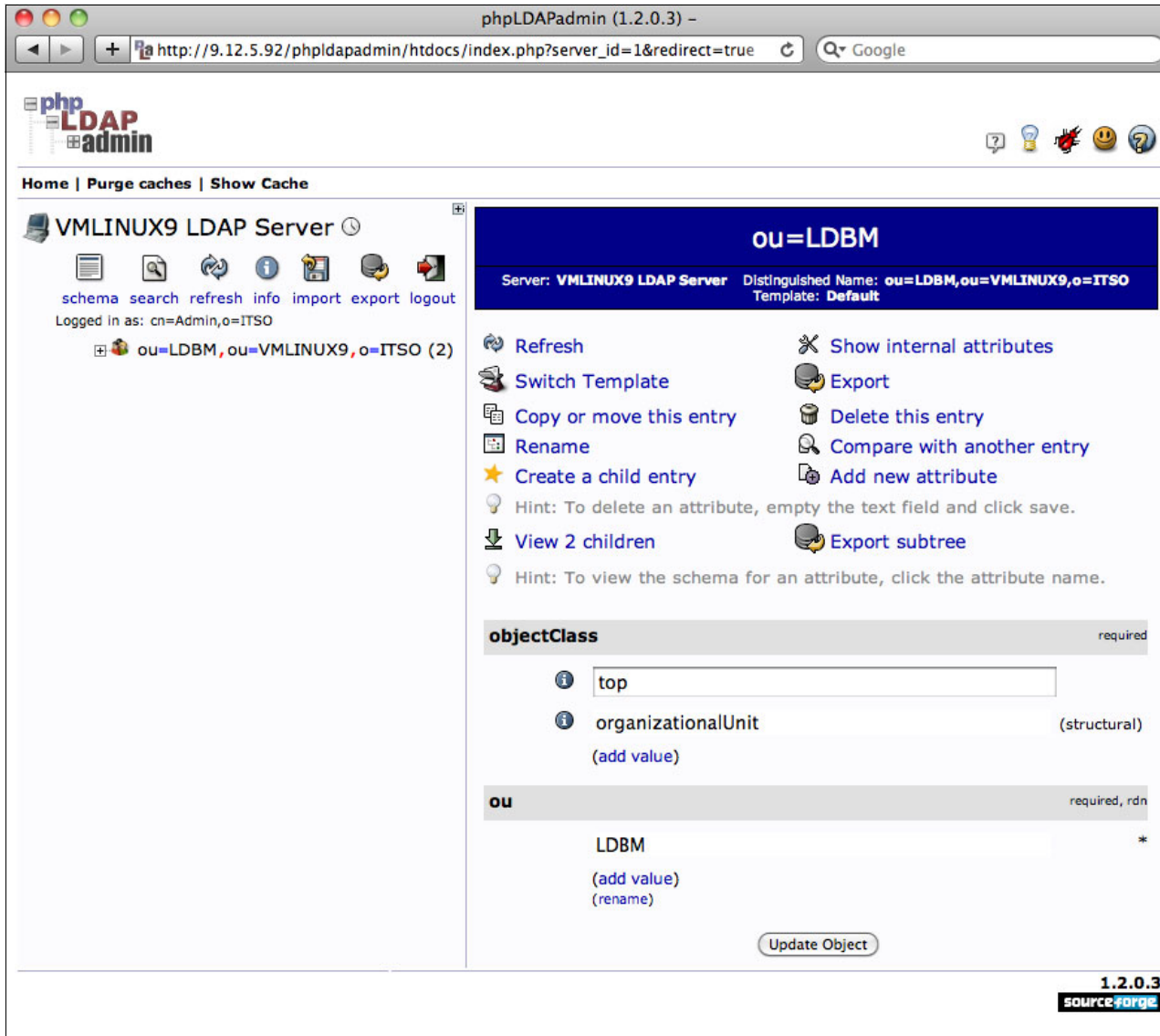


Figure 3-4 Viewing LDAP object attributes with phpLDAPadmin

We clicked **Create a child entry**.

Figure 3-5 shows the next stage of the process, where phpLDAPAdmin prompts us for the template to use for creating the new LDAP object.

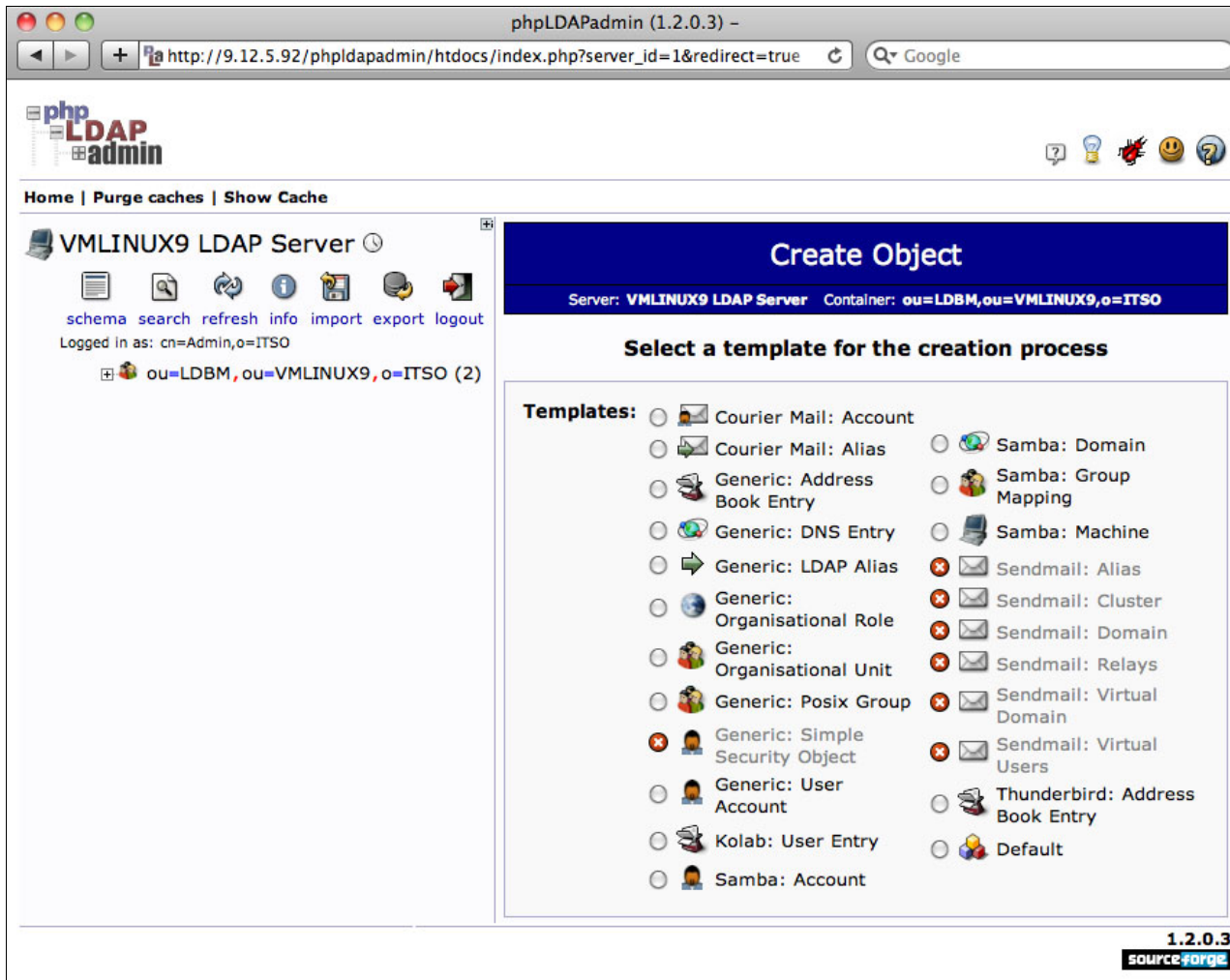


Figure 3-5 Selecting the template to use to create an object

Note: In Figure 3-5 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 3.4.2, “Common schemas supporting phpLDAPAdmin” on page 62.

To create the Groups organizational unit object, we clicked **Generic: Organisational Unit**.

The panel, shown in Figure 3-6 opened and we entered the name of the organizationalUnit object to be created.

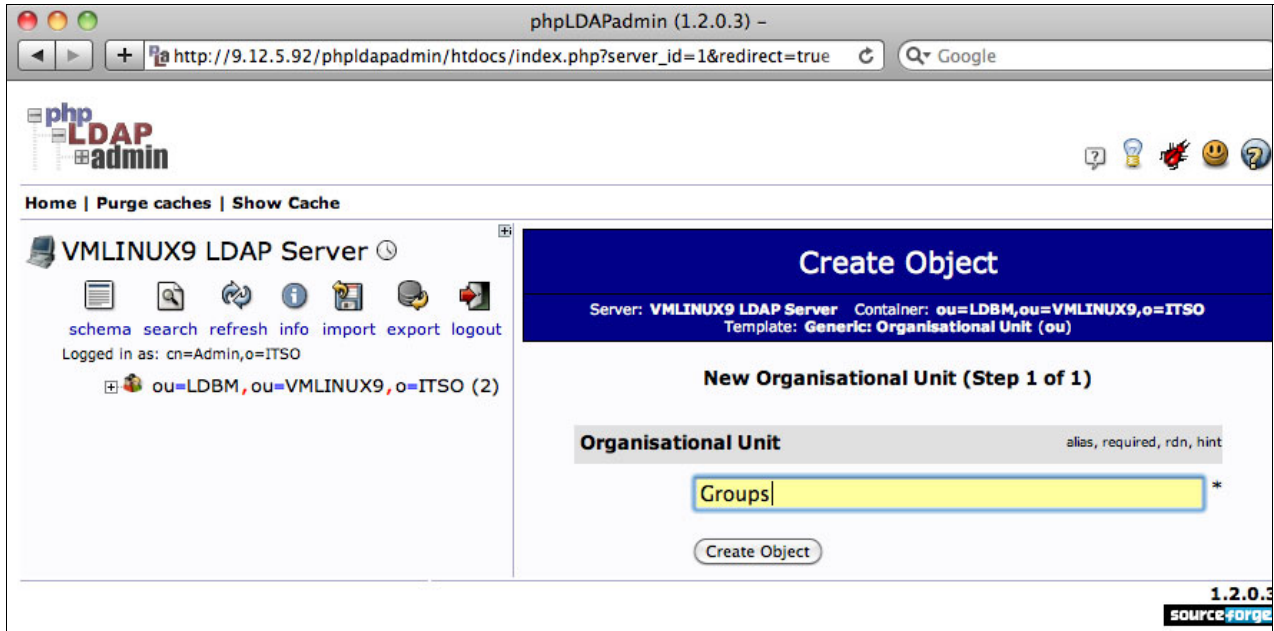


Figure 3-6 Specifying the name of the Organisational Unit object

We clicked **Create Object** and then phpLDAPAdmin asked us to confirm the operation we were about to perform. This is shown in Figure 3-7.

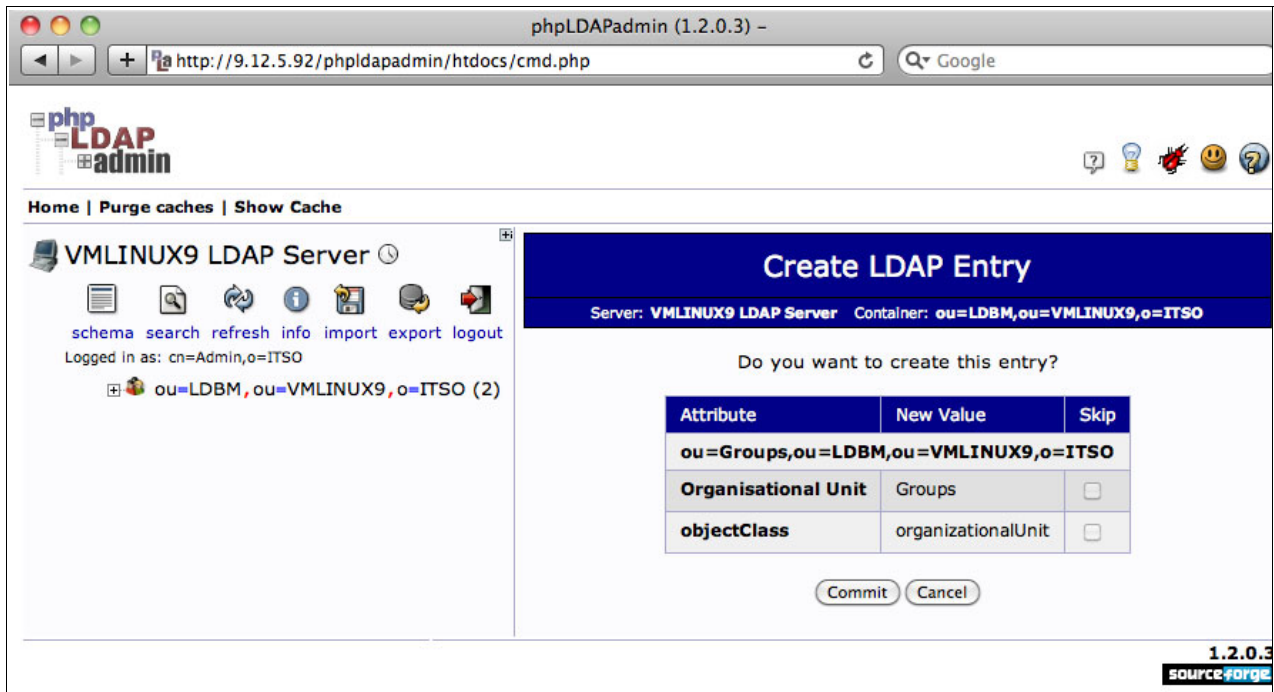


Figure 3-7 Confirmation of object creation

We clicked **Commit**, and then phpLDAPAdmin created the database object for us. The panel, shown in Figure 3-8 on page 67 opened, confirming the object creation and allowing us to edit it if necessary.

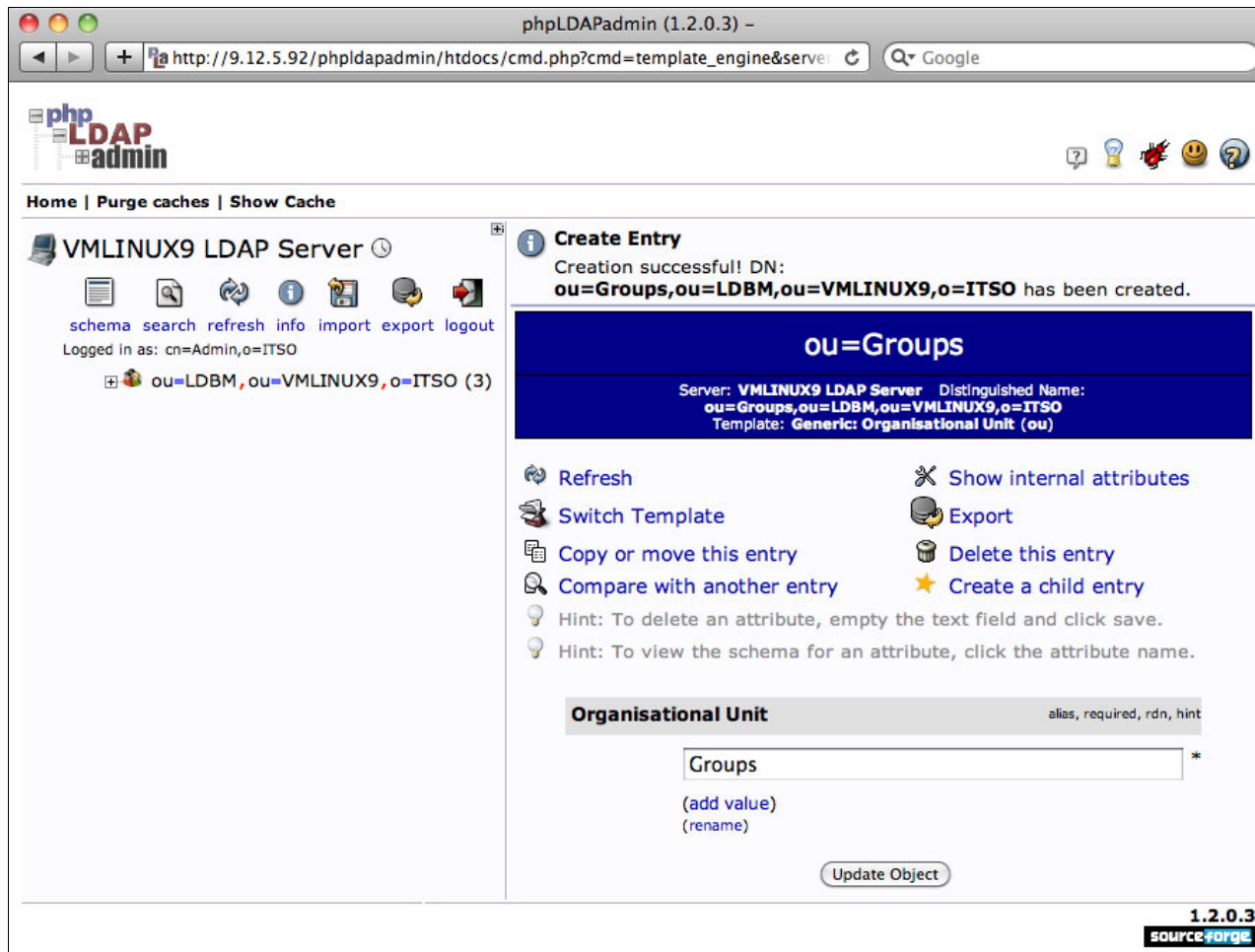


Figure 3-8 Object details, with confirmation of object creation

Next, we tried to create 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 3-9 on page 68, opened. We then encountered a problem with phpLDAPadmin and the way that it allocates group (and user) ID numbers.

The phpLDAPadmin administers the allocation of user and group ID numbers automatically. The way it does this is to search the LDAP database and find the highest number already allocated, then it adds one to that number and uses that for the new object. However, this fails no ID numbers are already in the database, as would be the case when adding the first record. Because phpLDAPadmin was set up to automatically allocate user and group numbers, the field was marked as read-only and did not allow a GID to be entered manually.

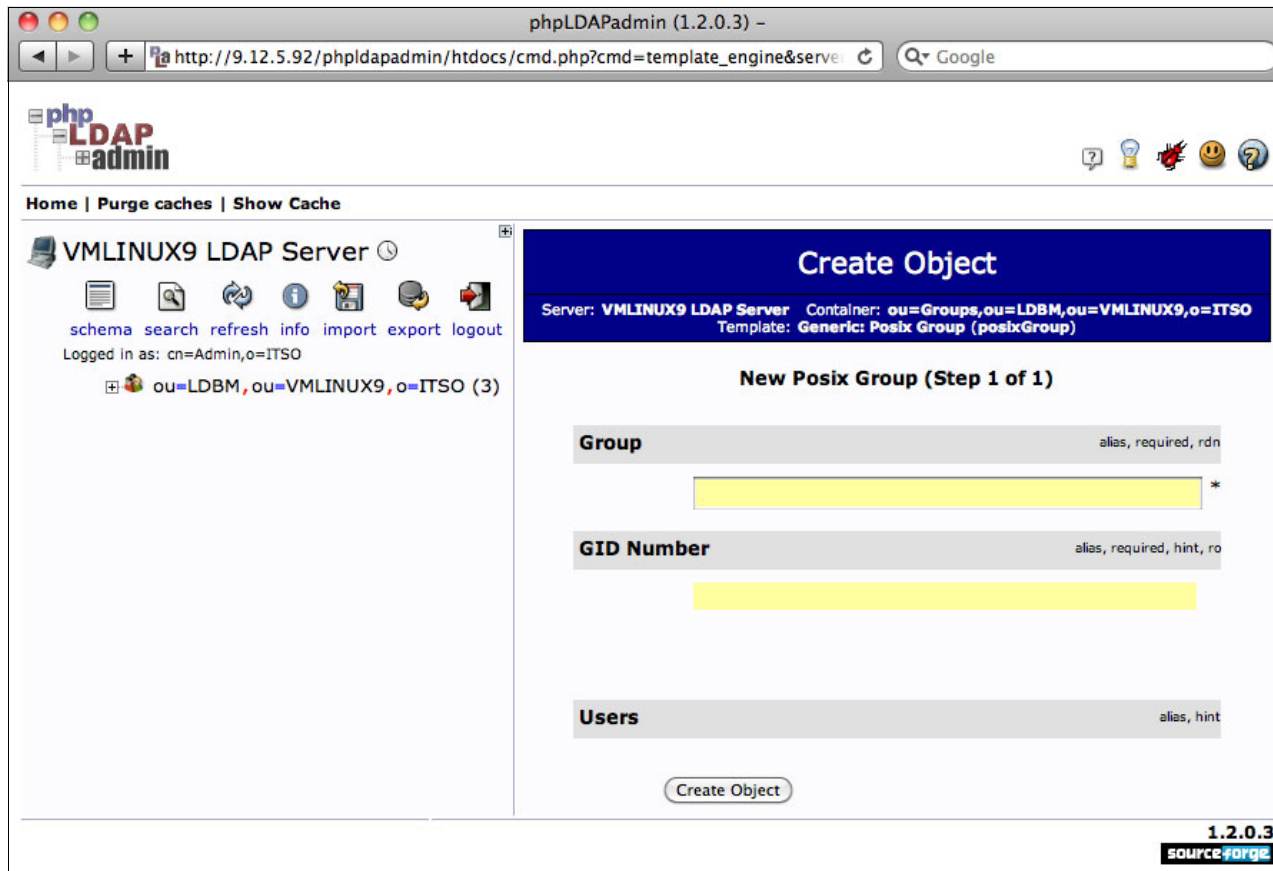


Figure 3-9 The New Posix Group panel, with the blank GID Number field

To work around this, we could have set phpLDAPadmin to not allocate IDs automatically. We do like the feature, however, so we added a *seed value* to the Groups object itself. We went back to the panel to edit the Groups object (we expanded the tree view on the left panel to expose the ou=Groups entry, then selected it). We had to select the **Switch Template** option on this panel, then **Default**, to give us the ability to add a new object class to the definition; this is shown in Figure 3-10 on page 69.

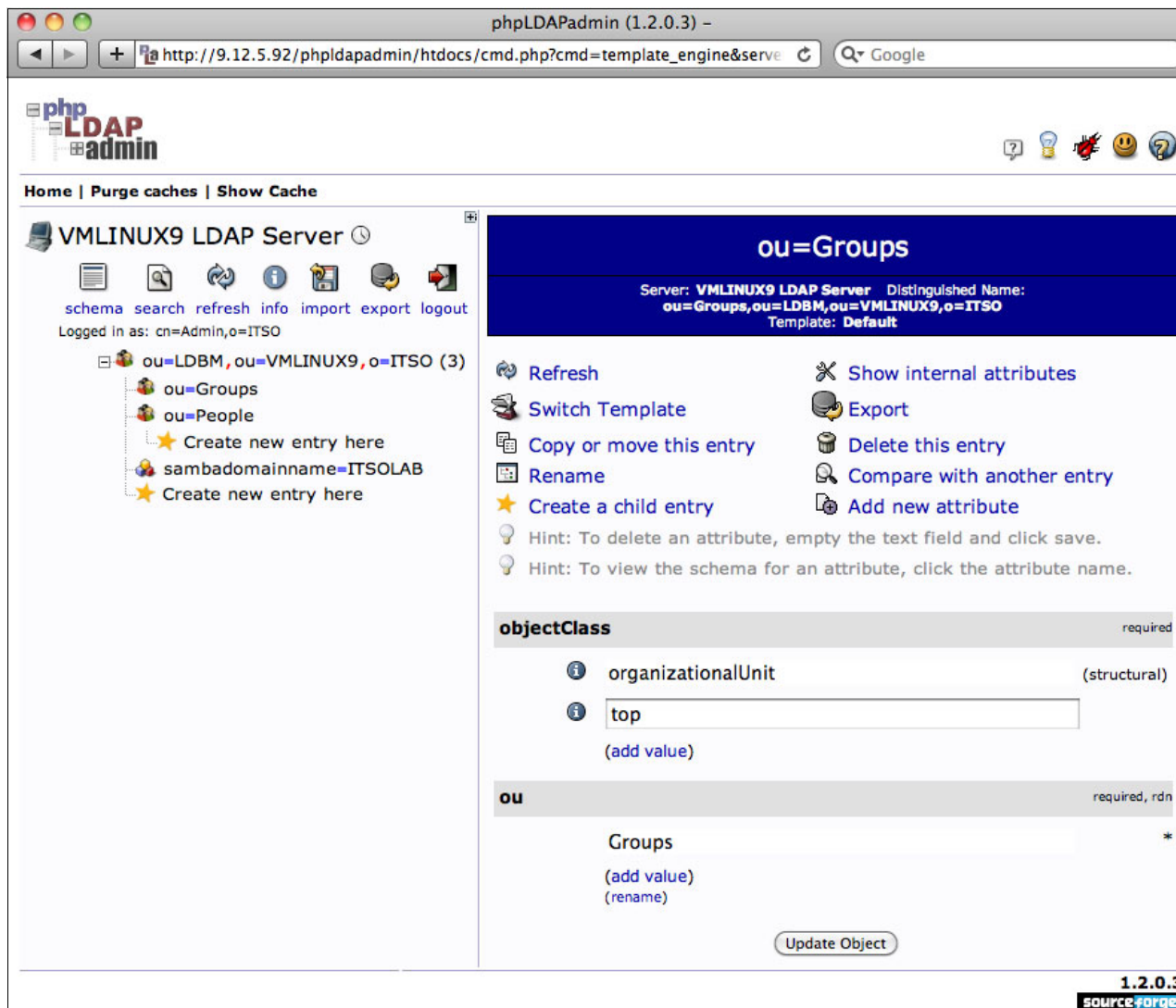


Figure 3-10 Editing the ou=Groups object to add an object class

We had to find an appropriate object class that would allow us to add the gidNumber attribute, without possibly causing the ID to be erroneously recognized as a group itself. A small schema definition for the phpLDAPadmin application itself provided the uidNumber and gidNumber attributes in object definitions by themselves. The schema file can be found in the doc directory of the phpLDAPadmin source, as the file uidpool.schema. We used the script we introduced in “Schema conversion script” on page 57 to add this schema to our database, and then added the gidPool objectclass to our ou=Groups object.

As shown in Figure 3-10, under the objectClass display of ou=Groups, we clicked **(add value)**, which opened the panel shown in Figure 3-11 on page 70.

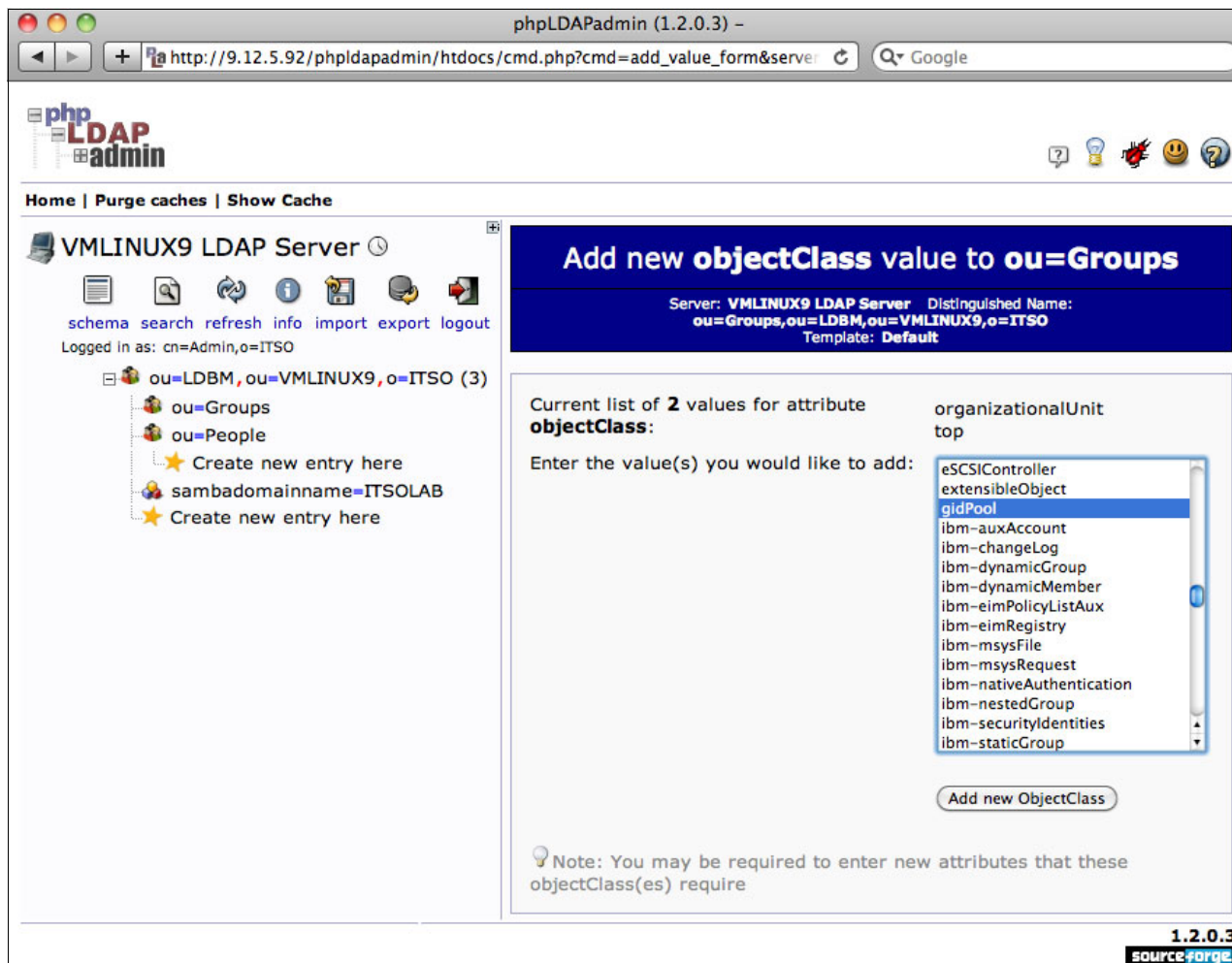


Figure 3-11 Selecting the object class to add to the ou=Groups object

As shown in Figure 3-11, we selected **gidPool** from the list. We then clicked **Add new ObjectClass**, and phpLDAPadmin opened the panel shown in Figure 3-12 on page 71. The definition of the gidPool object class has attributes that are defined as MUST in the schema, so phpLDAPadmin required that we enter values for these attributes.

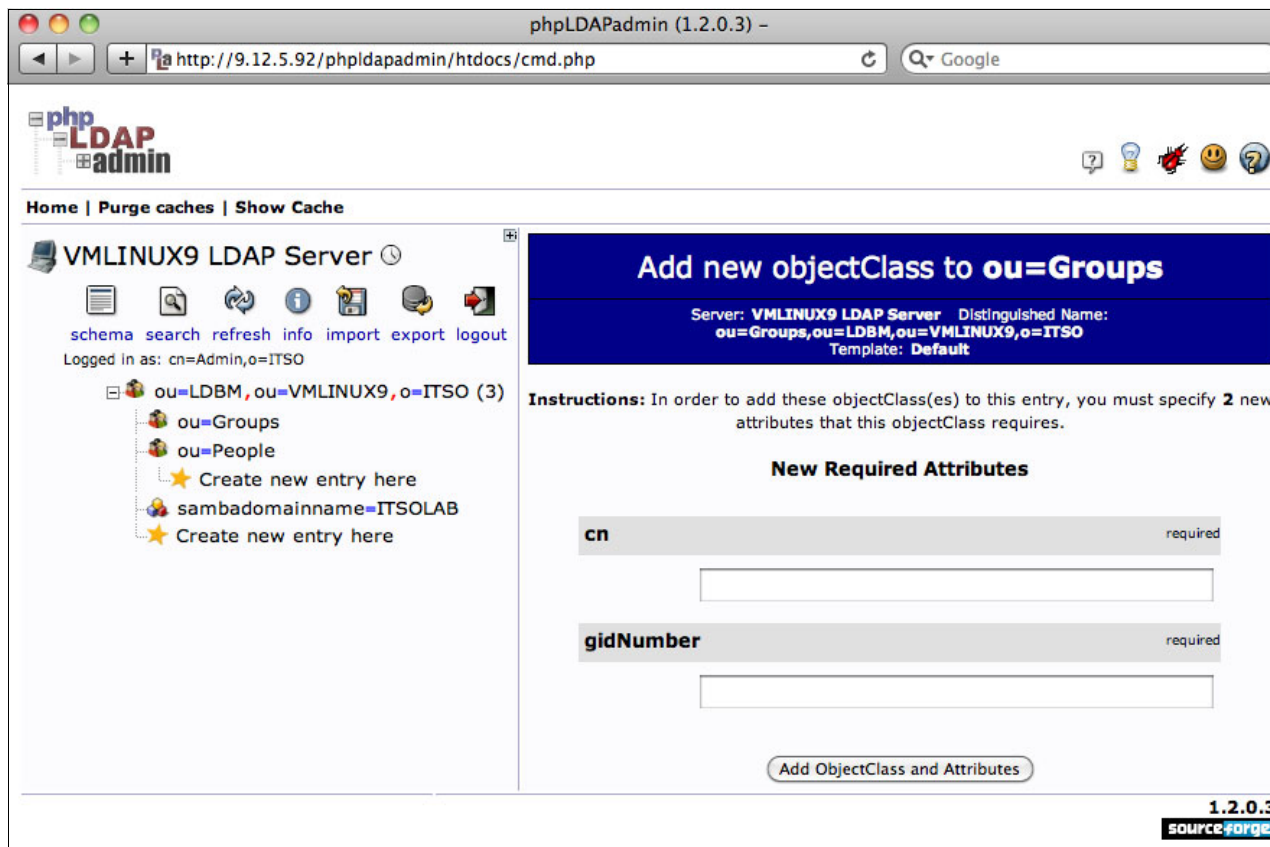


Figure 3-12 Completing the mandatory details required by the new object class

We filled in values for the required attributes (we simply used the name of the object as the `cn`, and the number 500 as the `gidNumber`), then clicked **Add ObjectClass and Attributes**. This gave us the display shown in Figure 3-13 on page 72, which asked us to confirm the operation.

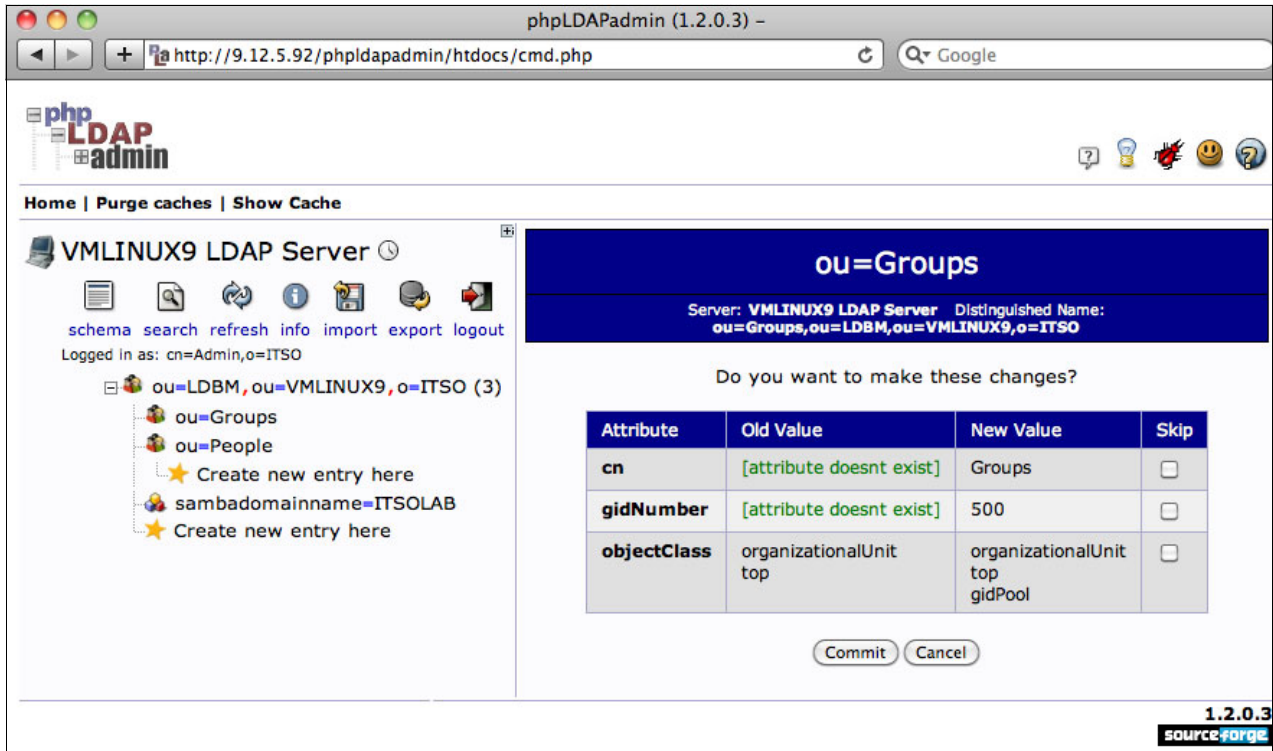


Figure 3-13 Confirming the change to the object definition

After we clicked **Commit**, phpLDAPadmin made the update and presented the modified object shown in Figure 3-14 on page 73.

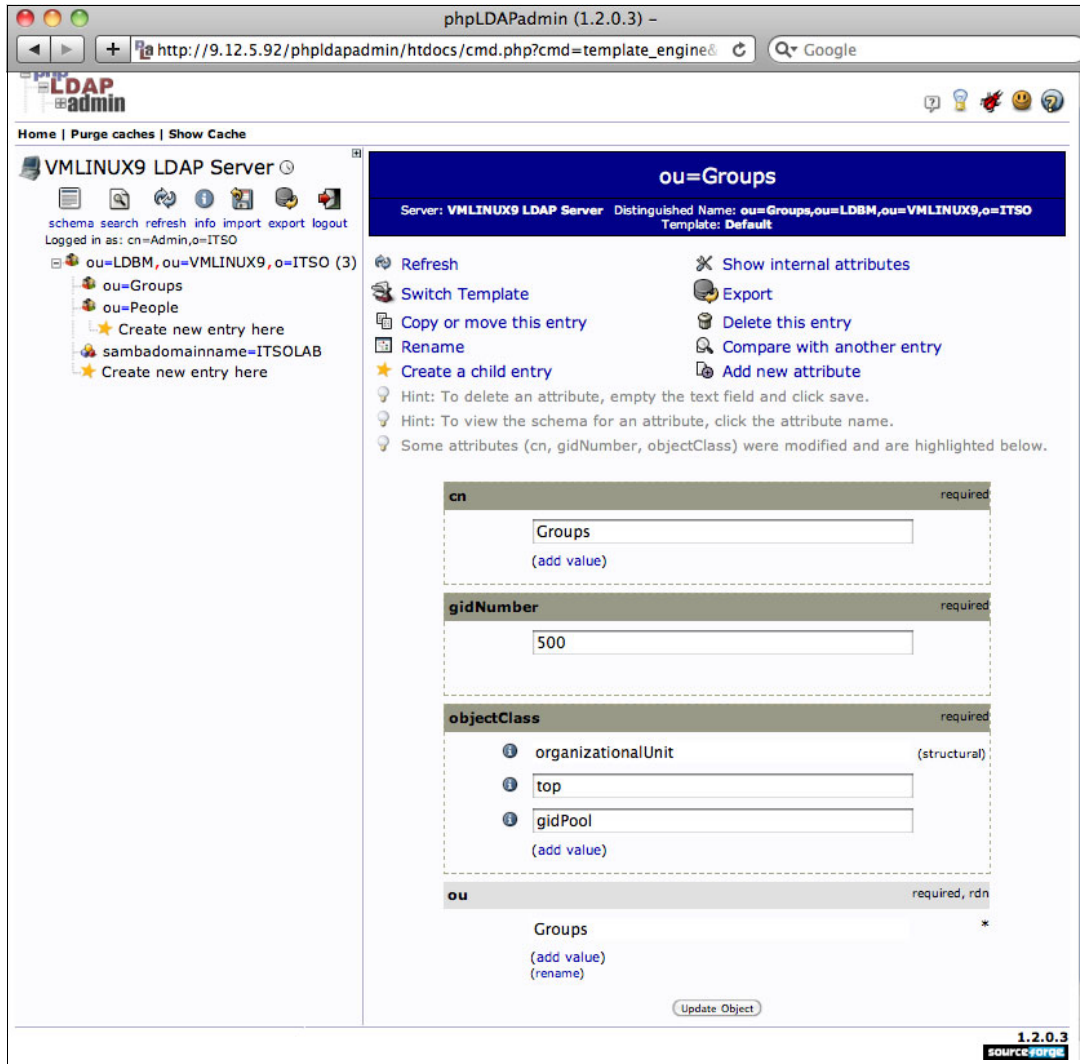


Figure 3-14 Object edit panel, with confirmation of updated attributes

We were then able to select the **Create a child entry** option again and select the **Generic: Posix Group** item. This time, instead of the blank `gidNumber` field seen in Figure 3-9 on page 68, we see the field populated with a `gidNumber` generated by phpLDAPAdmin (as shown in Figure 3-15 on page 74).

Note: We could also have simply added the first group manually by using an LDIF file. This would have been sufficient to prime the database for subsequent group additions using phpLDAPAdmin.

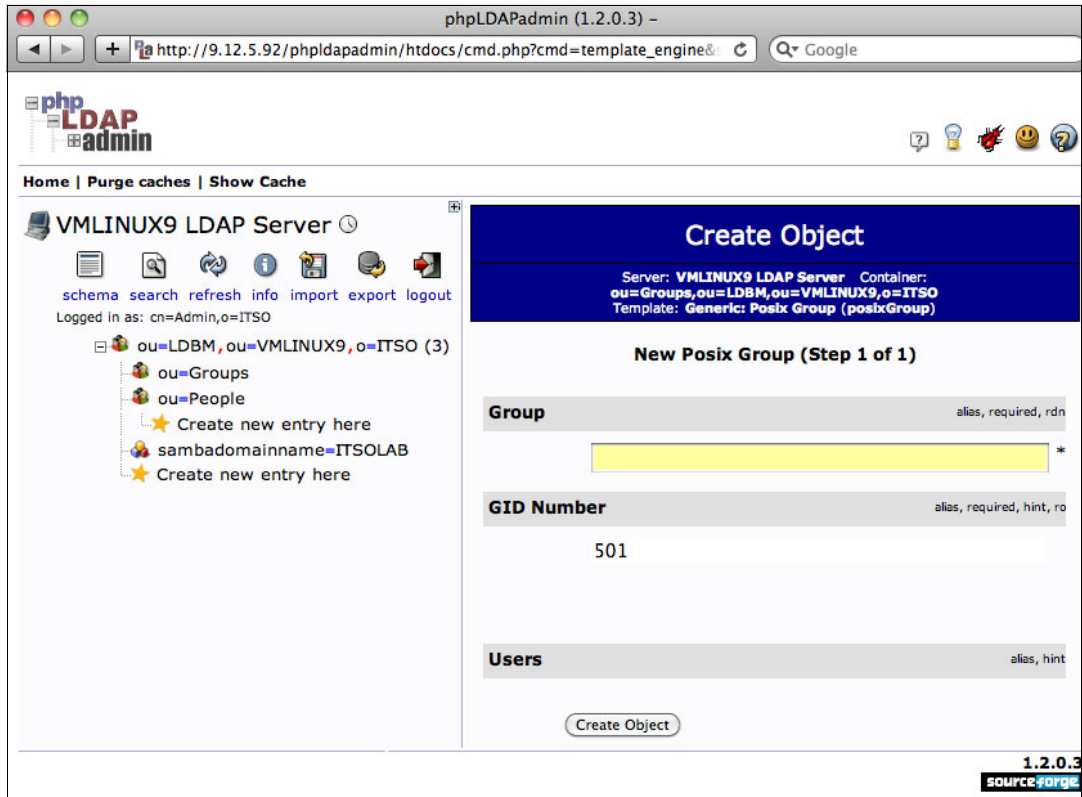


Figure 3-15 Creating a new LDAP object using the Generic: Posix Group template

We filled in the group name as `itsousers` and selected **Create Object**. The confirmation panel shown in Figure 3-16 opened.

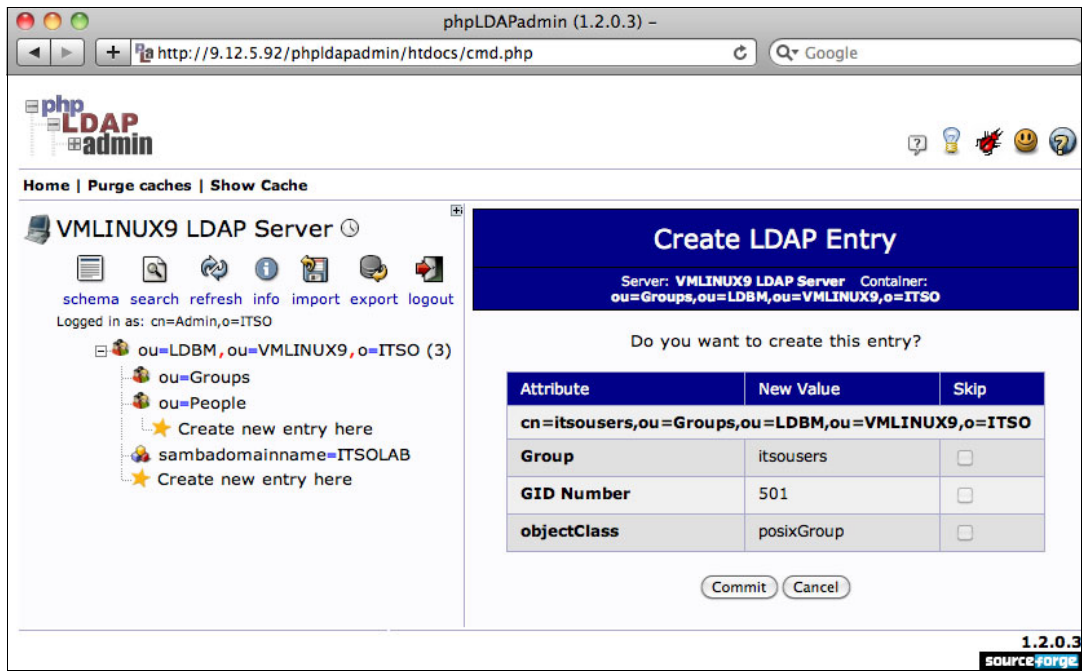


Figure 3-16 Confirming the details of the object to be added

We clicked **Commit**, and phpLDAPadmin created the new posixGroup object. This is shown in Figure 3-17.

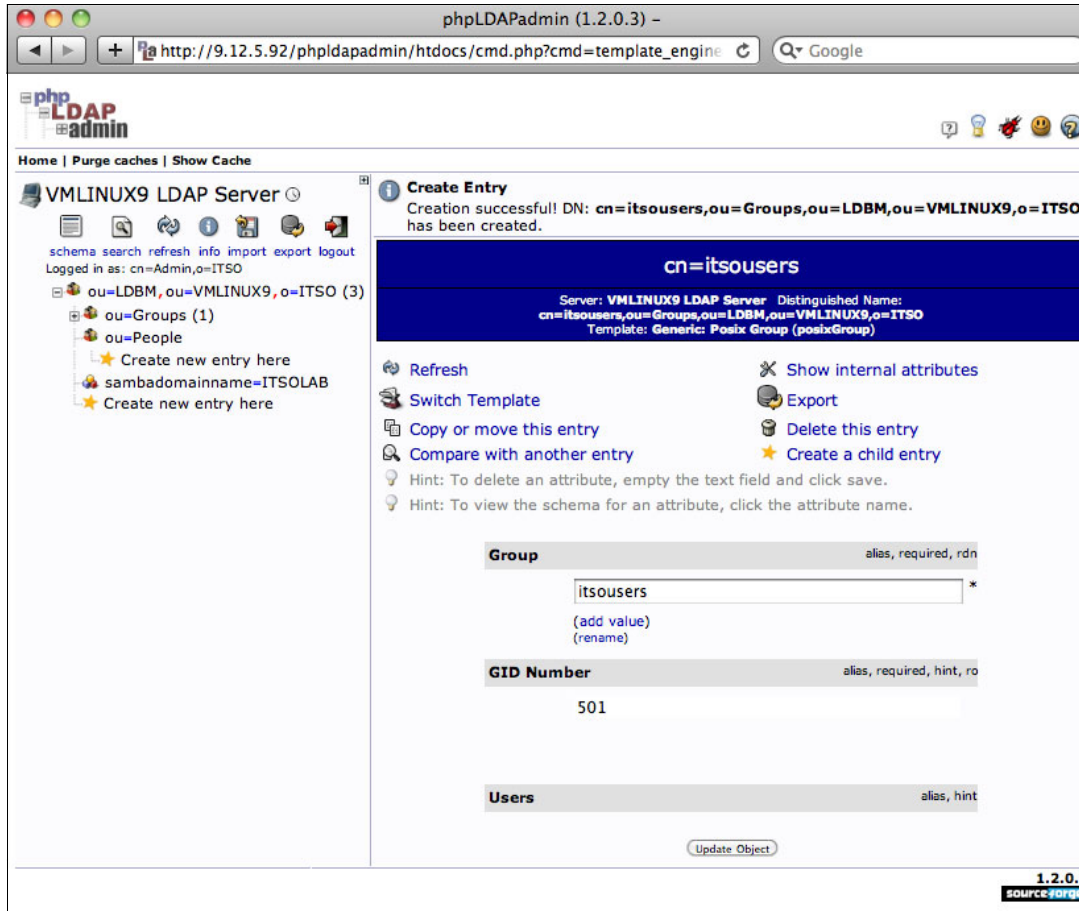


Figure 3-17 Object edit panel, with confirmation of the newly created object

We tried to add a user object at this time, but we encountered a similar problem to our attempt to add the itsusers group: the uidNumber field was blank and read-only. We added the uidPool object class to the ou=People object to prime phpLDAPadmin's search for a user ID. The end result of this modification is shown in Figure 3-18 on page 76.

Note: Again, as we mentioned with the GID case, we could simply have added the first user ID to LDAP using an LDIF file.

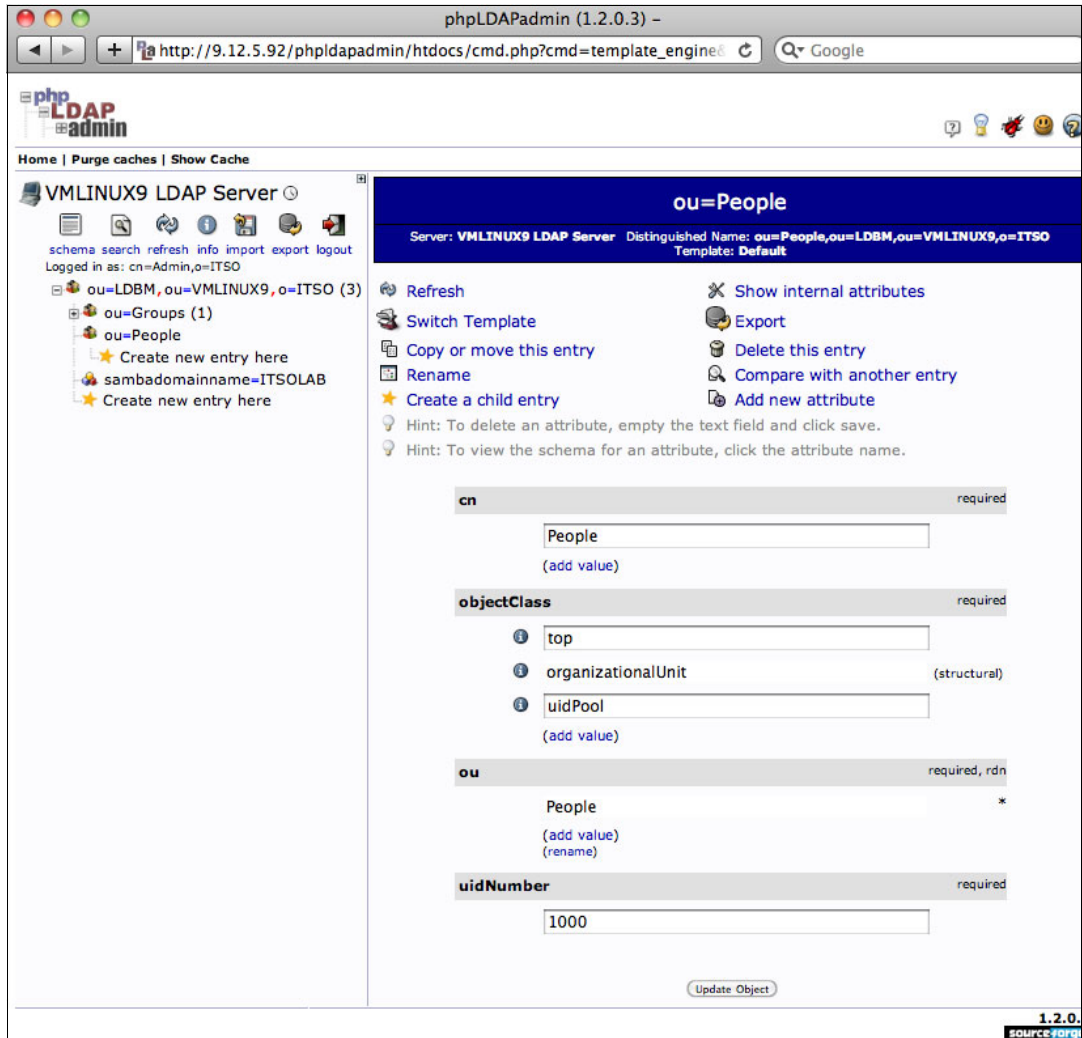


Figure 3-18 The ou=People object after the uidPool object class was added

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 3-19 on page 77, where we filled in the details of the user account to be created. The figure shows that the uidNumber field has been 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.

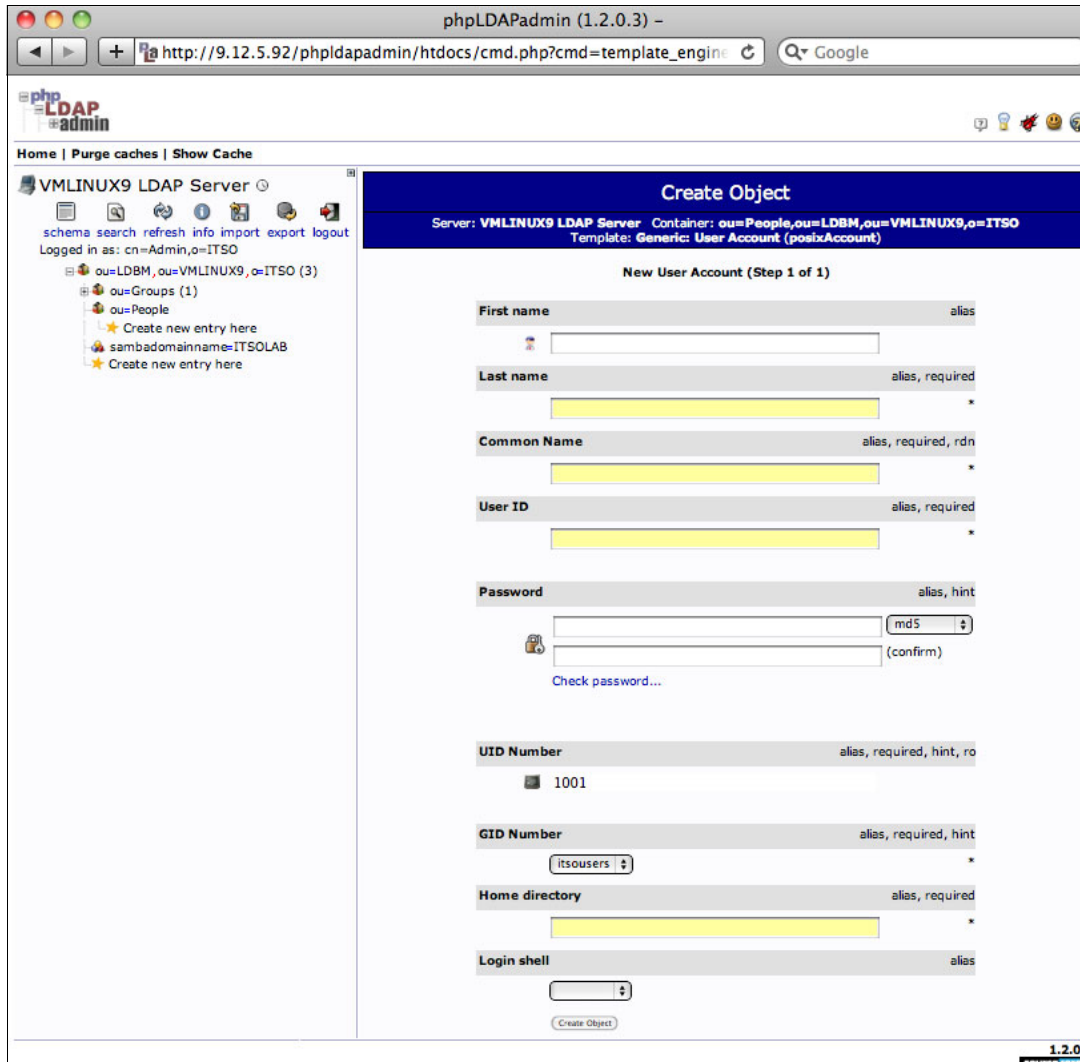


Figure 3-19 Adding a new LDAP object using the Generic: Posix Account template

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. The 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 3-20 on page 78 opened.

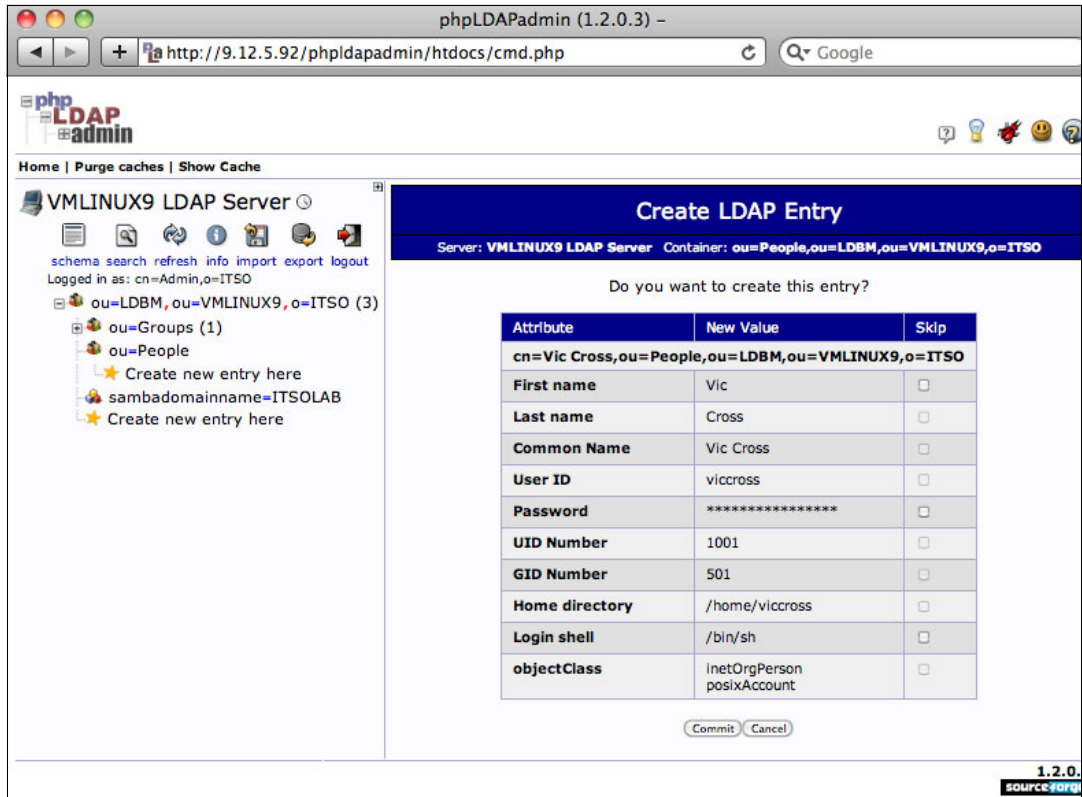


Figure 3-20 Confirming the attributes of the new Posix Account object

We clicked **Commit** and received the error message shown in Figure 3-21.

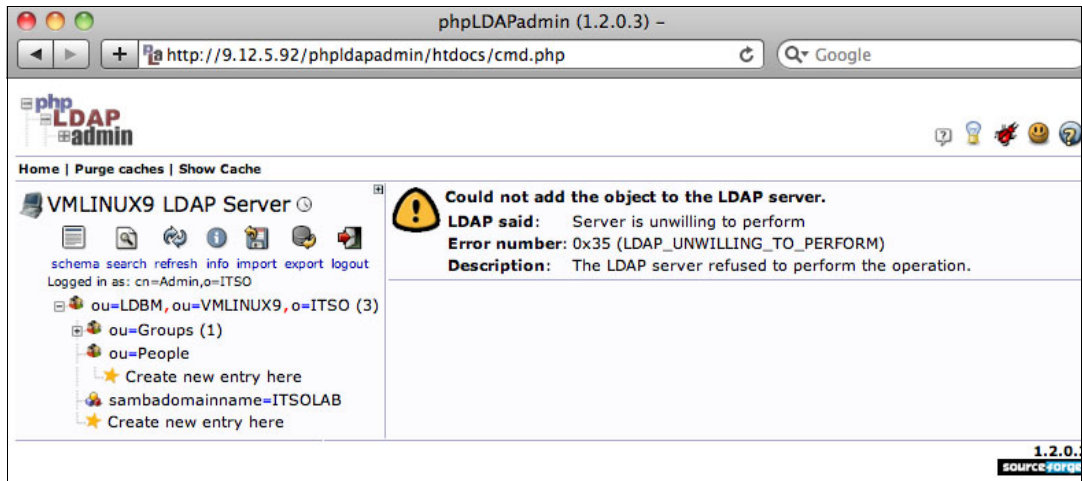


Figure 3-21 A phpLDAPadmin error message on attempt to create the Posix Account object

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 used the ERROR debug level. Example 3-13 shows the enabling of ERROR level debug logging on our z/VM LDAP server using Dynamic Server Operation.

Example 3-13 Enabling error logging in the z/VM LDAP server

msg ldapshr 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 3-14.

Example 3-14 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=ITS0', entryDN='cn=Vic Cross,ou=People,ou=LDBM,ou=VMLINUX9,o=ITS0', 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 added 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 back end on the LDAP server was configured for Native Authentication. The z/VM LDAP server appears to not allow the add operation on the userPassword attribute when Native Authentication is enabled (possibly to avoid a misunderstanding on where the password is validated).

To remove the error, we removed the userPassword attribute from the update phpLDAPadmin that was sent to the LDAP server. We clicked the **Back** button on the browser, and selected the **Skip** check box next to the userPassword attribute, as shown in Figure 3-22.

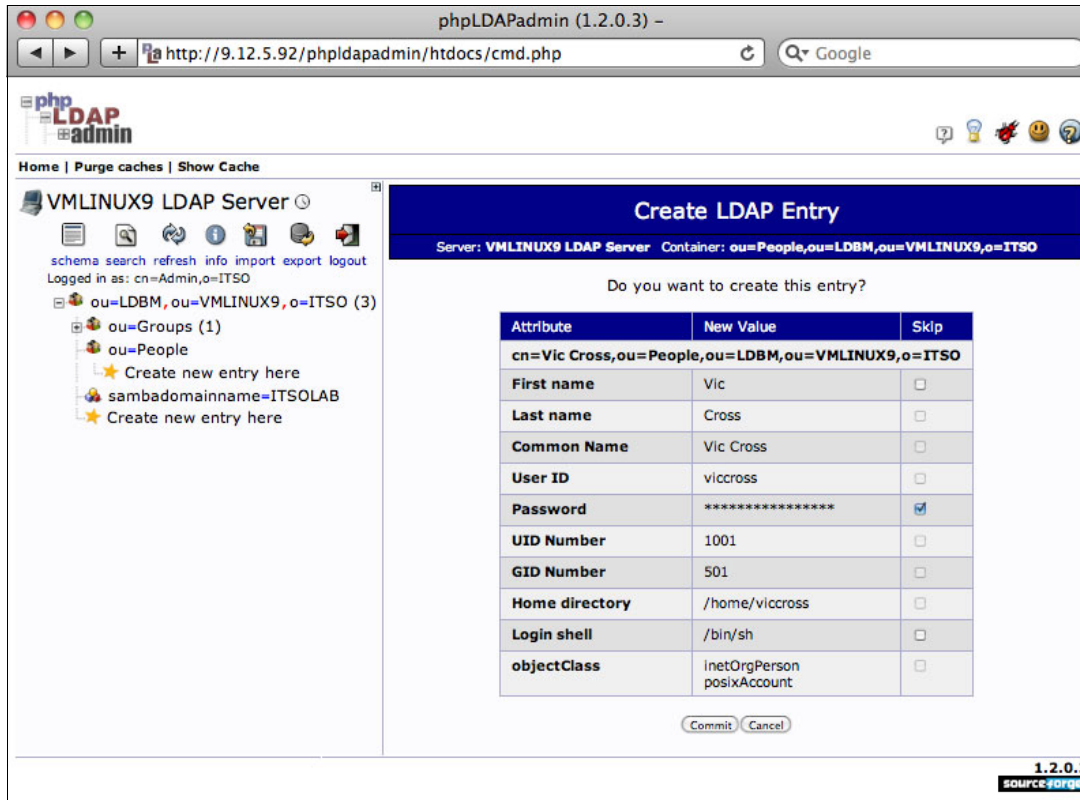


Figure 3-22 Confirming the LDAP update, with the userPassword attribute skipped

This time when we clicked **Commit**, the update was successful, as shown in Figure 3-23 on page 81.

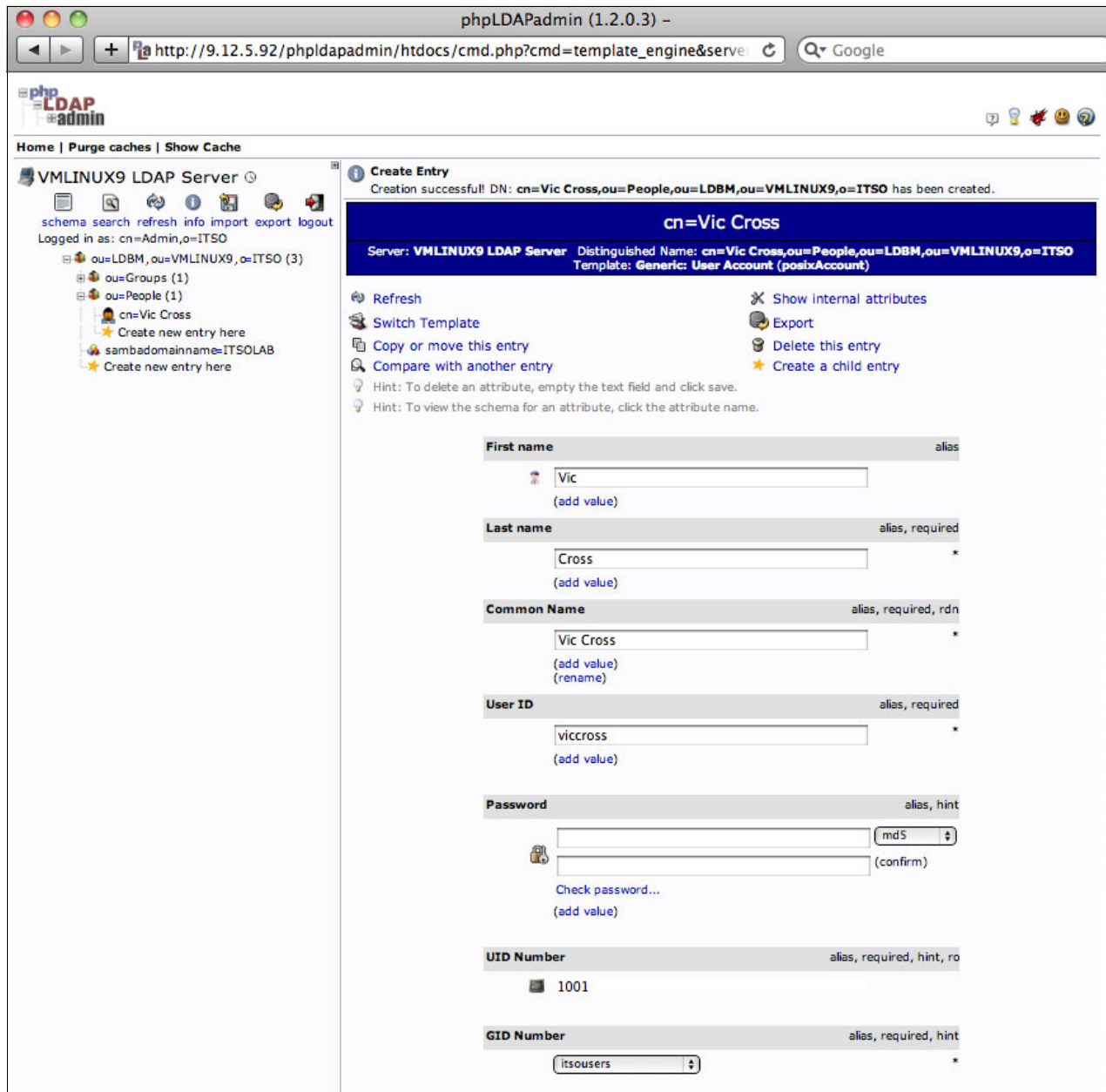


Figure 3-23 Object edit panel, with confirmation of the newly created Posix Account object

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.

In 3.6, “Linux authentication using the z/VM LDAP server” on page 83, we illustrate this by using a Linux system set up to authenticate to our z/VM LDAP server.

3.6 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.

3.6.1 Using YaST to enable LDAP on SLES 11

We used Yet another Setup Tool (YaST) to activate the LDAP client in our SLES 11 server. We used the command `yast2` to invoke YaST, then selected **LDAP Client** from the Network Services menu shown in Figure 3-24.

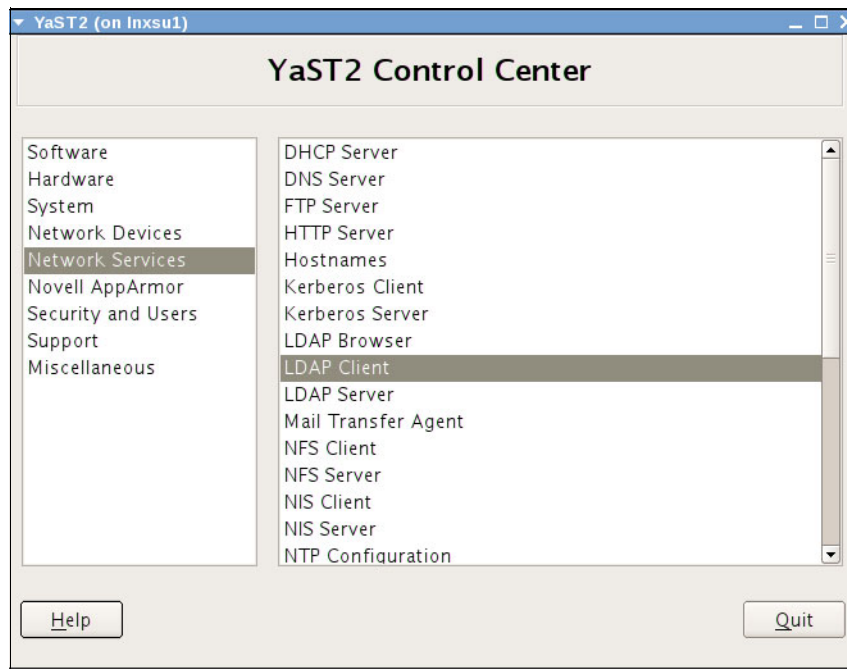


Figure 3-24 Selecting the LDAP Client configuration utility from the YaST2 Control Center

The LDAP Client configuration utility opens. We checked the **Use LDAP** box, and filled in the details of our z/VM LDAP server, as shown in Figure 3-25.

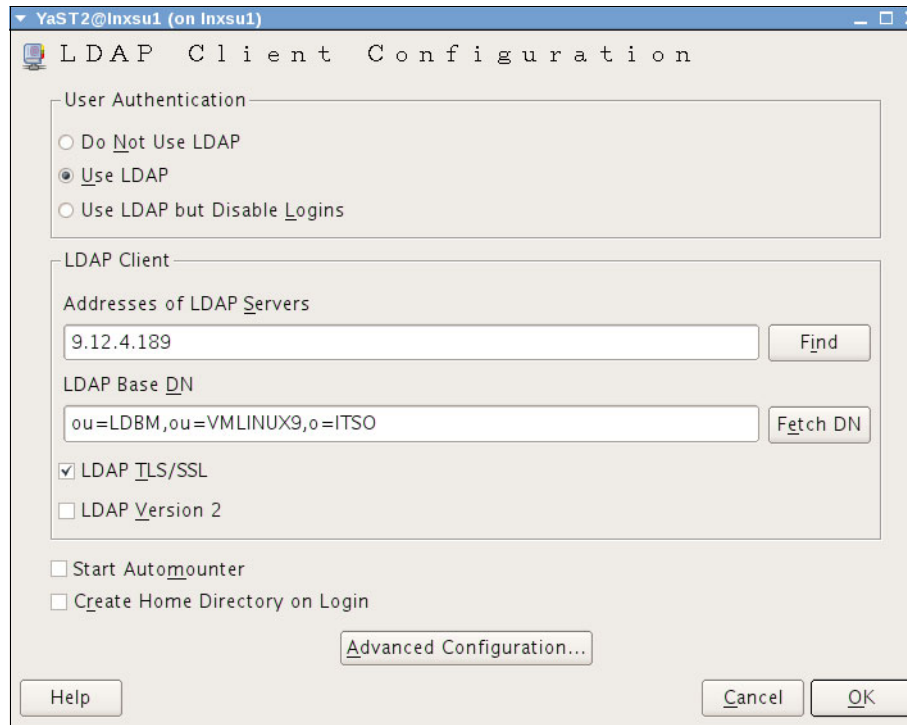


Figure 3-25 Configuring the LDAP Client

We clicked **OK**. A message from YaST, shown in Figure 3-26, indicated that additional packages had to be installed.

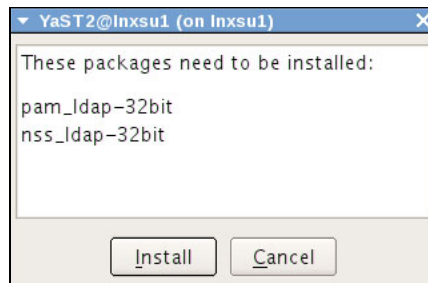


Figure 3-26 YaST2 information message about extra packages required for LDAP

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.

We clicked **Install**. YaST installed the additional required packages. Another message, shown in Figure 3-27 on page 85, informed us to restart the tasks so that they pick up the LDAP client support.

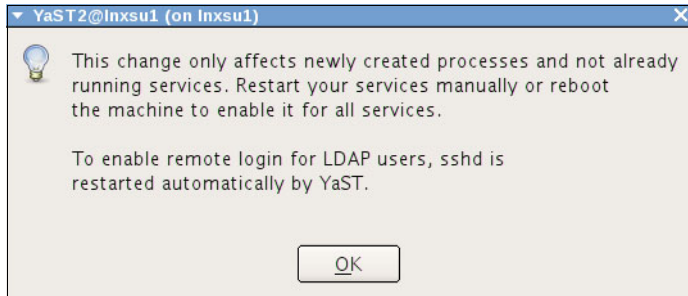


Figure 3-27 Task restart requirement warning from YaST2

We clicked **OK** at this message. The progress panel from YaST2, shown in Figure 3-28, opened to indicate that the system configuration was being performed.

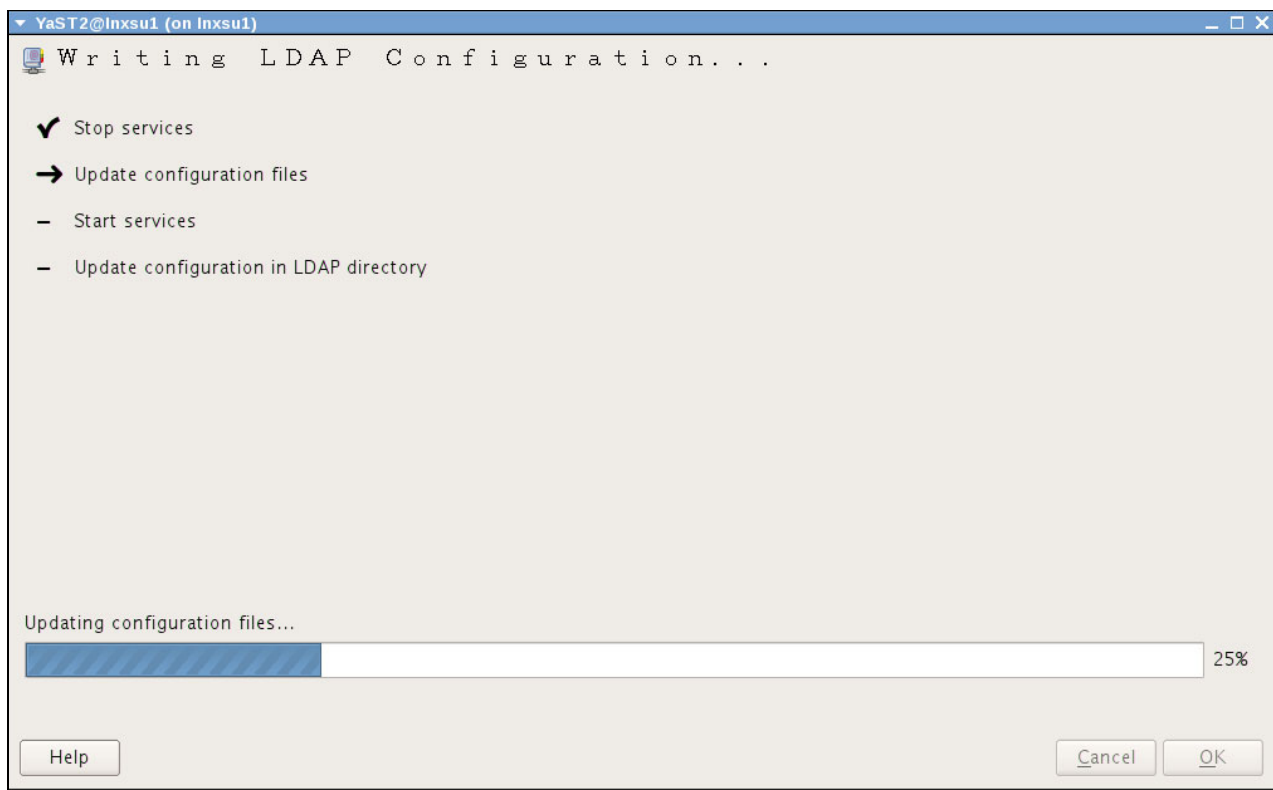


Figure 3-28 YaST2 progress panel showing the activation of the altered configuration

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 3-15, to display the user ID we added earlier.

Example 3-15 Using the ID command to test the LDAP client configuration

```
Inxsu1:~ # id viccross
uid=1001(viccross) gid=501(itsousers) groups=501(itsousers)
```

At this phase, however, we were unable to use this account to log on to Linux. The reason is because we had not yet created the RACF account corresponding to the new LDAP object.

Example 3-16 shows the messages in the system log of our Linux server when we tried to log on.

Example 3-16 Authentication error from pam_ldap

```
Sep 29 08:43:30 lnxsu1 sshd[21592]: pam_ldap: error trying to bind as user "cn=Vic Cross,ou=People,ou=LDBM,ou=VMLINUX9,o=ITS0" (Operations error)
Sep 29 08:43:30 lnxsu1 sshd[21590]: error: PAM: Authentication failure for viccross from 9.12.5.220
```

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, 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 (shown in Figure 3-23 on page 81), we clicked the **(add value)** link under the `objectClass` heading. This displayed the panel shown in Figure 3-29 on page 87, where we selected **ibm-nativeAuthentication** from the list and clicked **Add new ObjectClass**.

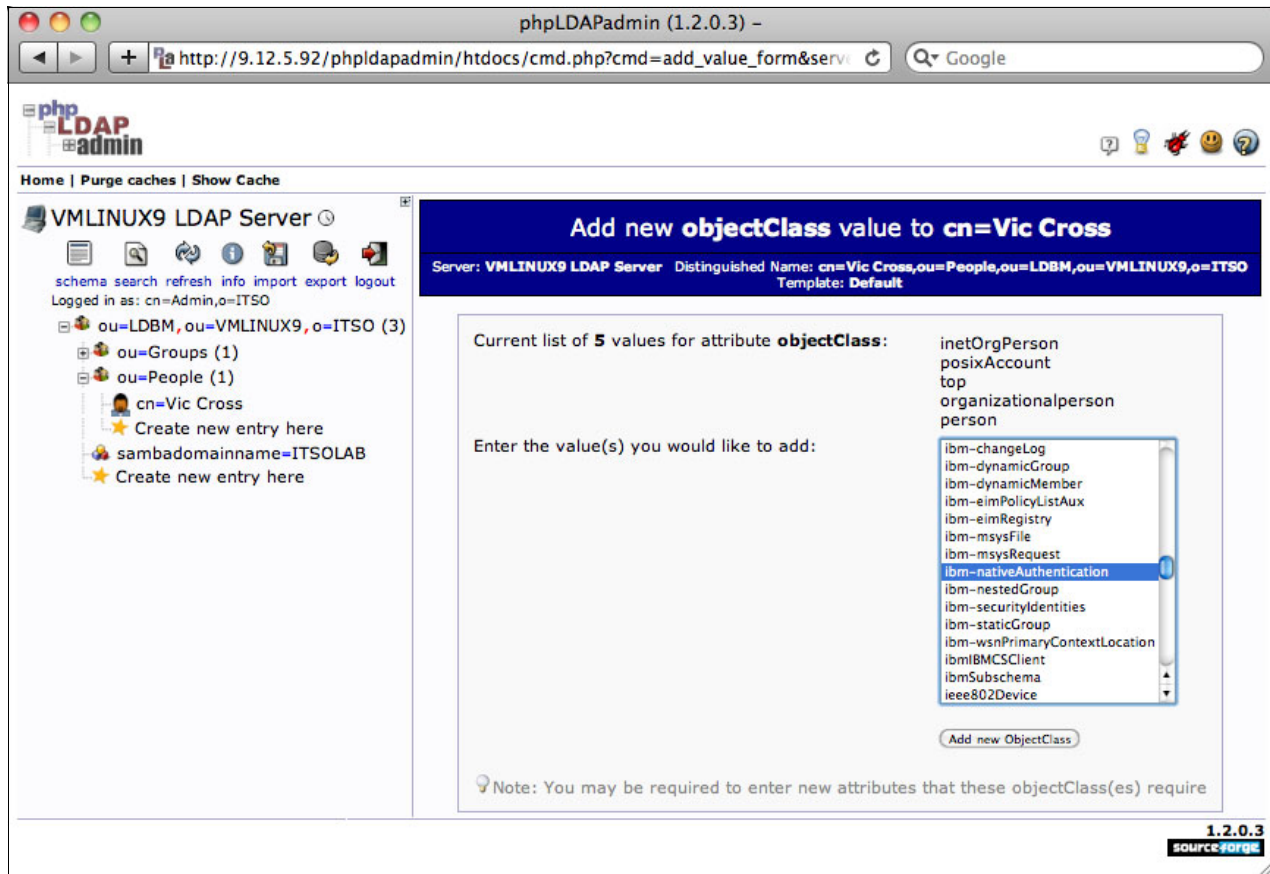


Figure 3-29 Adding the `ibm-nativeAuthentication` object class to an LDAP object

As we saw previously, phpLDAPadmin checks the new object class for any mandatory attributes and prompts for these. This is shown in Figure 3-30 on page 88 (with the value for `ibm-nativeId` filled in).

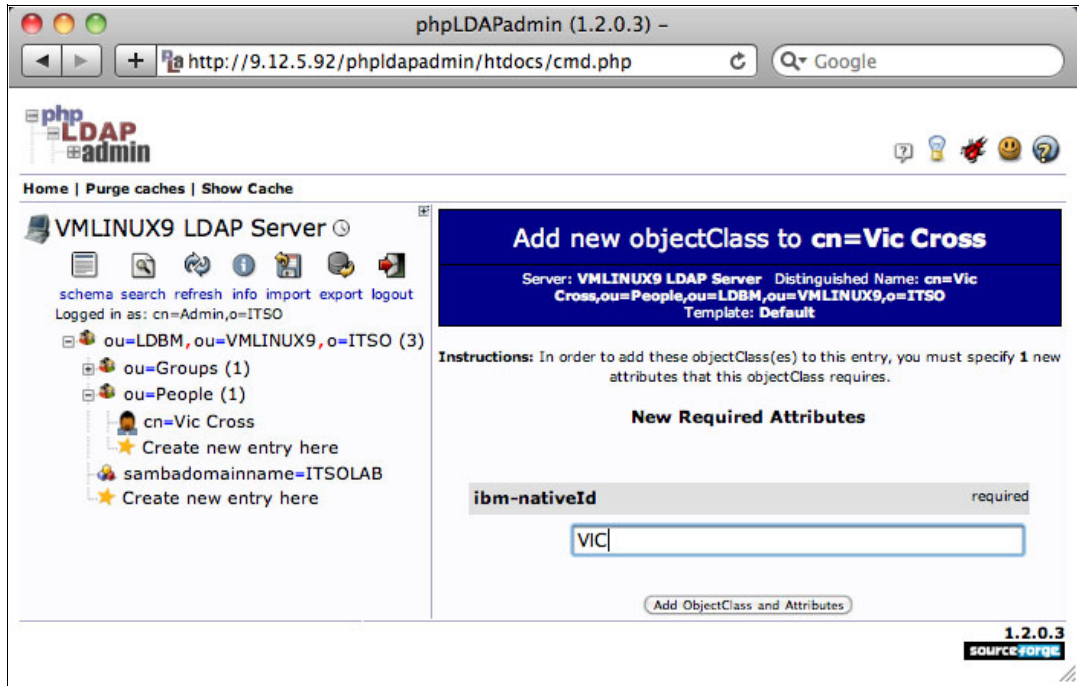


Figure 3-30 Filling in the value for the `ibm-nativeId` attribute

As shown in Figure 3-31, phpLDAPadmin then asks us to confirm our LDAP changes.

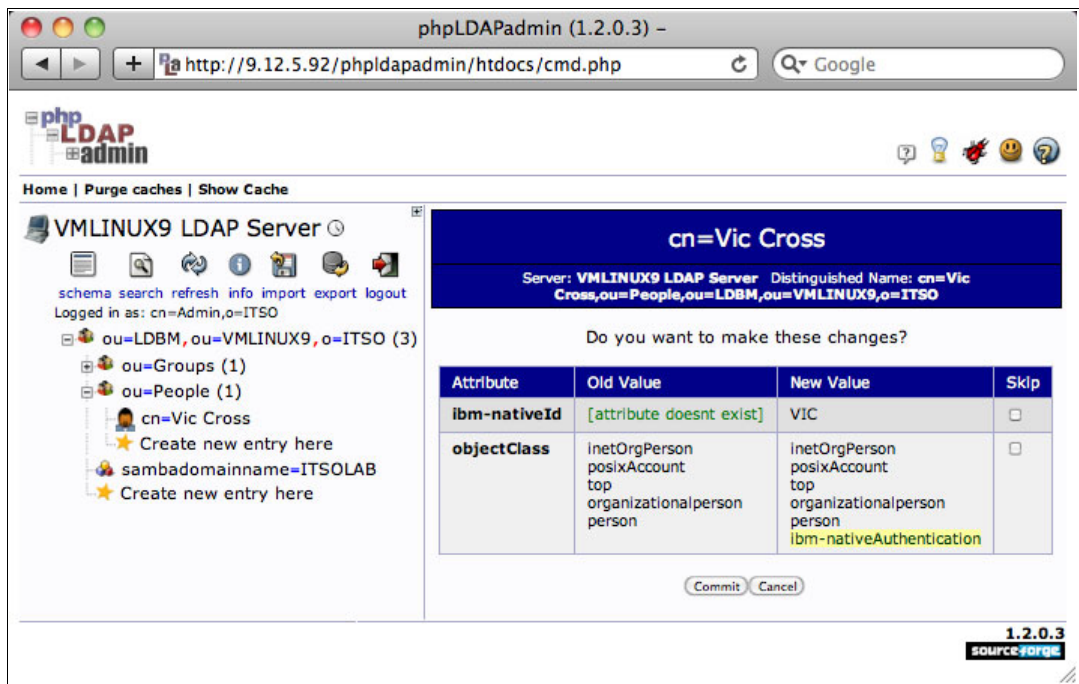


Figure 3-31 Confirmation of the change to be made to our LDAP user object

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 3-32 on page 89.

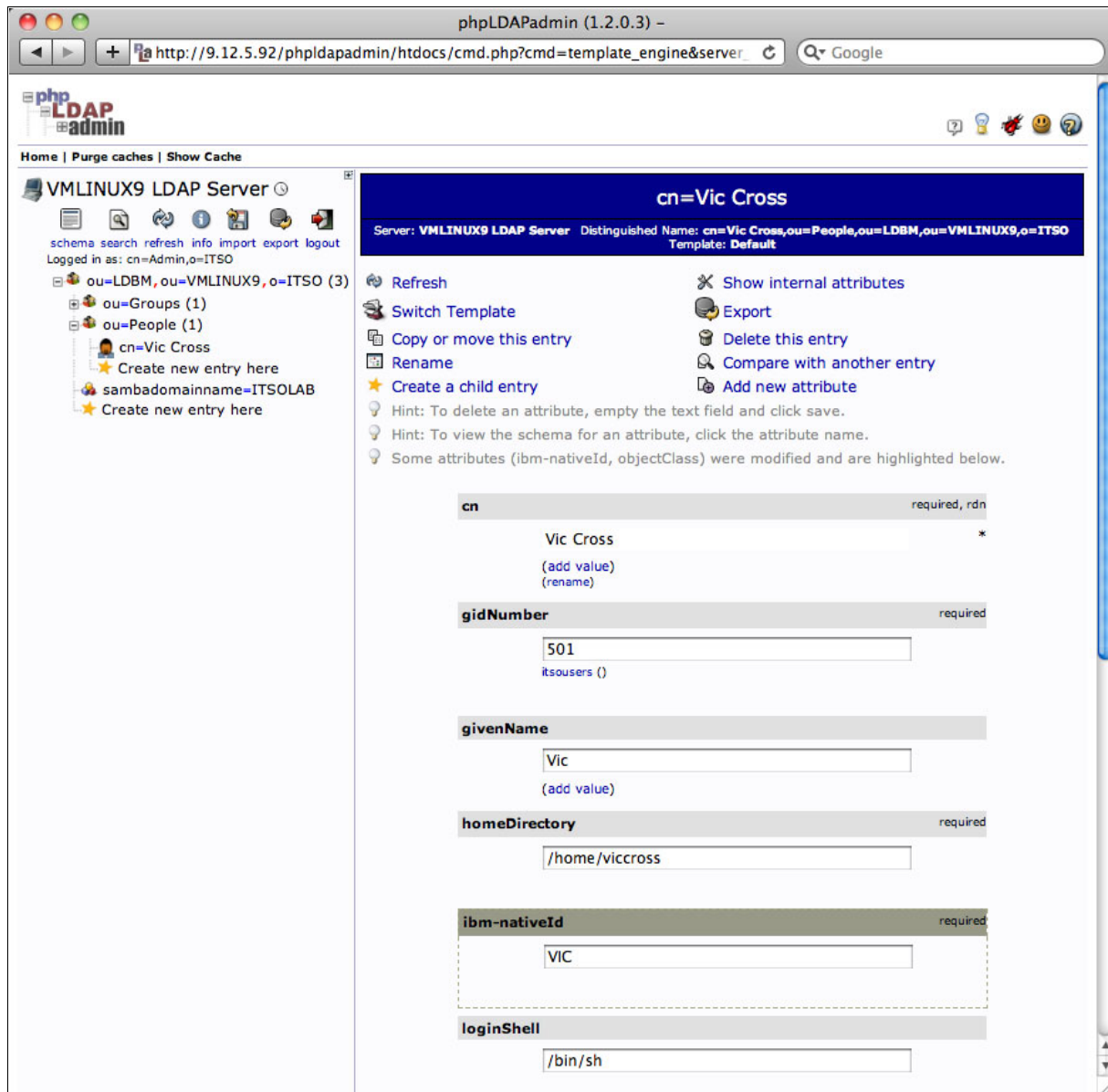


Figure 3-32 The LDAP user object successfully updated

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 3-17 (the LDAP update was performed between the two login attempts).

Example 3-17 Testing login by using LDAP credentials

```
login as: viccross
Using keyboard-interactive authentication.
Password: <RACF password entered>
Access denied
Using keyboard-interactive authentication.
Password: <RACF password entered>
Could not chdir to home directory /home/viccross: No such file or directory
lnxsul:> id
uid=1001(viccross) gid=501(itsousers) groups=501(itsousers)
lnxsul:>
```

3.7 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 3-33 depicts this landscape.

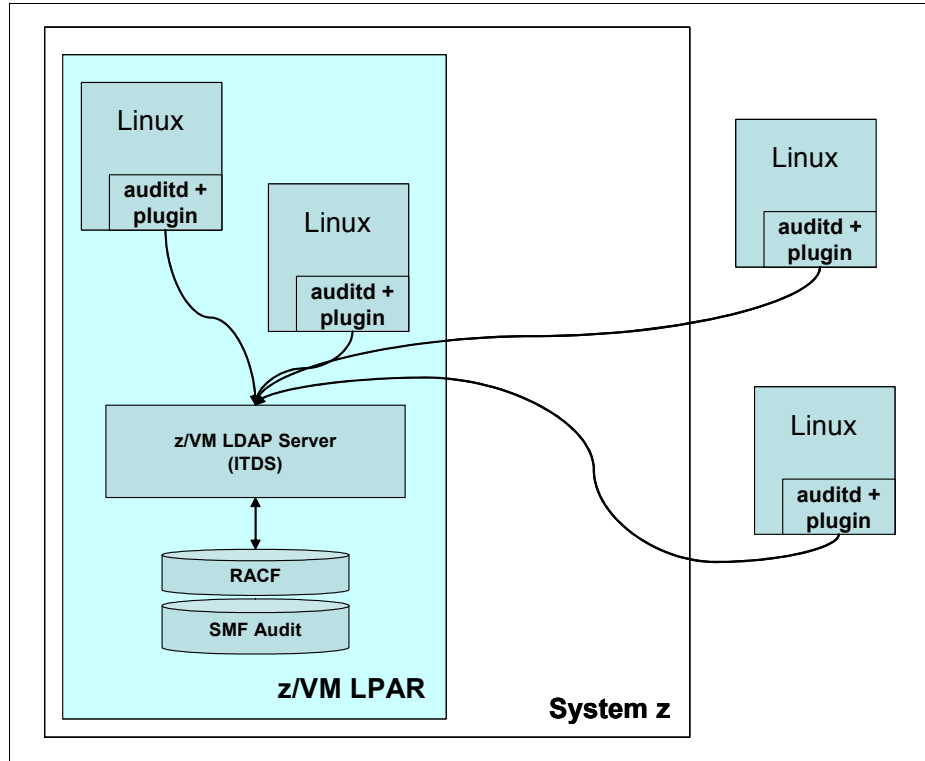


Figure 3-33 Centralized Linux auditing with z/VM LDAP server and RACF

The auditing rules are defined in the `/etc/audit/audit.rules` file. Several examples are provided in `/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:

- ▶ 3.7.1, “Enabling extended operations support in z/VM LDAP server” on page 91
- ▶ 3.7.2, “RACF configuration” on page 91

3.7.1 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 3-18.

Example 3-18 LDAP configuration for remote services

```
#ICTX extended operations support section
plugin clientOperation ITYBIC31 ICTX_INIT "CN=ICTX"
```

3.7.2 RACF configuration

Next, we create a RACF user, shown in Example 3-19, 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 3-19 Creating binduser in RACF

```
rac adduser binduser
rac alu binduser password(s01e1l) 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 3-20 details the commands.

Example 3-20 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 3-21 on page 92.

Example 3-21 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 is 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: ITYX0R03 Password is incorrect, or userID is
undefined/revoked
```

3.7.3 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 3-22.

Example 3-22 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 on 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 3-23 is the assembler code we used to add the @LINUX class to RACF VM, based on the arguments of the z/OS command.

Example 3-23 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 3-24.

Example 3-24 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.

3.7.4 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 3-25. Also make sure the path to the dispatcher is correct.

Example 3-25 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 3-26.

Example 3-26 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 3.7.2, “RACF configuration” on page 91.

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 3-27.

Example 3-27 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
```

3.8 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 back end of the z/VM LDAP server.

We can think of only several 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 to LDIF” on page 56) 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.

Although other reasons might also exist, we believe that the LDBM back end of the z/VM LDAP server offers a high degree of function for supporting Linux applications and is the best 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 requests to the userPassword attribute to a different DN.

Note: The rwm overlay requires the slapo_rwm overlay, which is still considered experimental.

3.8.2 Configuring OpenLDAP

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 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. To add support for the rwm overlay and configure the referral of user passwords to SDBM in the z/VM LDAP server, we created the LDIF file.

We then used the following command to update the LDAP server:

```
1nxsu1:~ # ldapmodify -Y external
```



Authentication and access control

The process of access control is critical. It defines how users and systems can communicate and their approach. Access control controls or limits 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 an established practice to reduce the administration effort in an environment with multiple Linux systems.

To use LDAP for user authentication, you must install and configure `pam_ldap`, `nss_ldap`, and LDAP client, and configure each service that should participate in LDAP authentication (SSH, Telnet, sudo, FTP and so forth). The user information must be accessible from LDAP.

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 is standard Linux security, and it 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.

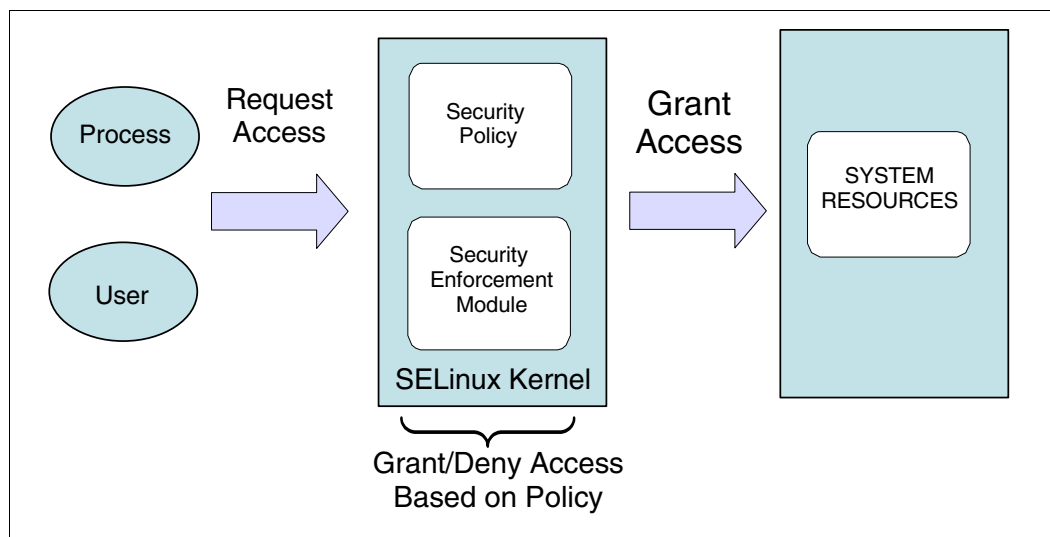


Figure 4-1 Schematic overview of SELinux

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>/rpolity`
This directory will have runtime configuration files and binary policy. 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 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 edge server such as a firewall or a Web server, but the added level of security on a back-end database server might not justify the loss in performance. Alternatively, a back-end 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. 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 5.4

```
[root@lnxrh1 ~]# 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 5.4

```
[root@lnxrh1 ~]# sestatus
SELinux status:                enabled
SELinuxfs mount:              /selinux
Current mode:                  enforcing
Mode from config file:        enforcing
Policy version:                21
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 100.

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. The steps necessary for enabling SELinux in Linux for each RHEL version are provided here. For details of how custom policies are supported, see 4.1.4, “Policies” on page 104.

You must enable SELinux in the kernel so that Linux on System z can take advantage of SELinux features. We briefly examine the steps here. See your Red Hat Linux 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 99.

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 101.

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 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 **getenforce** command, as shown in Example 4-2 on page 99, 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 revision, there is a directory containing the files specific to its version. Select the appropriate directory for the RHEL version 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 revision, 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:

```
[root@lnxrh1 ~] # /usr/sbin/load_policy /etc/selinux/targeted/policy/policy.18
```

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, we 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 5.4

```
[root@lnxrh1 ~]#setenforce 1
[root@lnxrh1 ~]#getenforce
Enforcing
```

To verify and edit the security level that is presently being used on the machine, start the firewall customization panel by issuing the `system-config-securitylevel-tui` command:

```
[root@lnxrh1 ~]# system-config-securitylevel-tui
```

See Figure 4-2 for an example of running this command.

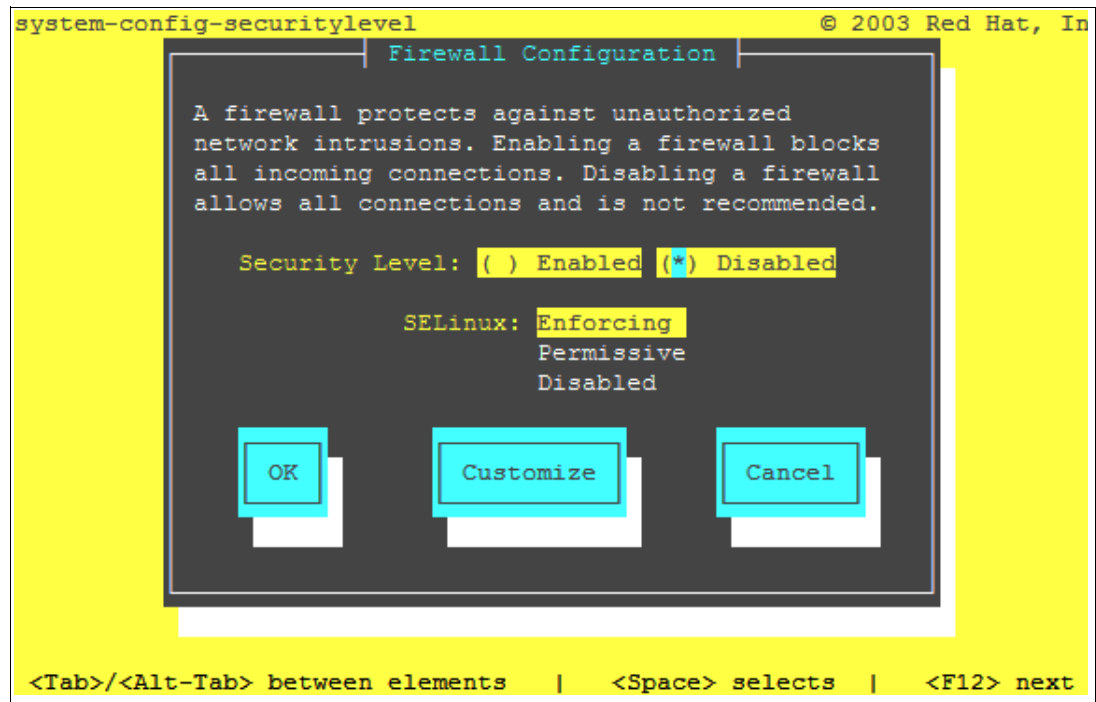


Figure 4-2 System-config-securitylevel-tui in Red Hat Enterprise Linux 5.4

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@lnxrh1 ~]# ftp 0
Connected to 0.
220 (vsFTPd 2.0.5)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (0:root): test1
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/home/test1
Login failed.
```

An SELinux error is logged to `/var/log/messages` file, as shown in Example 4-11. Note that the SELinux policy has prevented the FTP daemon from reading users' home directories.

Example 4-11 Sample of error in /var/log/messages

```
setroubleshoot: SELinux is preventing the ftp daemon from reading users home
directories (./home). For complete SELinux messages. run sealert -l
a670614c-e48b-48b0-b7cc-911ffafaceabd
```

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 enable ftp_home_dir

```
setsebool -P ftp_home_dir=1
```

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 of the system. In these cases, turning off SELinux enables administrators to correct the issue.

Important: This is *not* a recommendation 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 running kernel in the boot loader (`/etc/zipl.conf`) as shown in Example 4-13.

Example 4-13 Sample `/etc/zipl.conf` file with disabled SELinux in Red Hat Enterprise Linux 5.4

```
[defaultboot]
default=linux
target=/boot/
[linux]
    image=/boot/vmlinuz-2.6.18-164.e15
    ramdisk=/boot/initrd-2.6.18-164.e15.img selinux=0
    parameters="root=LABEL=/"
```

Note: Parameter line changes take affect 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*, which is a recursive acronym. To enable SELinux on your system, certain RPMs are required, which are:

- ▶ libselinux-utils
- ▶ libselinux
- ▶ libselinux-python
- ▶ selinux-policy-devel
- ▶ selinux-policy-targeted
- ▶ libselinux-devel
- ▶ selinux-policy

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 2.0 provides easy-to-use Linux security software that protects Linux servers and applications from malicious threats. Novell AppArmor features are:

- ▶ Yet another Setup Tool (YaST)-integrated administration tools for configuration, maintenance and automated development of 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/`

Example 4-14 on page 106 lists the files for AppArmor.

Example 4-14 File list for AppArmor

```
lnxsul:/etc # ls -ltr /etc/apparmor.d
total 68
-rw-r--r-- 1 root root 724 Feb 20 2009 usr.sbin.traceroute
-rw-r--r-- 1 root root 1861 Feb 20 2009 usr.sbin.ntpd
-rw-r--r-- 1 root root 1246 Feb 20 2009 usr.sbin.nscd
-rw-r--r-- 1 root root 853 Feb 20 2009 usr.sbin.mdnssd
-rw-r--r-- 1 root root 842 Feb 20 2009 usr.sbin.identd
-rw-r--r-- 1 root root 696 Feb 20 2009 usr.sbin.avahi-daemon
-rw-r--r-- 1 root root 1087 Feb 20 2009 sbin.syslogd
-rw-r--r-- 1 root root 1130 Feb 20 2009 sbin.syslog-ng
-rw-r--r-- 1 root root 788 Feb 20 2009 sbin.klogd
-rw-r--r-- 1 root root 732 Feb 20 2009 bin.ping
drwxr-xr-x 2 root root 4096 Sep 11 15:46 tunables/
drwxr-xr-x 2 root root 4096 Sep 11 15:46 program-chunks/
drwxr-xr-x 2 root root 4096 Sep 11 15:46 abstractions/
drwxr-xr-x 5 root root 4096 Sep 11 15:46 ./
drwxr-xr-x 83 root root 12288 Sep 21 18:37 ../
lnxsul:/etc # ls -ltr /etc/apparmor
total 52
-rw-r--r-- 1 root root 179 Dec 16 2008 reports.crontab
-rw-r--r-- 1 root root 955 Dec 16 2008 reports.conf
-rw-r--r-- 1 root root 10398 Feb 20 2009 severity.db
-rw-r--r-- 1 root root 4179 Feb 20 2009 logprof.conf
-rw-r--r-- 1 root root 2031 Feb 25 2009 subdomain.conf
drwxr-xr-x 3 root root 4096 Sep 11 15:46 profiles/
drwxr-xr-x 3 root root 4096 Sep 11 15:50 ./
drwxr-xr-x 83 root root 12288 Sep 21 18:37 ../
```

4.2.2 Enable or disable AppArmor by using YaST

The two ways in which AppArmor can be enabled or disabled by using YaST are:

- ▶ Novell AppArmor Control Panel
- ▶ System Services

Enable or disable 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 **Novell AppArmor** → **AppArmor Control Panel**, as shown in Figure 4-3 on page 107.
3. Select **Enable AppArmor**, shown in Figure 4-4 on page 107.
4. Exit the AppArmor Control Panel with Done.

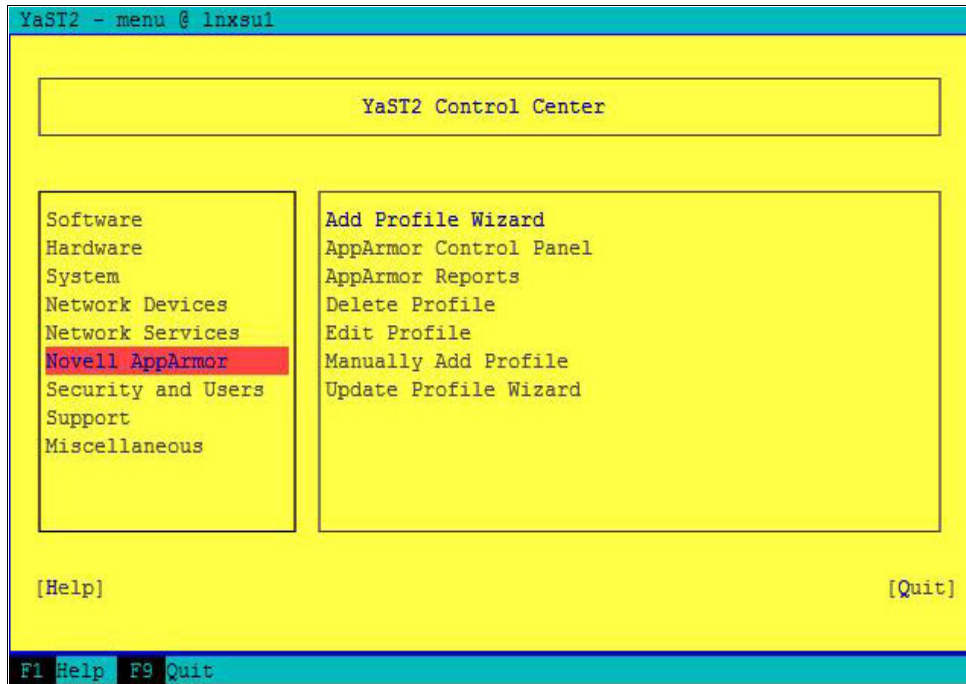


Figure 4-3 Access AppArmor using YaST

Figure 4-4 shows the AppArmor Control Panel set to enable AppArmor.



Figure 4-4 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 4-5):

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 @ inxsul

System Services (Runlevel): Details
( ) Simple Mode  (x) Expert Mode
Set default runlevel after booting to:
5: Full multiuser with network and display manager

Service      Running 0 1 2 3 4 5 6 B S Description
autoyast     Yes
boot.apparmor Yes
boot.cleanup No
boot.clock   No

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/Refresh] [Set/Reset]
[Help] [Cancel] [ OK ]

F1 Help F7 Expert Mode F9 Cancel F10 OK
```

Figure 4-5 Enable AppArmor using system service

Adding a profile using the wizard

The YaST utility for AppArmor includes a wizard that helps you easily add a profile to the server AppArmor configuration. To add a profile using the wizard:

1. Start YaST.
2. Select **Novell AppArmor** → **Add Profile Wizard**.
3. Enter the name of the application, and select **Create**. This runs `aa-autodep`, a Novell AppArmor tool.
4. Select **Scan system log for AppArmor events**. This generate a chain of questions that must answered to guide the wizard in generating the security profile.

The Add Profile Wizard begins 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. In Novell AppArmor, click **Manually Add Profile**.
3. Browse your system to find the application for which to create a profile. When you find the application, select it and click **Open**. A basic, empty profile appears in the Novell AppArmor Profile Dialog window.
4. From the AppArmor Profile Dialog window, you are able to edit Novell AppArmor profiles manually by adding, editing, or deleting entries. Select **Add Entry**.
5. 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 4-15 lists the RPMs required for AppArmor.

Example 4-15 List of RPMs for AppArmor on SLES 11

```
libapparmor
apparmor-admin_en
perl-libapparmor
apparmor-profiles
apparmor-profile-editor
yast2-apparmor
apparmor-docs
apparmor-utils
apparmor-parser
```

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. Success 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 required libraries for this module are:
 - `/lib/security/pam_securetty.so`
 - `/lib/security/pam_pwdb.so shadow nullok`
 - `/lib/security/pam_nologin.so`
- ▶ An *account* module checks to make sure that a user is allowed 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. The required library for this is:
 - `/lib/security/pam_pwdb.so`
- ▶ The *password* module sets the password. Required libraries for this module are:
 - `/lib/security/pam_cracklib.so` (enforces password strength)
 - `/lib/security/pam_pwdb.so nullok use_authtok md5 shadow` (performs authentication token updates)
- ▶ The *session* module manages the user session (such as setting up and terminating log in sessions). The required library is:
 - `/lib/security/pam_pwdb.so`An optional library may be used for this module too:
 - `/lib/security/pam_console.so`

For more information about each of these libraries, see:

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

For Novell SUSE Linux on System z, additional required modules can be found at:

<http://www.novell.com/products/linuxpackages/server10/s390x/pam-modules-32bit.html>

You enable applications to make use of 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 from 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 4-1.

Table 4-1 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 serves to route services through a defined module path. An overview of the organization of PAM on Linux is shown in Figure 4-6.

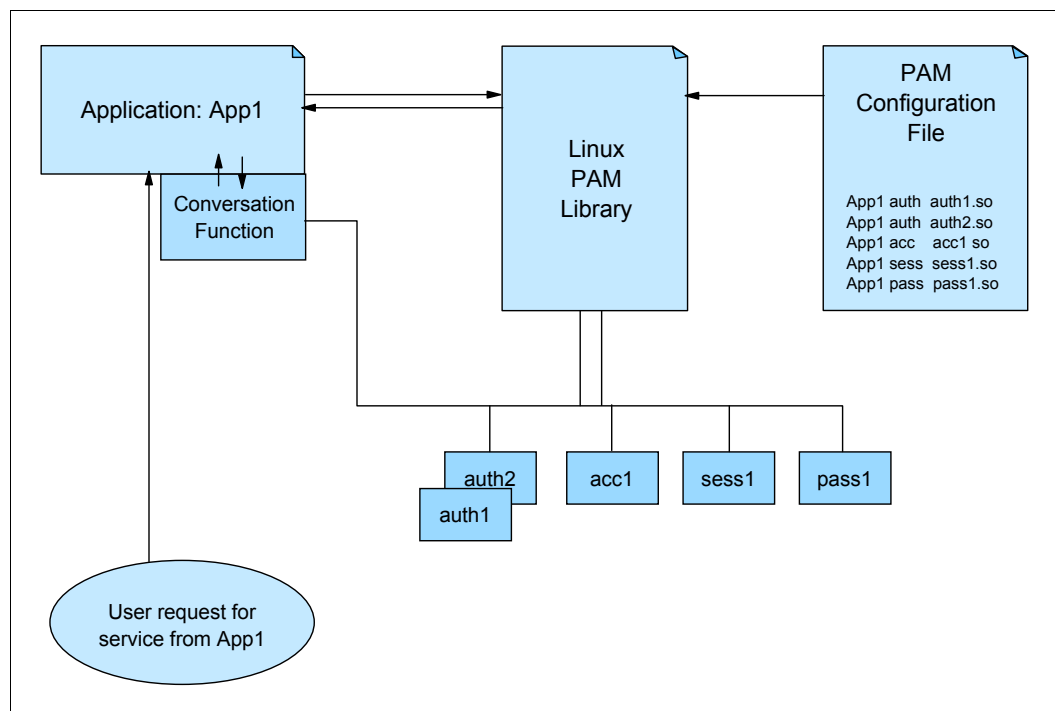


Figure 4-6 Overview of the PAM environment on Linux

The application has no knowledge of the authentication methods in use, and 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 which is built into the application (as part of its PAM support code).

See Example 4-16 for a list of configuration files.

Example 4-16 The PAM configuration files in the /etc/pam.d directory

```
[root@lnxrh1 pam.d]# ls
atd          ekshell    other      runuser-1  sudo-i
authconfig   gssftp     passwd     screen     su-l
authconfig-tui halt       poweroff   setup      system-auth
chfn         kshell     reboot     smtp       system-auth-ac
chsh         ksu        remote     smtp.sendmail system-config-network
config-util  login      rhn_register sshd       system-config-network-cmd
crond        neat       run_init   su
cups         newrole    runuser    sudo
[root@lnxrh1 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 program `login` 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. The file shown in Example 4-17 is an example.

Note: For additional information, refer to PAM documentation at:

<http://www.kernel.org/pub/linux/libs/pam/index.html>

Example 4-17 Sample PAM /etc/pam.d/login login file

```
##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 include    system-auth
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 optional    pam_keyinit.so force revoke
```

Example 4-18 on page 113 is a sample `/etc/pam.d/passwd` file, which is used by the `passwd` program (to update a user password).

Example 4-18 Sample /etc/pam.d/passwd file

```
##PAM-1.0
auth    include    system-auth
account include    system-auth
password include    system-auth
```

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 syntax that allows common definitions to be placed in a single file and included from any other configuration file. The previous PAM configuration file examples show this syntax, by having INCLUDE statements for the system-auth file.

The system-auth configuration file provides common configuration statements for the basic authentication methods used on the system. See Example 4-19.

Example 4-19 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_succeed_if.so uid < 500 quiet
account   required      pam_permit.so

password  requisite     pam_cracklib.so try_first_pass retry=3
password  sufficient    pam_unix.so md5 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 to perform the configuration options. The 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 `/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 4-20 after making changes to the `/etc/pam.d/other` file.

Example 4-20 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 user 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 login for null password:

```
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 4-21.

Example 4-21 Sample file for list of allowed users to do ssh: `/etc/ssh/allow-user`

```
root
user1
user2
```

2. Make sure file is owned by root user and permission is 600 as shown in Example 4-22.

Example 4-22 Sample example for checking file details

```
[root@lnxrh1 ~]# ls -ltr /etc/ssh/allow-user
-rw----- 1 root root 18 Sep 28 19:27 /etc/ssh/allow-user
[root@lnxrh1 ~]#
```

3. Make changes to `/etc/pam.d/ssh` file; add the following line to `auth session`:

```
auth required pam_listfile.so item=user sense=allow file=/etc/ssh/allow-user onerr=fail
```

Example 4-23 shows the `/etc/pam.d/sshd` file after making this change.

Example 4-23 Sample example for sshd file after making changes

```
[root@lnxrh1 ~]# cat /etc/pam.d/sshd
#%PAM-1.0
auth      required      pam_listfile.so item=user sense=allow
file=/etc/ssh/allow-user onerr=fail
account   required      pam_nologin.so
account   include         system-auth
password  include         system-auth
session   optional        pam_keyinit.so force revoke
session   include         system-auth
session   required      pam_loginuid.so
```

4.3.3 PAM and LDAP

In 3.6, “Linux authentication using the z/VM LDAP server” on page 83, 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 more 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 operation is likely.

- ▶ Use automatic logon of a Linux *operator* ID at the console.

With access to the z/VM console controlled as described in 2.5, “Securing console access to z/VM virtual machines” on page 8, 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 5 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.



Crypto hardware

This chapter describes how the System z cryptographic functions can be used to provide offloading of clear key and secure key cryptographic operations for Linux virtual machines.

The base for all cryptographic operations on the System z10 servers is the CP Assist for Cryptographic Function (CPACF). CPACF must be enabled through the no-cost feature code FC3863 for most cryptographic function to be available on the server.

For an in-depth description of the cryptographic functions on the System z10 servers, see:

- ▶ *IBM System z10 Enterprise Class Technical Guide, SG24-7516*
- ▶ *IBM System z10 Business Class Technical Overview, SG24-7632*

5.1 Clear key

This chapter introduces 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 Express2 or Crypto Express3 features in accelerator mode (CEX2A 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 Set up of CPACF, Crypto Express2, and Crypto Express3

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 (PU) 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 helps your applications perform cryptographic operations faster and more secure are the Crypto Express2 (CEX2) and Crypto Express3 (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 user” on page 120.

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.

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's 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 5-1, CPACF is enabled. If not, you must enable it.

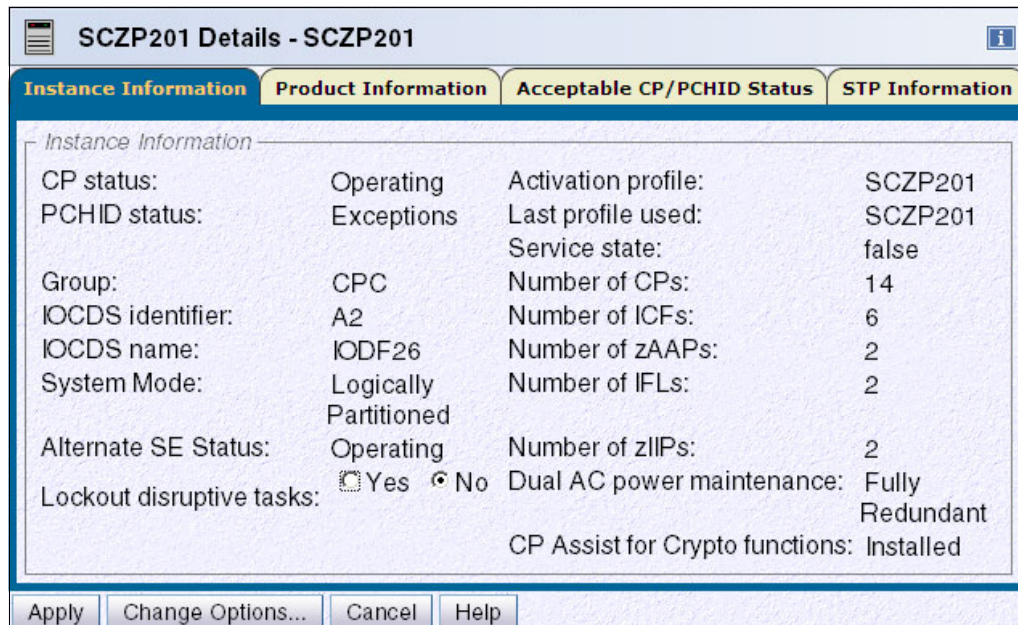


Figure 5-1 CPC details dialog in Support Element interface

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 5-1, but you still have a few algorithms marked with `yes`, you are probably not running on a System z10 machine.

Example 5-1 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 user

Note: To enable the Crypto Express2 or Crypto Express3 feature you have to enable CPACF first. See “Enabling CPACF” on page 119 for more information.

Every Crypto Express2 and Crypto Express3 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 a meaning for Crypto Express adapters in coprocessor mode which is discussed in 5.2, “Secure Key Crypto” on page 157.

You can also find a white paper describing the setup in the additional materials for this Redbooks publication in Appendix C, “Additional material” on page 237.

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 screen 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 5-2 on page 121, 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**.

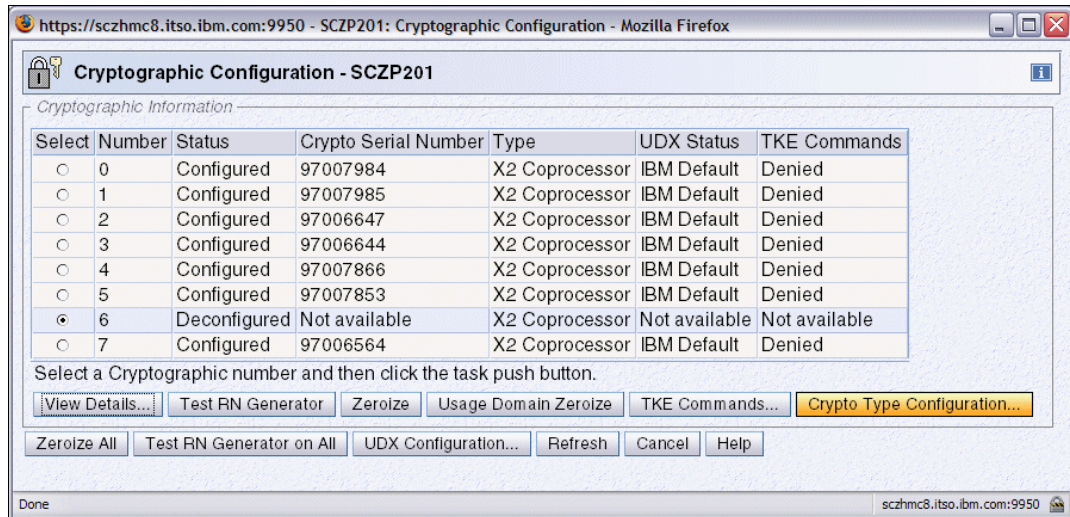


Figure 5-2 List of cryptographic adapters installed in CPC

- In the dialog, as shown in Figure 5-3, you can select the mode of operation that you want to set for the Crypto Express adapter. Choose **Accelerator** and click **OK**.

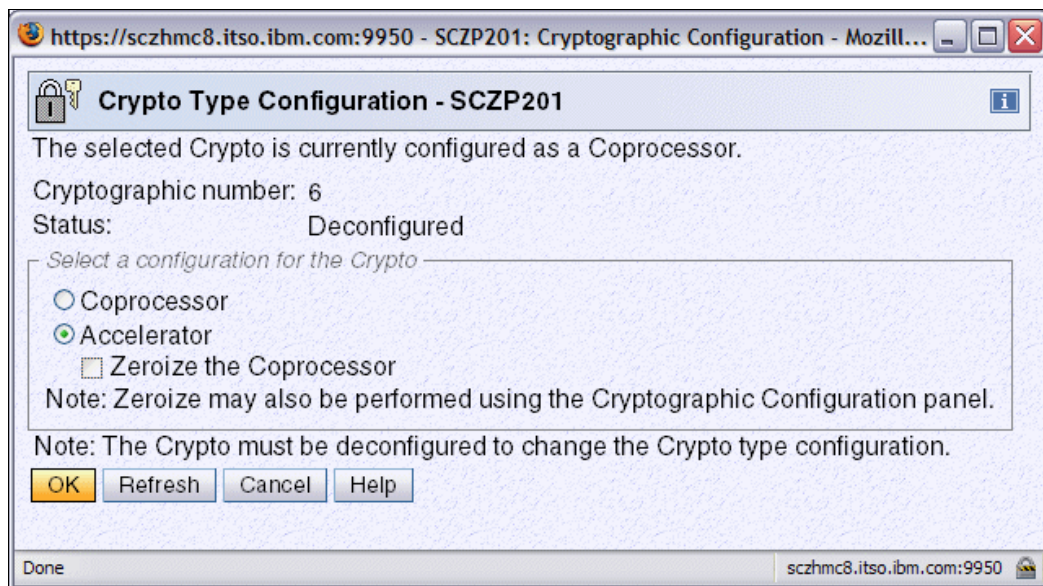


Figure 5-3 Crypto adapter reconfiguration dialog

- The dialog in Figure 5-4 on page 122 opens.

Note: The word *zeroized* in this context means that all information such as keys and certificates will be erased from the adapter. Make sure this is what you want before you click **Yes** in this step.

Confirm your selection by clicking the **Yes** button.

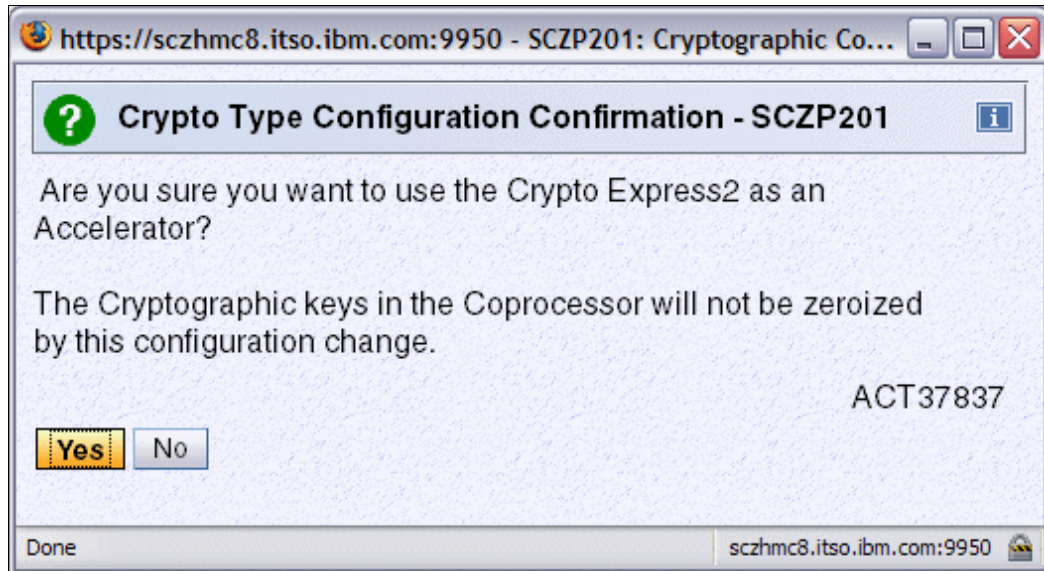


Figure 5-4 Confirmation dialog for zeroizing cryptographic adapter

When the adapter has been reconfigured and is ready for use the Support Element presents you with the message as seen in Figure 5-5.

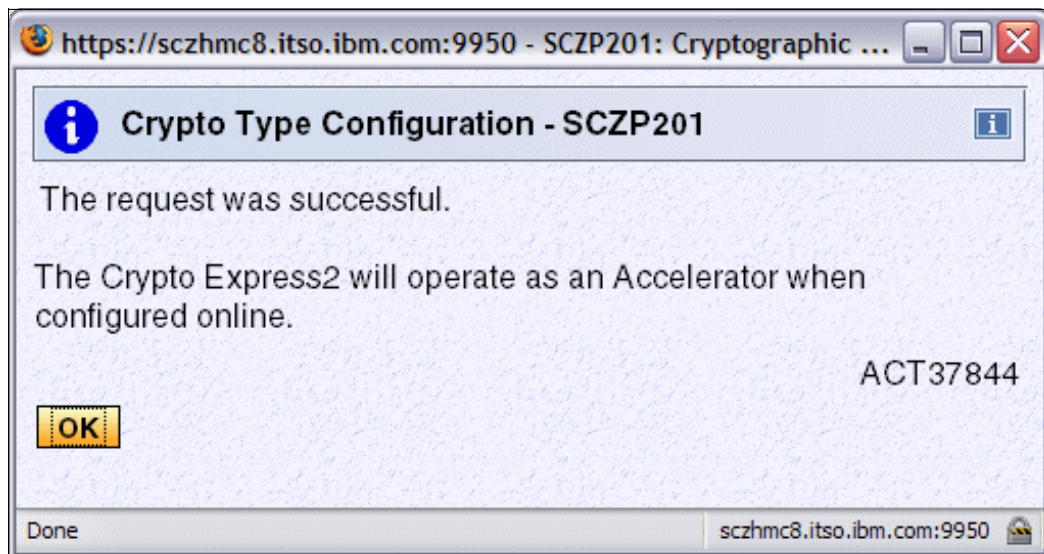


Figure 5-5 Message dialog that confirms that adapter is now in accelerator mode

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 5-6 on page 123, 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 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 120 for more information about setting up the adapters.

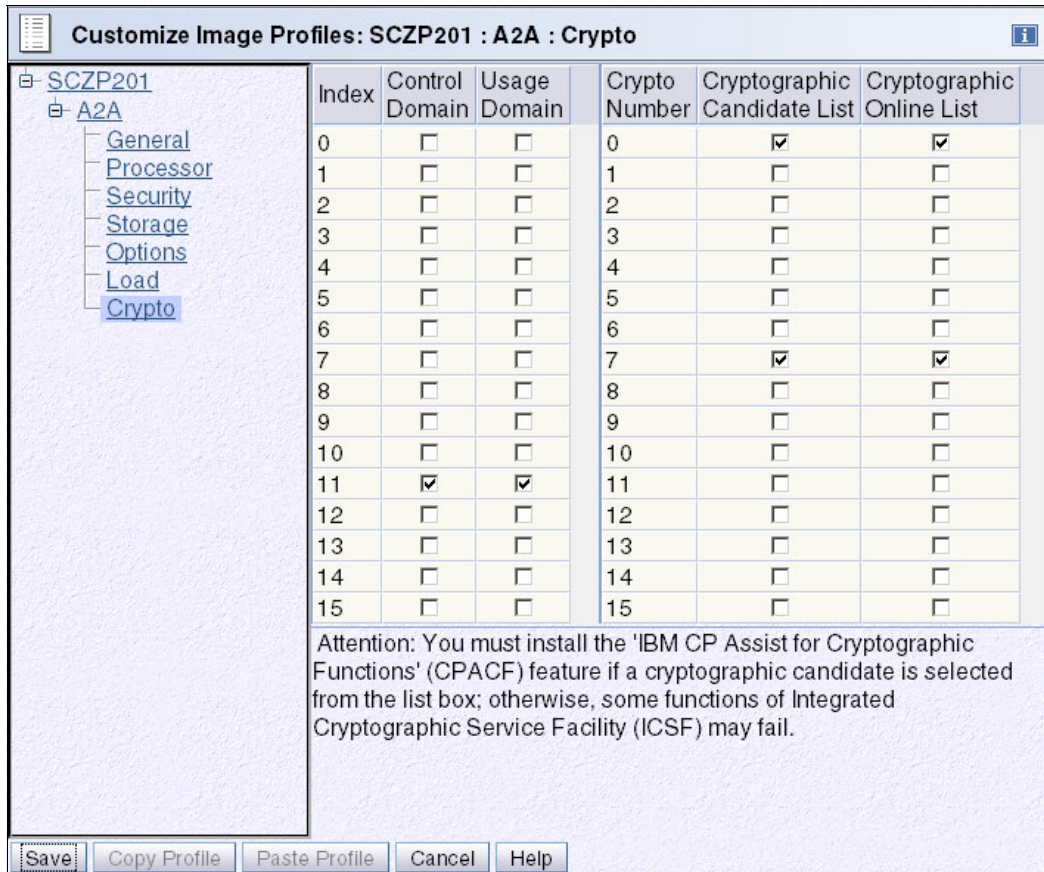


Figure 5-6 Crypto configuration for A2A LPAR

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 5-2 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 5-2 User directory with CRYPTO APVIRT statement

```

USER LNHRH2 ***** 512M 1G G
  INCLUDE IBMDFLT
  CRYPTO APVIRT
  IPL CMS PARM AUTOOCR
  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 CEX2A configuration, issue a QUERY CRYPTO AP statement. If the response looks like the one in Example 5-3, 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 5-3 shows the output of a QUERY CRYPTO AP statement if a CEX2A 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 5-3 QUERY CRYPTO AP output with one CEX2C and one CEX2A 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 5-4. Also, be sure to read the previous *Important* note (on page 124) about changing CRYPTO directory control statements.

Example 5-4 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 `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 5-5.

Example 5-5 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 -a1F /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 5-6, the CRYPTO APVIRT statement is probably not defined in the user directory and Linux has no access to any Crypto Express adapter.

Example 5-6 The modprobe output if no crypto 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 5-7.

Example 5-7 Listing the virtual crypto 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 5-8 shows the contents for a z/VM Linux user that has access to one CEX2A adapter that currently has no outstanding work, and has not yet completed any requests.

Example 5-8 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

```

5.1.2 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 5-9.

Example 5-9 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 5-10 and Example 5-11 on page 127 show modules that are available in a standard Red Hat Enterprise Linux 5.4 and SUSE Linux Enterprise Server 11.

Example 5-10 List of available s390 crypto modules in Red Hat Enterprise Linux 5.4

```

[nico@lnxrh2 ~]$ ls -a1F /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 5-11 on page 127, 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 5-11 List of available s390 crypto modules in SUSE Linux Enterprise Server 11

```
nico@lnxsu2:~> ls -a1F /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.

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 5-12, 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 5-12 Loading the hardware PRNG module

```
[nico@lnxrh2 ~]$ sudo modprobe prng
[nico@lnxrh2 ~]$ ls -a1F /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 5-13 on page 128 provides a sample `udev` rule to make `/dev/prandom` world-readable.

Example 5-13 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 rmmmod 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 5-14 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 5-14 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 `mod_ssl` module for serving content over an encrypted communication channel. Example 5-15 shows the adjustments necessary for `mod_ssl` to access the PRNG device that utilizes CPACF.

Example 5-15 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 5-15 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 5-16 has to be modified or inserted into the `mod_nss` configuration file.

Example 5-16 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 creates 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 as available with SUSE Linux Enterprise Server 11, SC34-2595*.

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 152.

The third option might give rise to side channel attacks by examining the file sizes, timestamps 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 homogenous 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.

¹ <http://www.linuxjournal.com/article/9400>

Figure 5-7 shows how eCryptfs works. It positions itself between the userland applications accessing the VFS and an arbitrary underlying file system. All file system operations from 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.

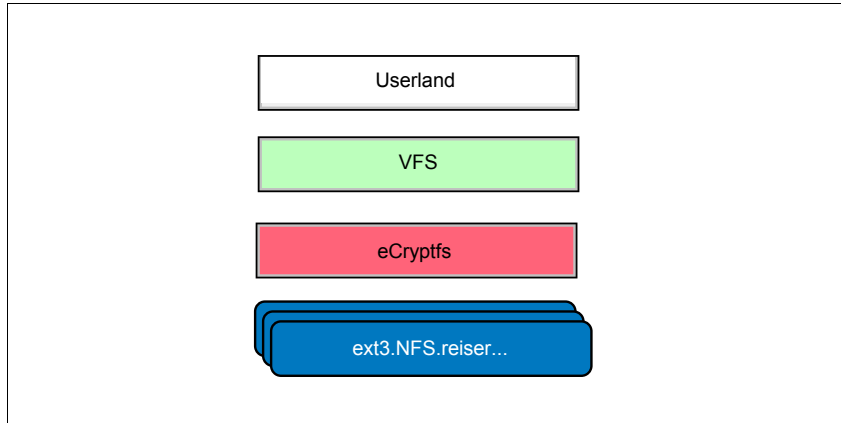


Figure 5-7 Diagram of where eCryptfs is logically positioned in the VFS stack

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 5-17 on page 131 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 choose 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 pass phrase 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_edc128 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.

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 5-17 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 5-18.

Example 5-18 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`N214`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....0J....b._C|
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.

dm-crypt 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 5-8 shows where the dm-crypt encrypted device is logically positioned among real disks, the logical volume manager (LVM) and the file systems.

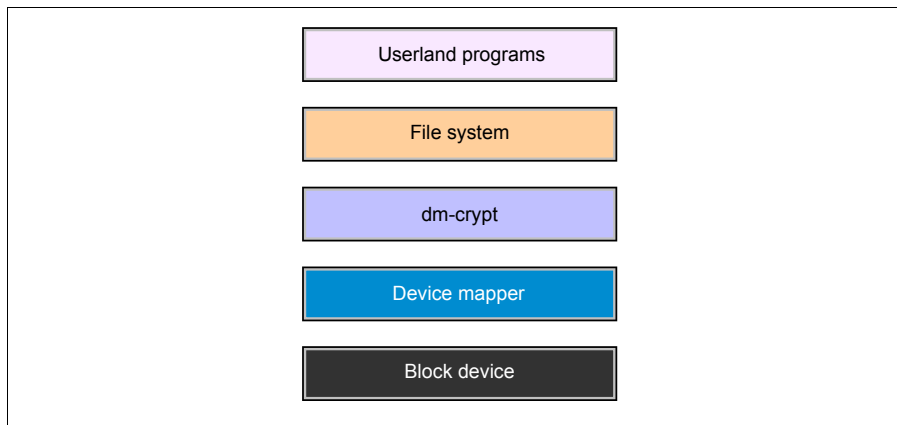


Figure 5-8 Logical position of dm-crypt in the block device and file system hierarchy

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 a 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 5-19 shows you 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 5-19 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 -a1F /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 like 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 5-20 shows the file for the device that was created in Example 5-19.

Example 5-20 The /etc/crypttab file for the device created in Example 5-19

```
#<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 5.1.5, “Statistics and performance” on page 155. 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 129 showed you 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 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 5-9 on page 135 describes the example network setup that will be used. IPsec is configured in transport mode with static keys. This setup will encrypt 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>

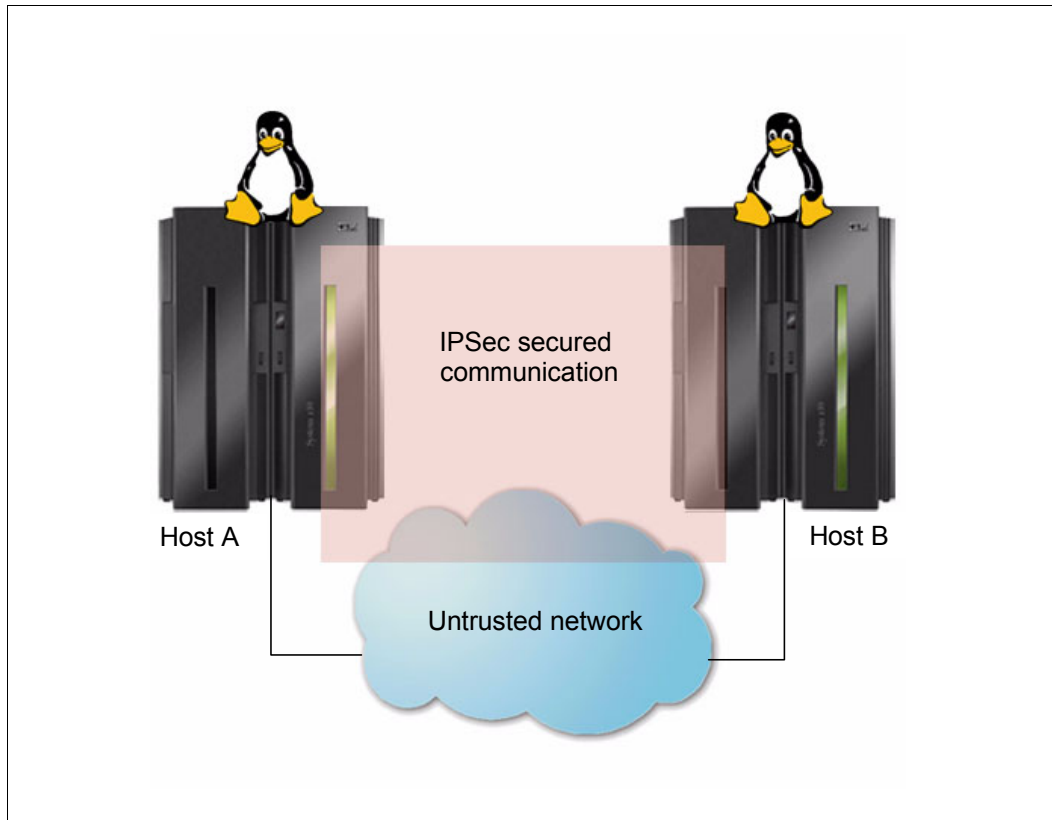


Figure 5-9 Sample network setup for IPsec

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 keys 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 5-21 requires all traffic to be encrypted and also mandates an authentication header to secure the IP headers.

Example 5-21 Configuration file for host A

```
# Flush the security policy database.
spdf flush;
# 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 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 5-22.

Example 5-22 Configuration for host B

```

# Flush the security policy database.
spdf flush;
# 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
0x1234567890123456123456789012345612345678901234561234567890123456;

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
0x1234567890123456123456789012345612345678901234561234567890123456;

# 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 5-23 to `/etc/rc.local` or a similar script that is being called at boot time.

Example 5-23 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 5-24. 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 5-24 Packet trace for a ping with IPSec in transport mode

```
nico@lnxsu2:~> sudo tcpdump -tni 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 5.1.5, “Statistics and performance” on page 155.

5.1.3 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 Express2 and Crypto Express3 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 companies policies.

If you need a full HSM, you can use the Crypto Express adapters in coprocessor mode. See 5.2, “Secure Key Crypto” on page 157 for more information about secure key processing.

PKCS#11

The most widespread standard for access tokens to security modules is PKCS#11 from the RSA Laboratories, and is nicknamed *Cryptoki*. PKCS#11 allows applications to access cryptographic objects on 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 CEX2A 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 a 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 5-25 details the commands that are used to set up the slot and token. Slot #0 is used here to access the clear key cryptographic functions on the CEX2A adapter.

Example 5-25 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 it is not stored inside the Crypto Express2 or Crypto Express3 feature; it is on the local file system in `/var/lib/openssl/openssl/`. Be sure that access to this directory is properly restricted.

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 token's 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 by using the `certutil` utility as described in Example 5-26. 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 5-26 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 performed using the `modutil` tool. Example 5-27 on page 140 shows you the information about the currently defined implementation, which is currently only the default PKCS#11 module that is provided with NSS in Red Hat Enterprise Linux 5.4.

Example 5-27 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 5-28 shows you 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 5-28 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 5-29 outlines the steps involved to create 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 5-29 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=ITS0,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 show in Example 5-30 on page 142.

Example 5-30 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 the **pkcs11-tool** that 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 5-31.

Example 5-31 Packages required for PKCS#11 access on SUSE Linux Enterprise Server 11

```
sudo zypper install engine_pkcs11 pkcs11-helper
```

Make sure the **pkcs11slotd** 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 5-25 on page 138. If everything is configured correctly, you should be able to list the available slots similar to Example 5-32.

Example 5-32 List of uninitialized slots after starting pkcs11slotd for the first time

```
nico@lnxsu2:/var/lib/openssl/openssl> pkcs11-tool --module \
/usr/lib64/openssl/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 request 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 5-33 to make the `pkcs11` engine globally available.

Example 5-33 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/openssl/libcryptoki/libcryptoki.so
default_algorithms = ALL
init = 1
```

If the engine has been configured correctly, you should see output similar to Example 5-34 when you run the `openssl engine -c` command.

Example 5-34 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 by using the `pkcs11-tool` command and generating a certificate request for it. Example 5-35 shows how to use the `pkcs11` OpenSSL engine to access the new private key on the token. The engine uses the `openCryptoki` as the back end which in turn accesses the `ICA` library to manipulate the token.

Example 5-35 Generating a new key on the token using the pkcs11-tool command

```
nico@lnxsu2:~> pkcs11-tool --module /usr/lib64/openssl/libcryptoki/libcryptoki.so \
--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/openssl/libcryptoki/libcryptoki.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=ITS0/0=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 15. 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 5-36. 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 5-36 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 -0
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 (IBM HTTP Server), which is discussed in “IBM HTTP Server” on page 152. 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 signing 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 137 and then start iKeyman. The main window of the utility is shown in Figure 5-10. Because no database has yet been opened, no certificates are displayed.

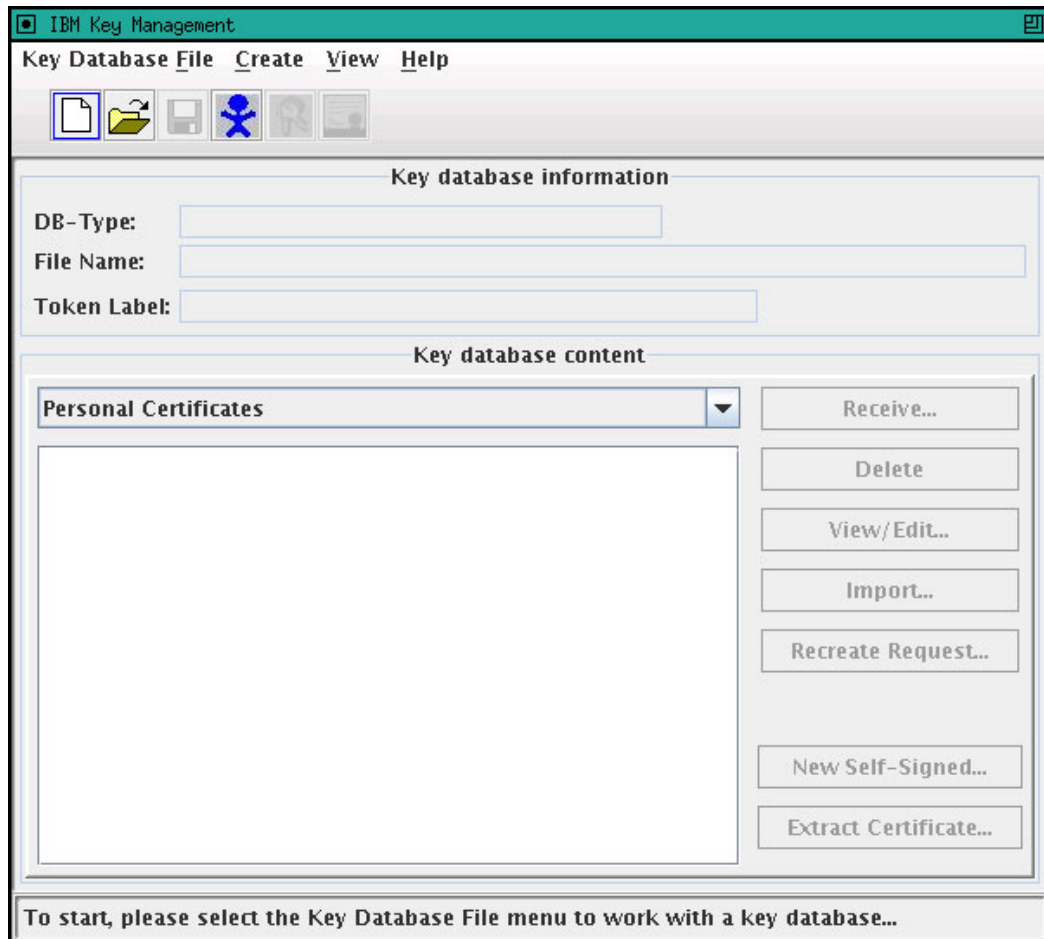


Figure 5-10 The iKeyman key store management utility main window

2. Open a database:
 - a. Either select **Key Database** → **Open** or click **Open a key database file** toolbar item. A new dialog opens, as shown in Figure 5-11 on page 146.
 - 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**.

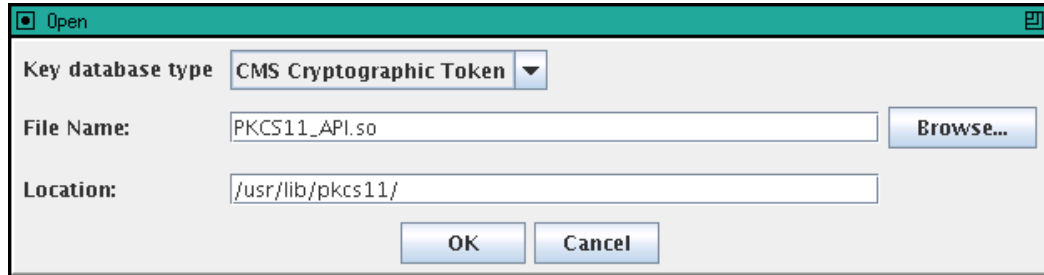


Figure 5-11 Opening a PKCS#11 key store with iKeyman

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 5-12, to work only with the cryptographic token as a key store. Click **OK** to confirm your settings and open the token.

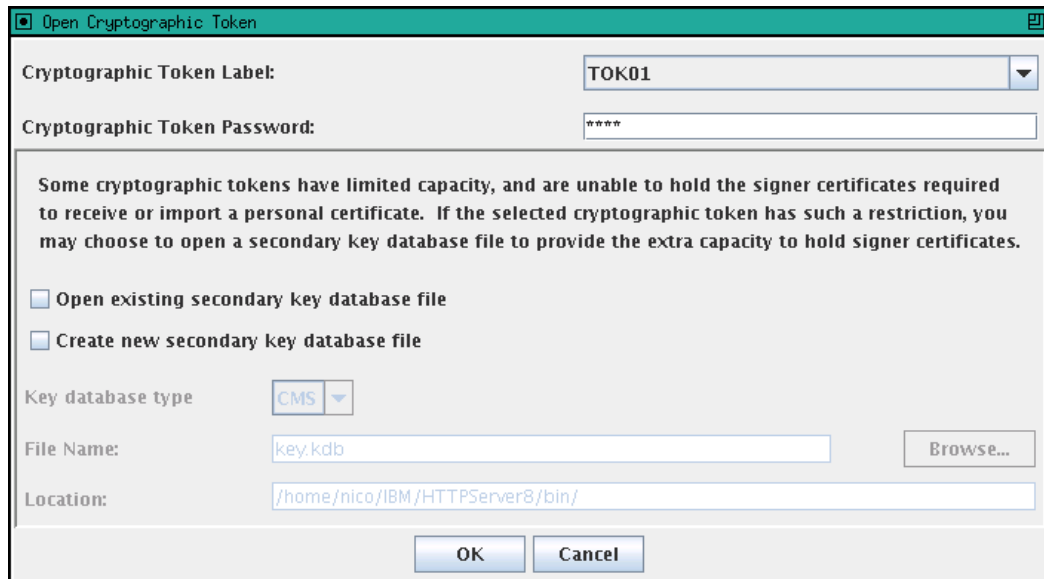


Figure 5-12 Options when opening PKCS#11 key store with iKeyman

4. If you did not create any keys previously, your iKeyman window appears almost exactly like the one in Figure 5-10 on page 145. To create a new key and certificate request in:
 - 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 5-13 on page 147 opens and you can enter properties for the new key and certificate request.

- c. Click **OK** to create the 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 Browse...

OK Reset Cancel

Figure 5-13 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 5-14 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 Browse...

Location: /home/nico/

OK Cancel

Figure 5-14 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 5-15 on page 148. Select your CA certificate name by using the **Browse** button and clicking **OK** to import.

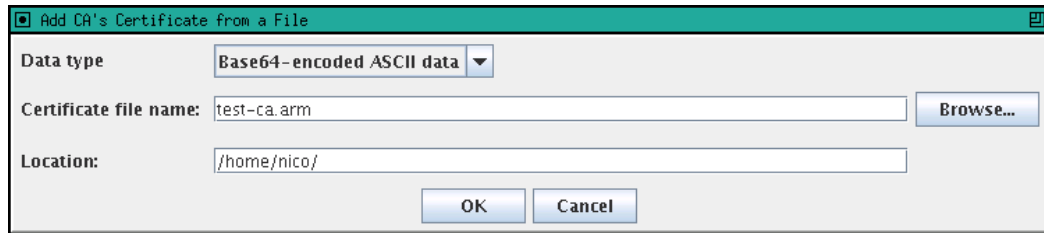


Figure 5-15 Adding CA certificate to key store with iKeyman

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 152 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/>

5.1.4 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 set 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 5-37 on page 149 shows how to query the engine for cryptographic operations that are supported by the *ibmca* library and a Crypto Express2 adapter in accelerator mode.

Example 5-37 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 5-38, where it is used to generate a new RSA key with a key length of 2048 bits.

Example 5-38 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 5-38. We set up a sample certificate authority in Example 5-39 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 5-38.

Example 5-39 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/akey.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 5-40 shows the steps involved. It queries the cryptographic drivers statistics to verify that CEX2A adapter is indeed being used in the SSL connection.

Example 5-40 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 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 5-41 and restart the Web server for the changes to take effect.

Example 5-41 Statement to set the mod_ssl crypto engine to use

```
SSLCryptoDevice ibmca
```

You can get the number of currently open handles to your Crypto Express adapter by reading the `/proc/driver/z90crypt` `procfs` entry. Enabling the ICA engine in the Web server should increase the number. See Example 5-42.

Example 5-42 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 5-43.

Example 5-43 Statistics for total number of handled requests for 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 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 the **Reload** button in your Web browser might not work because the old SSL connection might be reused. As shown in Example 5-44 on page 152, 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 5-44 Using the s_client command of OpenSSL to make a 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 144.

Every access to the cryptographic token requires you to authenticate yourself with a PIN. To have you Web server start automatically, you have to store the PIN in a *stash* by using the **sslstash** utility. Example 5-45 shows how to create a stash for PIN 1111 that will be used to access a cryptographic function.

Example 5-45 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 5-46 shows a sample configuration to make IBM HTTP Server use following items:

- ▶ The key named key01 in the token TOK01
- ▶ The openCryptoki driver for PKCS#11 operations
- ▶ The stash file that we created earlier

You have to restart the Web server for the changes to take effect.

Example 5-46 Sample configuration for IBM HTTP Server using 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 5-42 on page 151.

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 144 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 5-47 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 5-47.

Example 5-47 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 internal list of providers
            Security.addProvider(provider);
        }
        catch(Exception e){
            throw(new RuntimeException(e));
        }
    }

    public static void main(String[] args) throws Exception {
        // Get a new random 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(0xFF1 & buffer[1023]);
    }
}
```

Compiling and running the PKCS11Test class is performed in Example 5-48. 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 5-48 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 steps makes migrating your applications as easy as changing the key store name and key label. Example 5-49 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 5-49 PKCS11Test1.java - Signing data with private key from 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 5-50 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.

5.1.5 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 Express2. 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 5.1.1, “Set up of CPACF, Crypto Express2, and Crypto Express3” on page 118 for the list of algorithms that are supported by CPACF. As of writing, there are no performance measurements for the Crypto Express3 adapters that are running with Linux on System z.

Figure 5-16 on page 156 shows the CPU cost savings that can be achieved when using the ICA OpenSSL engine, which offloads certain ciphers to the CPACF (see “OpenSSL” on page 148 for more information about using the ICA engine). Especially interesting is the almost twelve-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).

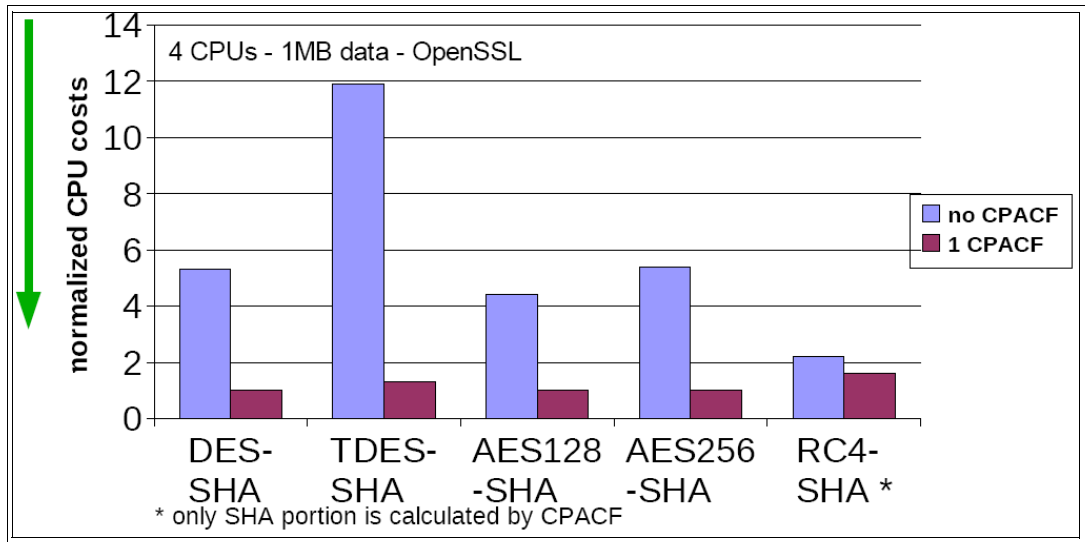


Figure 5-16 Performance measurements for CPACF

As described in “Apache httpd and mod_ssl” on page 151, the Crypto Express adapters in accelerator mode can be used to speed up SSL handshakes of Web servers. Figure 5-17 is a performance comparison between a Web server that is using standard SSL and using CEX2 support. Obviously, offloading the encryption involved in the handshakes provides huge savings in CPU time and the number of connections that can be processed per second.

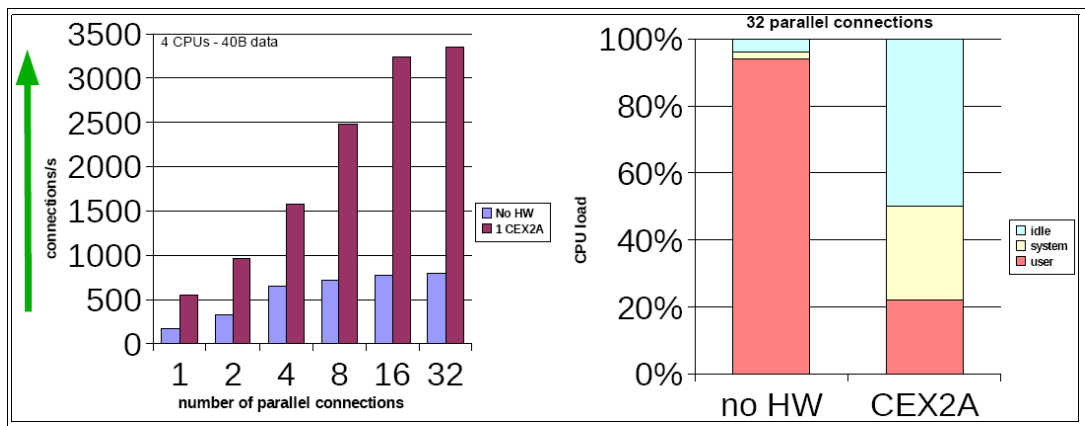


Figure 5-17 Performance measurements for CEX2A

The final step in improving your Web servers performance is to combine CPACF with CEX2A. Figure 5-18 on page 157 shows that you can practically double your throughput while reducing your CPU costs to up to a sixth when compared to running without hardware cryptography support.

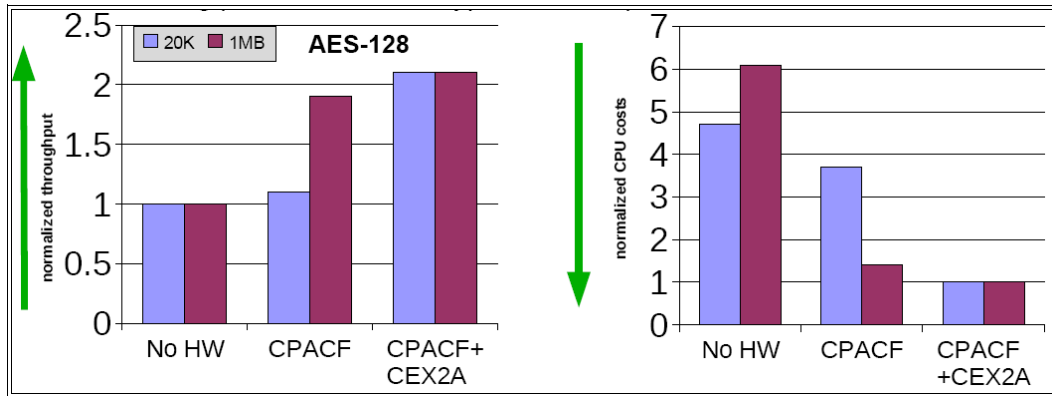


Figure 5-18 Performance comparison full System z hardware supported cryptography

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.

5.2 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 adapters². 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.

5.2.1 Comparing secure key and clear 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 if 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

² 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.

5.2.2 Secure key functions

This section discusses 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 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 5-49 on page 154.

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 by being non-repudiable 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 that 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.



Physical and infrastructure security on System z

This chapter provides 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 physical locations where the systems are installed and from which the system 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 are 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 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 recommendation 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.

6.3 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 strongly protected.

6.4 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 applications systems evolve, we will see additional zones appear for infrastructure, applications and data. See Figure 6-1 for an example.

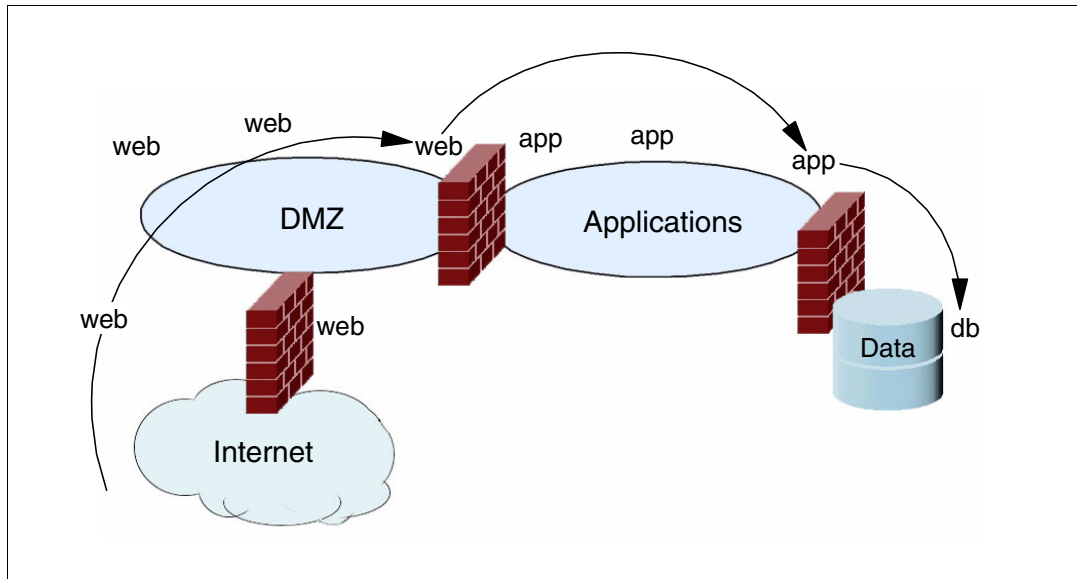


Figure 6-1 A multizone network

6.4.1 The multizone concept

In a multitier, or multizone, configuration of individual server images, or groups of server images, 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 watch-dogs, 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 customers have implemented such a configuration. If you want more information about how a firewall can be implemented in Linux on System z see 6.6, "Linux firewalls" on page 165

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 those reasons, among others, some customers 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.

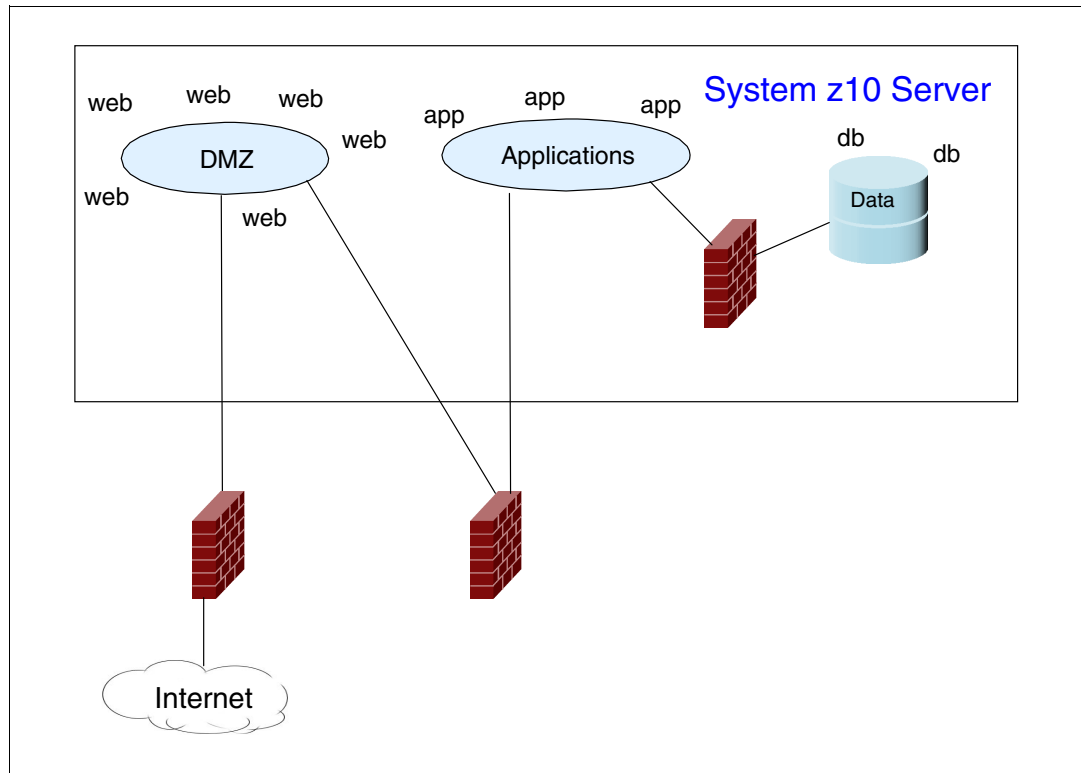


Figure 6-2 System z based multizone network

6.4.2 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 7, “Best practices” on page 197 for details about securing your zoned environment.

6.4.3 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 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.

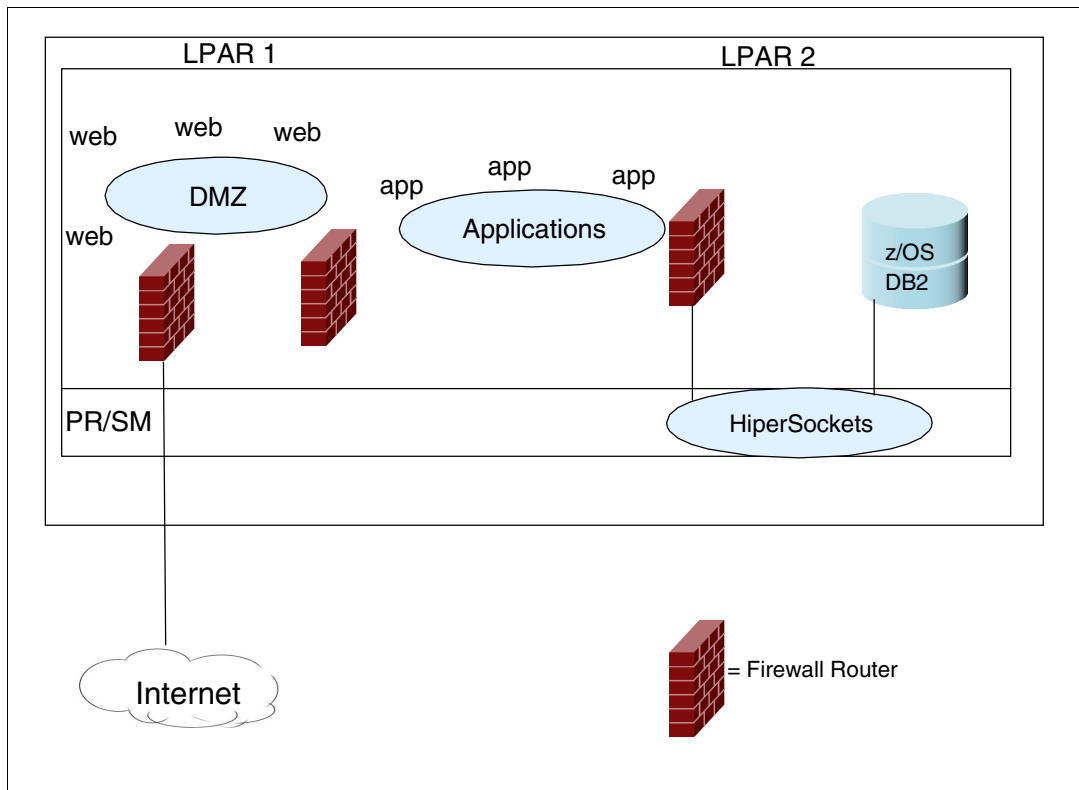


Figure 6-3 Using HiperSockets in your System z network

Numerous other possibilities exist for protecting the HiperSockets connections. Figure 6-4 on page 164 shows an example where the HiperSockets connection is protected on the z/OS side by using z/OS packet filters.

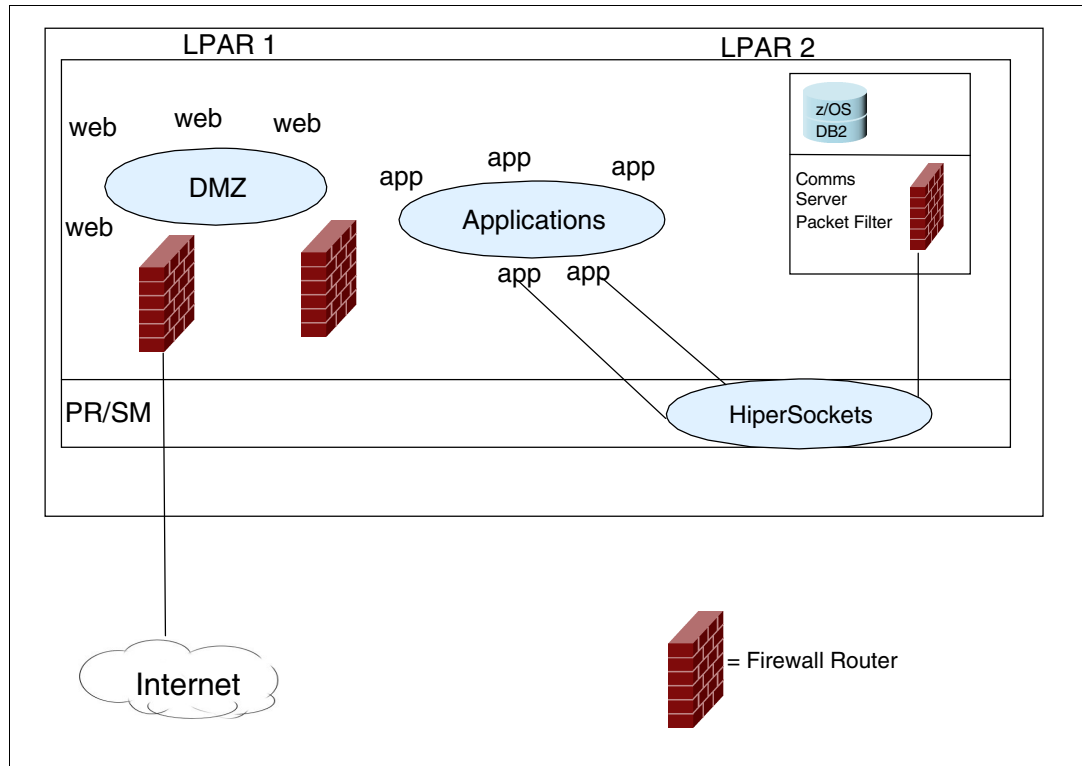


Figure 6-4 Using HiperSockets and z/OS packet filtering

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 2.7.5, “Centralized audit” on page 29.

6.5 IBM Proventia products

IBM Proventia® Server for Linux Intrusion Prevention System (IPS) is a software product that helps you meet compliance regulations. IPS protects your servers against malicious threats that could compromise sensitive data. In addition, IPS helps you simplify compliance requirements, as follows:

- ▶ Maintains compliance with tracking and reporting tools.
IPS receives alerts for activities regarding privilege escalations and unauthorized configuration changes
- ▶ Protects servers against known and unknown attacks without the requirement for patches.
IPS offers vulnerability-centric intrusion prevention to block network worms. Buffer overflow prevention blocks those types of attacks.
- ▶ Enforces network access policies.
Sets IP and port restrictions for inbound and outbound traffic such so that the attack surface is reduced.

- ▶ Administers application-level intrusion detection and auditing.
Detects and responds to attacks and unauthorized activities, maintaining system integrity and compliance. It is implemented through log monitoring of the operating system and the application activity.
- ▶ Is optimized for lean resource consumption.
It is designed for minimal processor utilization and memory footprint.
- ▶ Offers Web application protection.
Applications are protected by inspecting SSL encrypted traffic for malicious activity.
- ▶ Integrates with Active Directory
Uses Active Directory structures to manage policies, monitor events, and create reports.

6.5.1 Preemptive protection

Some host protection solutions provide intrusion prevention signatures in addition to a local firewall. Much like anti-virus signatures, signature-based intrusion prevention works well against only known threats, but is useless against unknown exploits. Preemptive protection, based on a combination of vulnerability-centric intrusion prevention coupled with other protection technologies can keep valuable servers ahead of the threat.

6.5.2 Buffer overflow exploit prevention

Buffer overflow exploit prevention (BOEP) is a signatureless technology that actively looks for malicious code exploits that are in memory buffer overflows. BOEP technology is vulnerability-independent so that it can offer protection even before knowing that particular vulnerabilities exist. The technology stops worms from propagating and prevents attackers from using buffer overflows to run arbitrary code on your systems. BOEP technology hooks system calls (not simply the API level) and prevents such exploits without noisy pop-ups.

6.5.3 Firewall

The firewall capabilities of Proventia Server IPS for Linux reduce the attack surface by blocking unauthorized access to ports and IP addresses, preventing IP spoofing and terminal hijacking.

For more information about the Proventia product, see:

http://www-935.ibm.com/services/us/iss/pdf/proventia_server_ips_for_linux_datasheet.pdf

6.6 Linux firewalls

Being a multiple purpose kernel, Linux includes a multitude of network functionality among which include 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 you how to set up your firewall manually by using a command-line utility and with the help of tools.

6.6.1 Iptables

Iptables is an administration tool that controls the IPv4 packet filter rules and network address translation (NAT) in the Linux kernel. Iptable ensures that your systems are protected from external sources, such as malicious 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.

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 and some of them are for carrying protocols, which contain no data, and is 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.

With iptables, you can check the iptables status. See Example 6-1.

Example 6-1 Check status of iptables

```
[root@lnxrh1 ~]# service iptables status
Table: filter
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
[root@lnxrh1 ~]#
```

Iptables can be stopped, started, or restarted after booting from a command line. See Example 6-2.

Example 6-2 Start, stop, and restart iptables

```
[root@lnxrh1 ~]# service iptables start
Flushing firewall rules: [ OK ]
Setting chains to policy ACCEPT: filter [ OK ]
Unloading iptables modules: [ OK ]
[root@lnxrh1 ~]# service iptables stop
```

To view a list of all the rules assigned in iptables, see Example 6-3 on page 167. A newly installed Linux image does not have any of the iptable rules.

Example 6-3 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 Iptable

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 tables for iptables enable MAC networking rules to be managed separately from 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	Filters packets to the firewall.
OUTPUT	Originates the filtered packets from the firewall.
FORWARD	Receives the packets that are being routed through the system.

This section discusses the most commonly used chain options. To create a new chain with the name chainlist, see Example 6-4.

Example 6-4 Create a chain

```
[root@lnxrh1 ~]# iptables -N chainlist
[root@lnxrh1 ~]#
```

To delete a chain with the name chainlist, see Example 6-5.

Example 6-5 Delete a chain

```
[root@lnxrh1 ~]# iptables -X chainlist
[root@lnxrh1 ~]#
```

To add rules to the chains, see Example 6-6.

Example 6-6 Add a rule to a chain

```
[root@lnxrh1 ~]# iptables -A chainlist -s 9.12.5.92/24 -j ACCEPT
[root@lnxrh1 ~]#
```

To modify a chain to drop any packet that matches a rule, see Example 6-7.

Example 6-7 DROP packets

```
[root@lnxrh1 ~]# iptables -A chainlist -s 9.12.5.90/24 -j DROP
[root@lnxrh1 ~]#
```

To flush the rules out of the chain, see Example 6-8.

Example 6-8 Flush the chain

```
[root@lnxrh1 ~]# iptables -F chainlist
[root@lnxrh1 ~]#
```

To reset the packet and byte counter for chainlist, see Example 6-9:

Example 6-9 Example to reset packets

```
[root@lnxrh1 ~]# iptables -Z chainlist
[root@lnxrh1 ~]#
```

Rules

Rules are patterns that iptables uses to determine whether 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 replace a rule in the INPUT chain in the filter table, use the following command:

```
[root@lnxrh1 ~]# iptables -R INPUT rule_number rule
```

To append a rule in the INPUT chain in the NAT table, use the following command:

```
[root@lnxrh1 ~]# iptables -t nat -A INPUT rule
```

Example 6-10 shows how packets can be accepted (policy ACCEPT). The numbered sections of the example are explained after the example.

Example 6-10 ACCEPT packets

```
[root@lnxrh1 ~]# iptables -A INPUT -p tcp --dport ssh -j ACCEPT 1
[root@lnxrh1 ~]#
```

```
[root@lnxrh1 ~]# iptables -L 2
Chain INPUT (policy ACCEPT)
target     prot opt source                               destination
ACCEPT    tcp  --  anywhere                               anywhere
Chain FORWARD (policy ACCEPT)
target     prot opt source                               destination
tcp dpt:ssh
```

```
target    prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
Chain chainlist (0 references)
target    prot opt source                destination
```

```
[root@lnxrh1 ~]# iptables -A INPUT -p tcp --dport 8080 -j ACCEPT 3
```

```
[root@lnxrh1 ~]# iptables -L 4
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.** This command allows SSH to connect to external addresses.
- 2.** This command shows services that are blocked or allowed.
- 3.** This command enables alternative Web servers or services to connect to external addresses.
- 4.** This command lists the iptables again to show that both SSH and the webcache (8080 port) are now able to connect to external addresses.

Example 6-11 has a listing of the iptables and shows that only SSH connections and alternative Web services are allowed.

Example 6-11 Listing of iptable

```
[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
```

6.6.2 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 is the same as on any other Linux platform, the same tools can be used to manage the firewall.

This section presents two popular firewall tools and how to set them up on Linux on System z. Configuring actual firewalls are not covered because this topic is vast and is very dependent on the actual network topology.

Firewall Builder

The first tool we discuss is `fwbuilder` (Firewall Builder), which includes a graphical user interface (GUI).

Currently, no binary package for `fwbuilder` is available in either SUSE Linux Enterprise Server 11 or Red Hat Enterprise Linux 5.4, so we have to build one from source files. The `fwbuilder` developers provide a source RPM package that helps to more easily build a binary package by using the standard build tools on both SUSE and Red Hat.

For this example, we are using a SUSE system. Make sure you have the SUSE Linux Enterprise Server 11 SDK available because you will need it to install developer packages, which 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 6-12 details the commands that you issue to build the `fwbuilder` binary package for the s390x architecture. If you have any errors in the `rpmbuild` steps you might be missing required libraries. Check the errors and if necessary, install the indicated packages from the SUSE Linux Enterprise Server 11 SDK.

Example 6-12 Building fwbuilder from source on SUSE Linux Enterprise Server 11

```
nico@lnxsu2:~> sudo rpm -i ~/libfwbuilder-3.0.7-b1477.e15.src.rpm
nico@lnxsu2:~> cd /usr/src/packages/
nico@lnxsu2:/usr/src/packages> sudo rpmbuild -bb SPECS/libfwbuilder-3.0.7.spec
[ ... ]
nico@lnxsu2:/usr/src/packages> sudo rpm -i \
/usr/src/packages/RPMS/s390x/libfwbuilder-3.0.7-b1477.s390x.rpm \
/usr/src/packages/RPMS/s390x/libfwbuilder-devel-3.0.7-b1477.s390x.rpm
nico@lnxsu2:/usr/src/packages> sudo rpmbuild -bb SPECS/fwbuilder-3.0.7.spec
[ ... ]
nico@lnxsu2:/usr/src/packages> find . -iname '*fwbuilder*.rpm'
./RPMS/s390x/libfwbuilder-3.0.7-b1477.s390x.rpm
./RPMS/s390x/fwbuilder-3.0.7-b1477.s390x.rpm
./RPMS/s390x/libfwbuilder-devel-3.0.7-b1477.s390x.rpm
nico@lnxsu2:/usr/src/packages> sudo rpm -i \
/usr/src/packages/RPMS/s390x/fwbuilder-3.0.7-b1477.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; 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.

Start the VNC server with a command line as in Example 6-13. If this is the first time you are starting the VNC server, it asks you to enter a password to protect your VNC session. It will be stored in `~/vnc/passwd` and removing this file will reset it.

Example 6-13 Sample VNC server command line

```
nico@lnxsu2:~> vncserver -geometry 1024x768 -localhost -depth 16 :3 -localhost
You will require a password to access your desktops.
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
New 'X' desktop is lnxsu2:3
Starting applications specified in /home/nico/.vnc/xstartup
Log file is /home/nico/.vnc/lnxsu2:3.log
```

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 recommend 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 recommend using the *sudo* mechanism to provide appropriate access control and logging.

If you used the command line from Example 6-13 to start the VNC server, the VNC server will listen on port 5903 on your Linux host. If your firewall settings permit it and you can trust the network to your server, you can connect to the server directly. Otherwise you can tunnel a connection through your SSH client (if you choose this option, see your SSH client's manual for information about how to set up tunnels).

Figure 6-5 on page 172 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 pre-configured 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.

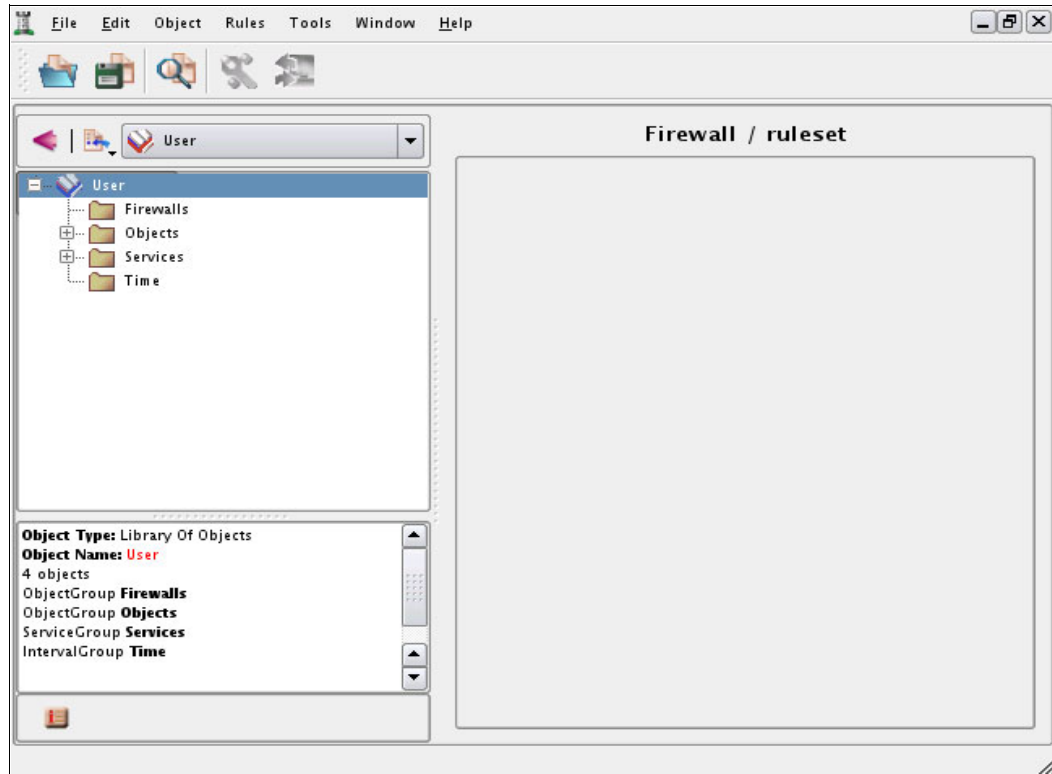


Figure 6-5 Initial view of fwbuilder GUI

Create a new firewall by right-clicking **Firewalls** in the tree and selecting **New Firewall**. A dialog opens, as shown in Figure 6-6. For this short introduction, select **Use preconfigured template firewall objects** and set the operating system and firewall software. Then, click **Next** to move to the next window.



Figure 6-6 The create new firewall dialog in fwbuilder

The dialog shown in Figure 6-7 on page 173 opens where 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 you Linux guest, whether it is an Ethernet or IP only device. Click **Finish** to select and load the template.

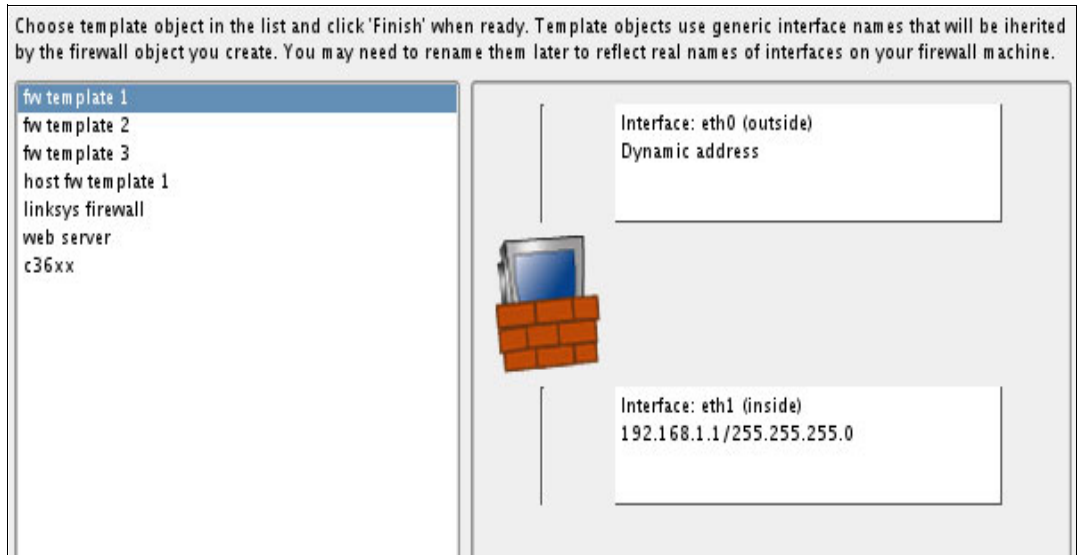


Figure 6-7 Template selection in fwbuilder GUI

Starting from the template entries, you may customize your firewall rules to your needs. Figure 6-8 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.

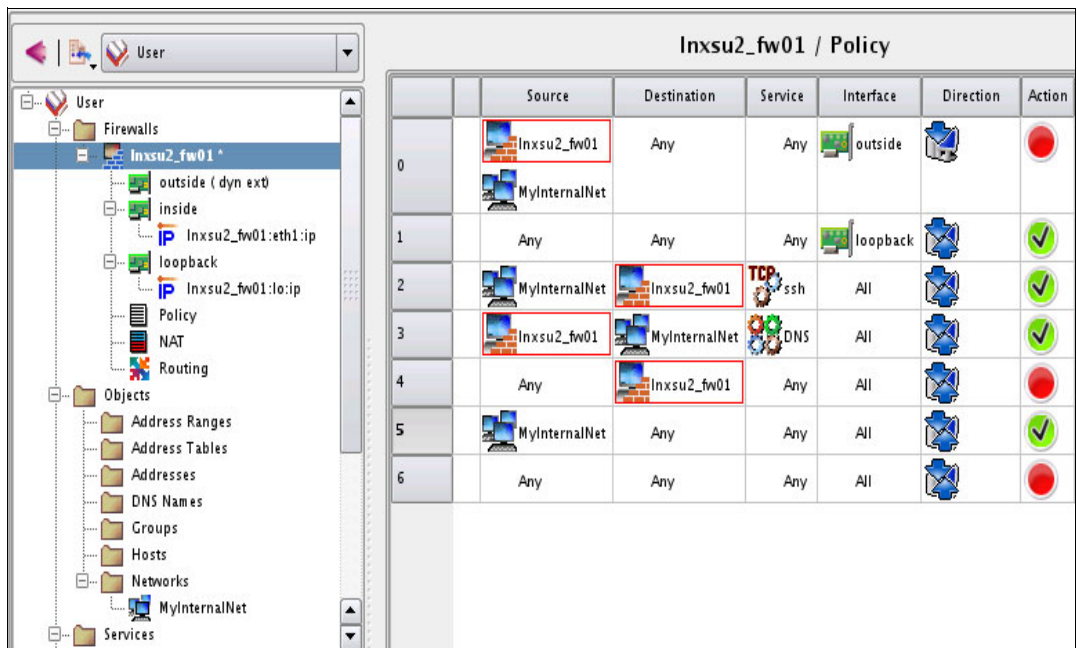


Figure 6-8 Example of a modified template

When you are done configuring your firewall, right-click on 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 6-14, 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 6-14 Firewall script generated by fwbuilder

```
#!/bin/sh
#
# This is automatically generated file. DO NOT MODIFY !
#
# Firewall Builder  fwbuilder v3.0.7-1477
[ ... ]
$IPTABLES -N In_RULE_0
test -n "$i_eth0" && $IPTABLES -A INPUT -i eth0 -s $i_eth0 -m state ...
$IPTABLES -A INPUT -i eth0 -s 192.168.1.1 -m state --state NEW -j In_RULE_0
$IPTABLES -A INPUT -i eth0 -s 10.1.0.0/16 -m state --state NEW -j In_RULE_0
test -n "$i_eth0" && $IPTABLES -A FORWARD -i eth0 -s $i_eth0 -m state ...
$IPTABLES -A FORWARD -i eth0 -s 192.168.1.1 -m state --state NEW -j ...
$IPTABLES -A FORWARD -i eth0 -s 10.1.0.0/16 -m state --state NEW -j ...
$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 Power System, 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>

The shorewall tool

Another popular firewall tool is *shorewall*. Unlike fwbuilder, it does not include a GUI and all configuration is performed by editing plain text files in `/etc/shorewall`.

As with fwbuilder, shorewall has no prepared package for Red Hat Enterprise Linux or SUSE Linux Enterprise Server; you have to build it from the sources. Fortunately, shorewall also provides a source RPM package for convenient building by using the `rpmbuild` tool.

Example 6-15 on page 175 shows an overview of the commands that are used to build the package. As you can see, the resulting package is architecture-independent and can be used on other platforms also, because shorewall itself is written in Perl and does not have to be compiled on architecture-dependent byte code.

Example 6-15 Building shorewall on Red Hat Enterprise Linux

```
[nico@lnxrh2 ~]$ sudo rpm -i shorewall-4.4.1-1.src.rpm
[nico@lnxrh2 ~]$ cd /usr/src/redhat
[nico@lnxrh2 redhat]$ sudo rpmbuild -bb SPECS/shorewall.spec
[ ... ]
[nico@lnxrh2 redhat]$ find RPMS/ -iname '*shorewall*.rpm'
RPMS/noarch/shorewall-4.4.1-1.noarch.rpm
RPMS/noarch/shorewall6-lite-4.4.1-1.noarch.rpm
RPMS/noarch/shorewall-lite-4.4.1-1.noarch.rpm
RPMS/noarch/shorewall6-4.4.1-1.noarch.rpm
[nico@lnxrh2 redhat]$ rpm -i \
/usr/src/redhat/RPMS/noarch/shorewall-4.4.1-1.noarch.rpm
```

The shorewall tool uses the concept of zones, very much like those introduced in 6.4.1, “The multizone concept” on page 161. You can control the network flow between different zones by defining policies that apply to all packets that pass through the firewall.

Assigning network interfaces to zones is done in the `/etc/shorewall/interfaces` file. Example 6-16 assigns the interface `eth0` to a zone for local addresses and `eth1` to the external zone. You can also limit the valid subset of IP addresses in a zone, as you can see with the local zone definition.

Example 6-16 Assigning network interfaces: /etc/shorewall/interfaces

```
lozit   eth0   detect  nets=(9.12.0.0/16)
extrn   eth1   detect
```

The zones themselves are defined in the `/etc/shorewall/zones` file. You can create any number of zones and arbitrarily choose their names to match your network nomenclature. The only restriction that applies is the allowed length of the zone name, which is configurable. Every zone has a name, a type, and can also have options. The *firewall* type zone is a reference to the firewall itself and you should include it in your zone configuration. Example 6-17 shows a sample zone configuration.

Example 6-17 Defining zones in /etc/shorewall/zones file

```
lozit   ipv4
extrn   ipv4
fw      firewall
```

After you have set up the interfaces and zones, you define default policies for the zones in the `/etc/shorewall/policy` file. Policies are applied to any packet that flows from one zone to another and has not been explicitly covered by any firewall rule. The policies in Example 6-18 on page 176 allow traffic from the internal network to any network, designated in the special zone name `all`. All traffic coming from external sources will be blocked and connection attempts will be logged. The firewall host can connect to any internal host.

Take note of the last rule that blocks all remaining traffic, ensuring that any traffic, which has not been covered by any previous policies, will not pass through the firewall. This approach ensures that the firewall acts as a whitelist, and is the recommended way to set up a secure firewall.

Example 6-18 The /etc/shorewall/policy file

```
losit  all ACCEPT
extrn  all DROP info
fw     losit  ACCEPT
all    all DROP
```

The only thing left for our simple shorewall example is to set up the rules for the traffic flows, which are contained in the `/etc/shorewall/rules` file. The rules in Example 6-19 allow any new connections from the local network to the firewall on TCP port 22, where the SSH daemon is usually listening. Remember that every connection that is not covered by the rules will be checked against the policies. This approach is sufficient for our simple example, but in your environment the rules can easily have several hundred lines.

Example 6-19 The /etc/shorewall/rules file

```
SECTION NEW
ACCEPT    losit      $FW      tcp    22
```

The new firewall configuration has to be compiled before it can be activated. To interact with shorewall, you use the command with the same name and provide an action to be performed as the second argument, as shown in Example 6-20.

Example 6-20 Compiling a firewall configuration with shorewall

```
[nico@lnxrh2 shorewall]$ sudo shorewall compile
Compiling...
Processing /etc/shorewall/params ...
Loading Modules...
Shorewall has detected the following capabilities:
  Address Type Match: Available
  CLASSIFY Target: Available
[ ... ]
Policy DROP from fw to lovpn using chain fw2lovpn
  Policy DROP from fw to extrn using chain fw2extrn
Generating Rule Matrix...
Creating iptables-restore input...
Compiling iptables-restore input for chain mangle:...
Shorewall configuration compiled to /var/lib/shorewall/firewall
```

Compiling the firewall rules place a shell script in `/var/lib/shorewall/firewall` that will be executed upon restart of the shorewall firewall, or a reboot of the system. As with the `fwbuilder` script, you can transfer the script to hosts that have a different architecture; you simply have to make sure that shorewall is installed there also.

If you want more information about building firewalls with shorewall, and the shorewall quick-start guides, see the shorewall Web site:

<http://www.shorewall.net>

6.7 Disk security

We have 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.

6.7.1 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.

6.7.2 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.

6.8 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 CEC, IOCP defines the physical resources that the LPAR will have access to. These resources include processors, memory (storage), network connections, and disk subsystems.

6.8.1 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.

6.8.2 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 re-create 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, these data sets 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 `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 recommendation 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.

6.9 Protecting Fibre Channel Protocol (FCP) disks

The security configuration of Fibre Channel SANs is usually done differently to 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.

6.9.1 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 disk 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 SET EDEvice command

FBA emulation devices are created by using the CP SET EDEvice command. The command syntax is shown at Figure 6-9.

```

>>-Set--EDEvice--edev----->
>>--+Type--FBA--ATTRibutes--+1750+--+| Paths |-----+--><
|                                     +-2105 +- '+ADD PATH - - - - +--| Paths | -' |
|                                     +-2107 +- ' DELEte PATH --' |
|                                     +-2145 +- |
|                                     ' - SCSI-' |
|-----|
|CLEAR-----|

Paths:

-----
V (1) |
|-----FCP_DEvice----rdev--WWPN--wwpn--LUN--lun--+-----|

Notes:
1. You can specify a maximum of 8 paths to the device

```

Figure 6-9 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.

We used the commands shown in Example 6-21 to add our FBA emulation device to CP.

Example 6-21 Commands adding an FBA emulation device to CP

```

set edev bc00 type fba attr 2105 fcp_dev b800 wwpn 5005076300cc9589 lun 520a0000
00000000
14:15:08 EDEV BC00 was created.
Ready; T=0.01/0.01 14:15:08
set edev bc00 type fba attr 2105 add path fcp_dev b900 wwpn 5005076300c89589 lun
520a000000000000
14:15:25 EDEV BC00 was modified.
Ready; T=0.01/0.01 14:15:25

```


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 6-22.

Example 6-22 Bringing an FCP EDEV online to z/VM

vary online bc00

14:16:59 BC00 varied online

14:16:59 1 device(s) specified; 1 device(s) successfully varied online

If a problem exists with any of the defined paths, CP generates an error message. Example 6-23 shows a situation we had with one of our emulated FBA devices when we were getting the correct SAN zoning defined.

Example 6-23 FCP EDEV errors with defined paths

vary online bc00

14:25:18 HCPSZP8701I Path FCP_DEV B900 WWPN 5005076300C89589 LUN 520A000000000000
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

Ready; T=0.01/0.01 14:25:18

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 6-24 shows the QUERY EDEVice DETAILS display for one of our FBA emulation devices.

Example 6-24 QUERY EDEVice DETAILS

q edev bc00 details

EDEV BC00 TYPE FBA ATTRIBUTES 2105

VENDOR: IBM PRODUCT: 2105800 REVISION: .135

BLOCKSIZE: 512 NUMBER OF BLOCKS: 9765632

PATHS:

FCP_DEV: B800 WWPN: 5005076300CC9589 LUN: 520A000000000000

CONNECTION TYPE: SWITCHED

FCP_DEV: B900 WWPN: 5005076300CC9589 LUN: 520A000000000000

CONNECTION TYPE: SWITCHED

FCP_DEV: B800 WWPN: 5005076300C89589 LUN: 520A000000000000

CONNECTION TYPE: SWITCHED

FCP_DEV: B900 WWPN: 5005076300C89589 LUN: 520A000000000000

CONNECTION TYPE: SWITCHED

Ready; T=0.01/0.01 12:11:22

Installing z/VM to FCP disks using FBA Emulation

To verify that installing z/VM to FCP disks is equivalent to installing on ECKD¹, 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 the *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 6-25 shows the commands and the responses we received.

Example 6-25 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
set edev bc00 type fba attr 2105 add path fcp_dev b900 wwpn 5005076300c89589 lun
521200000000000000
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
521300000000000000
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
521600000000000000
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
521700000000000000
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.

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.

3. The next task in the installation, task 5 of step 3, asked us to vary on the EDEVices we just created. Example 6-26 shows the result of the VARY ONLINE command we issued.

Example 6-26 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.

6.9.2 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, entered Single Object Operations for the CPC that is hosting our z/VM LPAR.
2. In the Groups Work Area, we double-clicked **Images**, then located our LPAR in the Images Work Area. We right-clicked on our LPAR and selected **CHPIDs**, which opened the CHPIDs Work Area.
3. We scrolled to find our FCP CHPID. We then right-clicked on it, and selected **CHPID Operations** → **Configure On/Off**, as shown in Figure 6-10 on page 184.

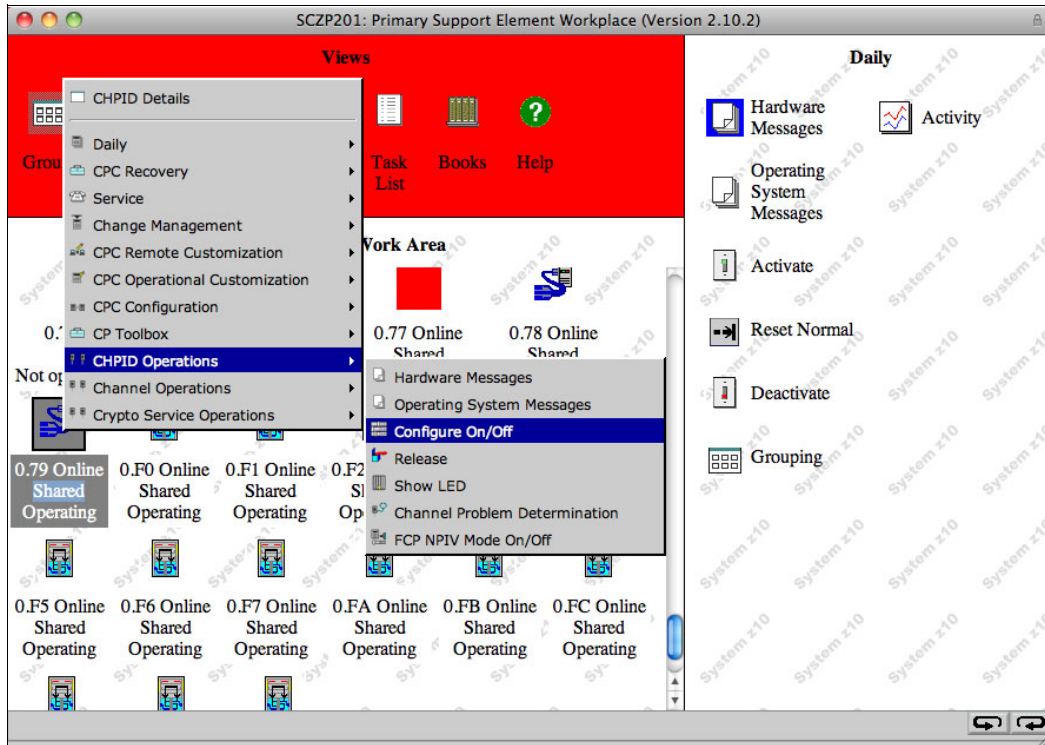


Figure 6-10 Configuring the FCP CHPID offline from our LPAR

- The panel to perform the CHPID configuration was displayed. We selected the CHPID to take offline, and clicked **Toggle**, as shown in Figure 6-11.

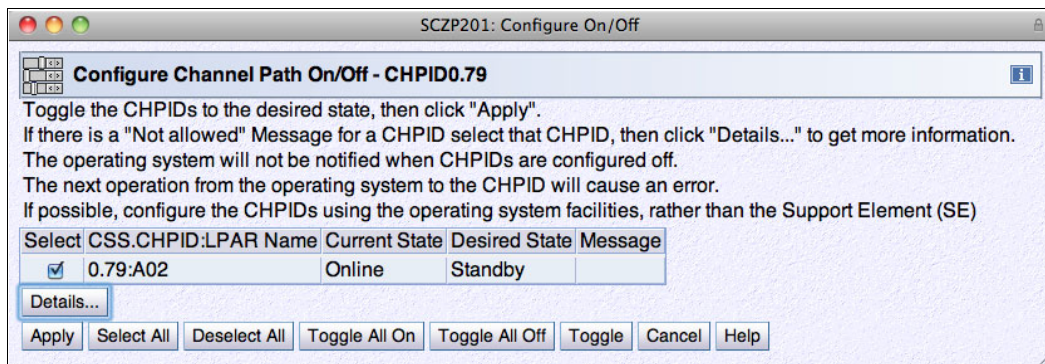


Figure 6-11 Configuring the FCP CHPID offline (2)

- We clicked **Apply**. The progress dialog opened, indicating that the operation was completed. See Figure 6-12 on page 185.

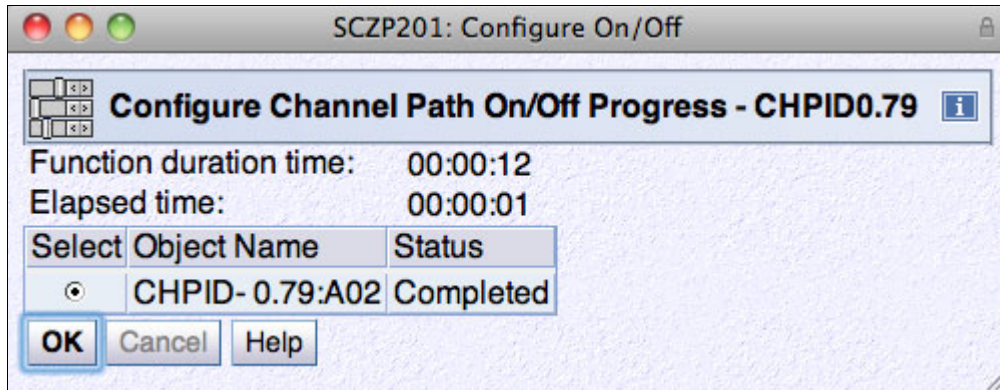


Figure 6-12 Configuring the FCP CHPID offline (3)

6. We clicked **OK**, then repeated the process for the other CHPID that had to be brought offline.
7. When the CHPIDs were offline, we could then change their NPIV state. We again right-clicked the **CHPID** icon, but instead, we selected **CHPID Operations** → **FCP NPIV Mode On/Off**, as shown in Figure 6-13.

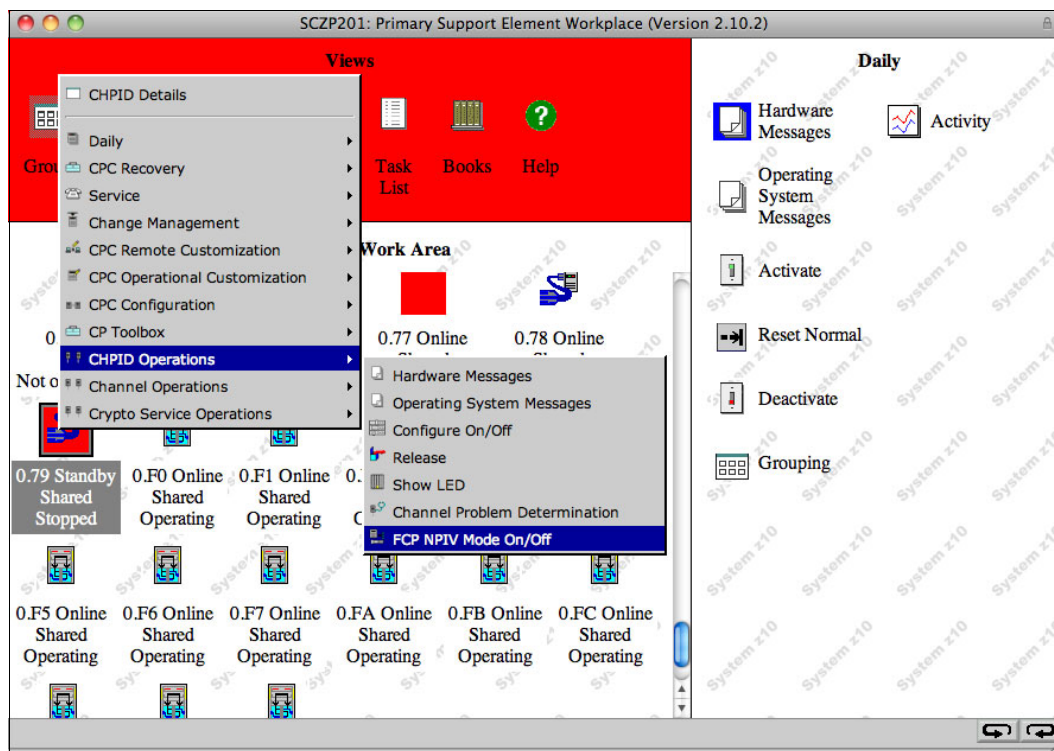


Figure 6-13 Selecting CHPID Operations FCP NPIV Mode On/Off

8. The NPIV Mode On/Off function was displayed. We selected the **NPIV Mode Enabled** check box, as shown in Figure 6-14 on page 186, and click **Apply**.

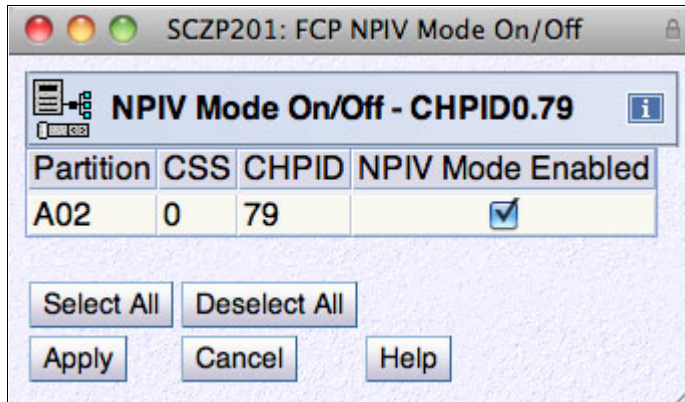


Figure 6-14 FCP NPIV Mode On/Off

- After a moment, the confirmation box opened, as shown in Figure 6-15, indicating that the change was successful.

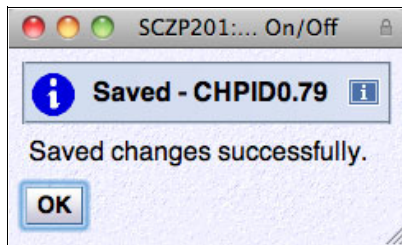


Figure 6-15 NPIV Mode confirmation box

- 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-enabled². 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.

² The IBM Redbooks paper *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 double-clicked **Groups**, then clicked **CPC**. We right-clicked on our CPC and selected **CPC Configuration** → **NPIV Configuration**, as shown in Figure 6-16.

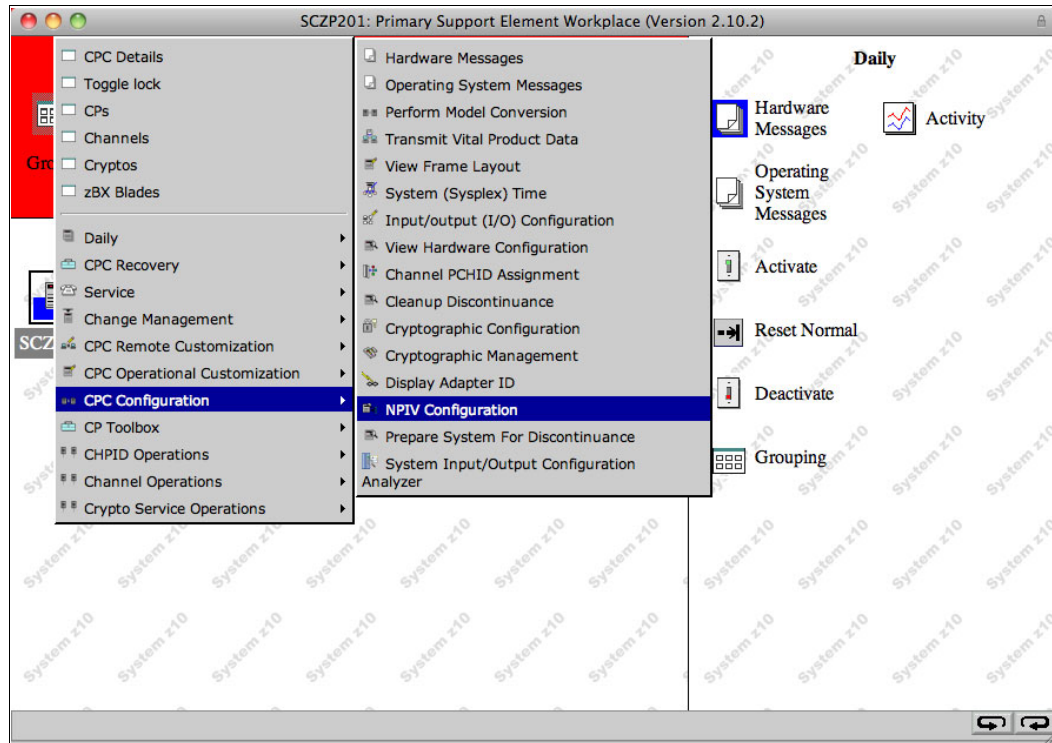


Figure 6-16 Selecting CPC Configuration NPIV Configuration

2. This operation displayed the FCP NPIV Port Names configuration selection panel, shown in Figure 6-17. The default action, **Display all NPIV port names that are currently assigned to FCP subchannels**, is the operation we used and selected **OK**.

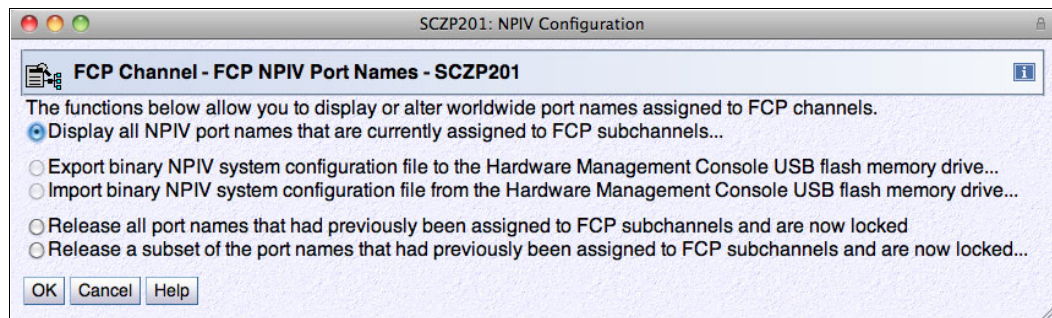


Figure 6-17 NPIV Configuration selection panel

- Another panel opened, shown in Figure 6-18, which allowed us to select the range of port names to be displayed. We selected **Display all assigned ports for an LPAR**, and then click **OK**.

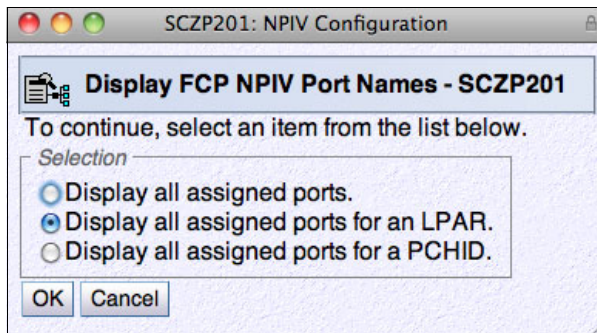


Figure 6-18 Port name range selection panel

- The panel, shown in Figure 6-19, 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**.

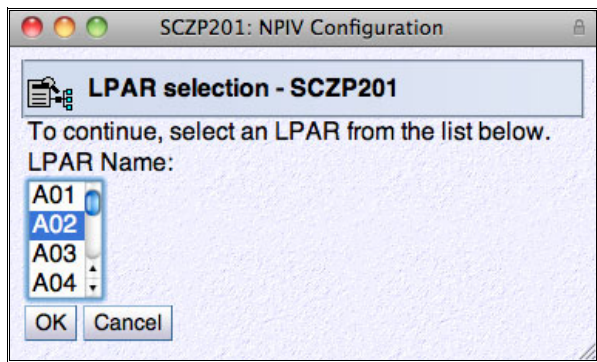


Figure 6-19 NPIV Port Names LPAR selection panel

The progress panel opened, as shown in Figure 6-20.

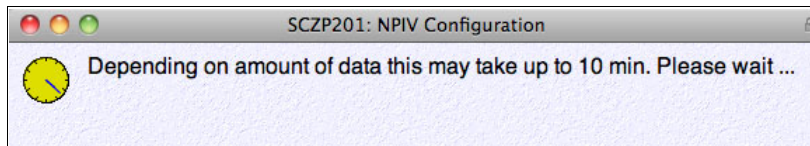


Figure 6-20 Progress panel for NPIV Port Name display

After several seconds, the Display Assigned Port Names list panel opened, as shown at Figure 6-21 on page 189.

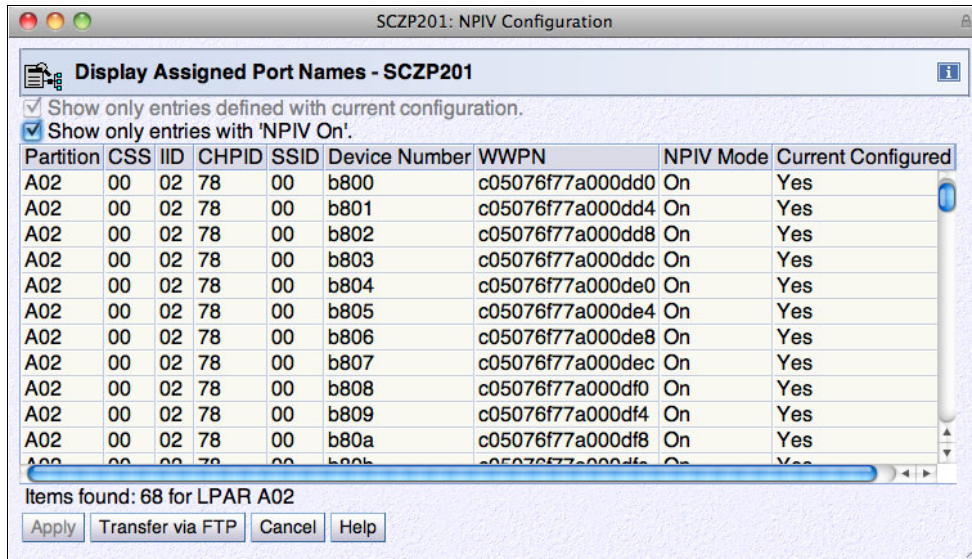


Figure 6-21 NPIV Display Assigned Port Names panel

- 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 6-22.

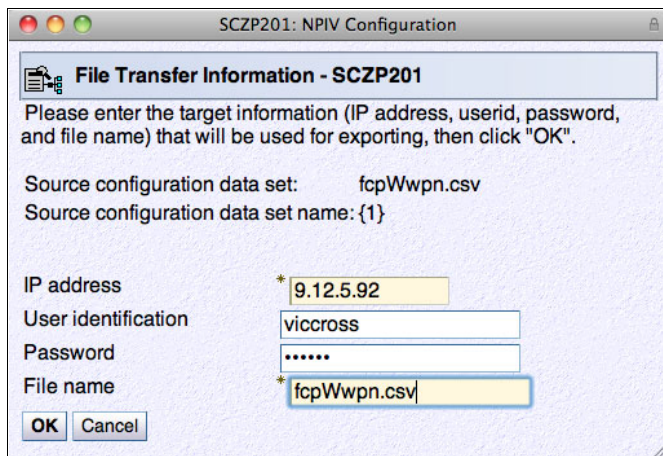


Figure 6-22 NPIV File Transfer Information panel

- When the FTP details were complete, we clicked **OK**, and after a little while we received the output shown in Figure 6-23.

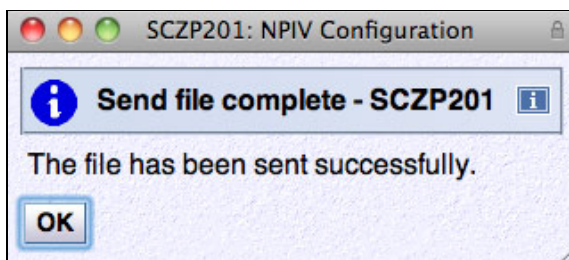


Figure 6-23 FTP file transfer confirmation

7. To verify the transferred file, we displayed it by using the `less` program on our Linux server; see Example 6-27.

Example 6-27 Transferred NPIV port name report (portion, displayed in the `less` program)

```
## Version: 1.0
## Machine serial number: 00002001DE50
## Current configuration filter enabled: Yes
## NPIV ON filter enabled: Yes
## Items for LPAR: A02
## partitionName,cssId,iid,chipidId,ssId,deviceNumber,wwpn,npiv mode,current conf
igured,pchid,phys. wwpn,
A02,00,02,78,00,b800,c05076f77a000dd0,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b801,c05076f77a000dd4,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b802,c05076f77a000dd8,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b803,c05076f77a000ddc,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b804,c05076f77a000de0,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b805,c05076f77a000de4,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b806,c05076f77a000de8,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b807,c05076f77a000dec,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b808,c05076f77a000df0,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b809,c05076f77a000df4,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b80a,c05076f77a000df8,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b80b,c05076f77a000dfc,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b80c,c05076f77a000e00,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b80d,c05076f77a000e04,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b80e,c05076f77a000e08,0n,Yes,0573,5005076401e25fca
A02,00,02,78,00,b80f,c05076f77a000e0c,0n,Yes,0573,5005076401e25fca
fcplwwpn.csv lines 1-22/74 27%
```

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 6-24.

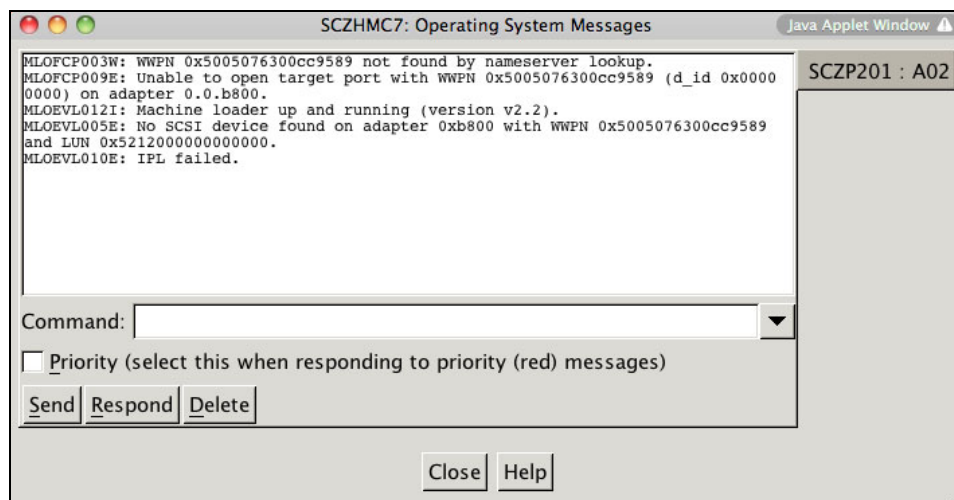


Figure 6-24 IPL errors because of improper SAN zoning

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 6-25.

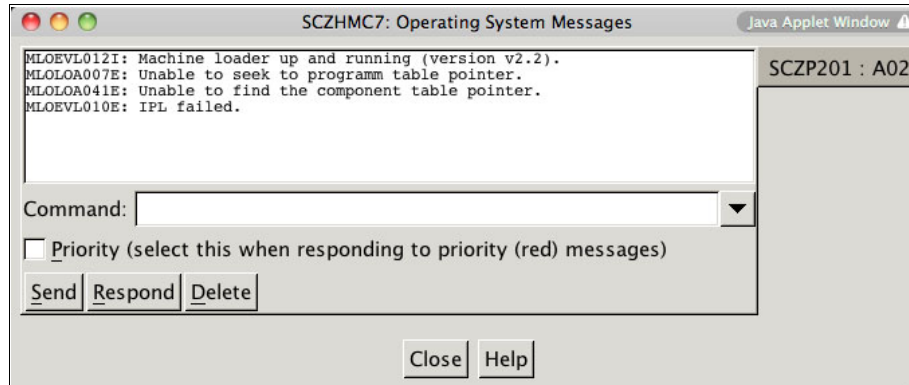


Figure 6-25 IPL error because of disk configuration

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 6-28 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 6-28 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.

6.10 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.

6.10.1 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 26.

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.

6.10.2 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 the *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 6-26 shows two ways that this might occur.

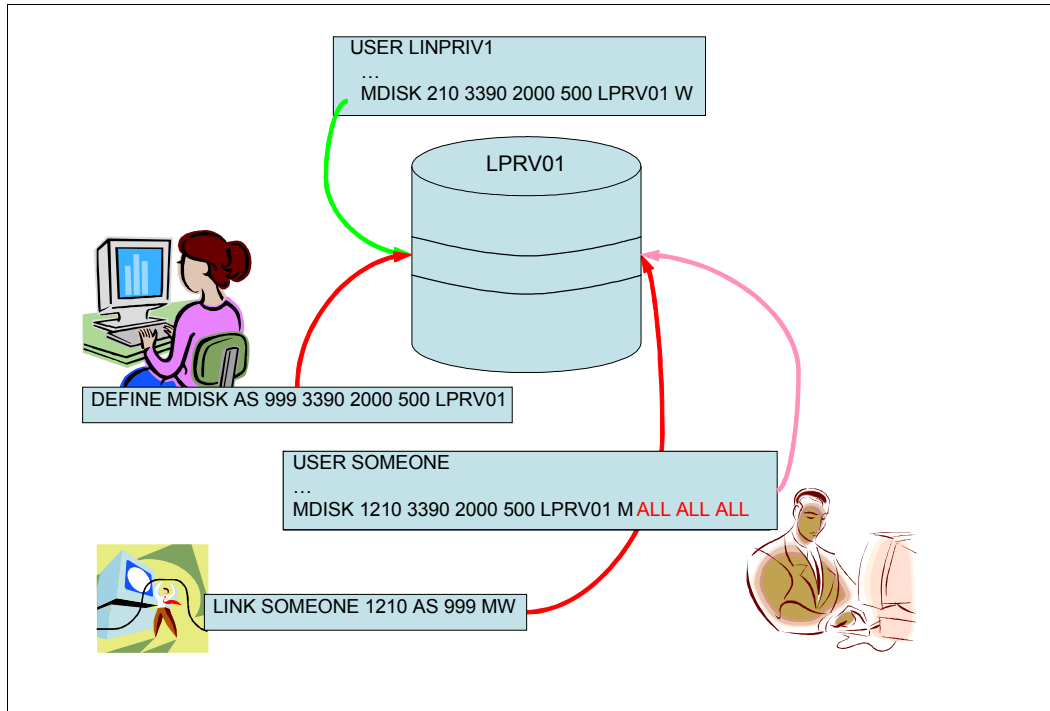


Figure 6-26 Overlapping minidisks causing data leakage

The minidisk is created and owned in the directory entry shown as the top of the diagram, and 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 non-existent 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 195.

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, by using 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 and 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 6.10.3, “Minidisk-owning user” on page 194, use discrete profiles for protecting minidisks. If generic profiles are used, a user can define an overlapping minidisk that matches the generic profile and be permitted access by RACF. 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.

6.10.3 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 protected according to the definition of that zone.

6.10.4 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 6.10.2, “Overlapping minidisks” on page 192, the exposure caused through 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 because 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.



Best practices

This chapter provides a collection of best practices for securing the 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 is already well documented by several sources. The practices can certainly be used to also secure a Linux on System z. However, several unique technologies that this platform leverages should be considered when you define the security policies, because these technologies can potentially 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.

This chapter discusses the best practices 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. Therefore, the chapter intends to serve as a quick implementation guide that you can use when assessing the security requirements for your environment.

7.1 Security checklist

The first recommendation to set up your virtual environment in a secure fashion is to at least take care of the items in the following check list. Your company most likely has specific requirements, in terms of information security, and these requirements should be taken into consideration when you build your own check list. We tried to build 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.

Throughout the next topics, not necessarily in this order, and maybe more than one at a time, the best practices, as we understand them, are discussed with the intention to make your virtual IT infrastructure secure, resilient, and compliant with your company's security policies.

7.2 Physical security

When outlining security policies for IT environments, system programmers or server administrators commonly take for granted the physical security. As the starting point for the best practices to secure your environment, make sure 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 159.

7.3 Securing the logical access to z/VM

While running Linux virtual guests with z/VM as the hypervisor, a logical assumption is that the z/VM security is as important as what you implement on 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 with this book, details can be found in *Security on z/VM*, SG24-7471. For 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.

7.3.1 z/VM user passwords

Although, on a z/VM system, the passwords for users are not world-readable, and only z/VM administrators with very specific privileges would be able to have access to the user directory, they still would be able to have access to all users passwords. For the sake of manageability and individual accountability, a strong recommendation is to have External Security

Management (ESM), such as RACF, as the back-end 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.

Nonetheless, we want to reinforce the importance of the rules regarding the reusable password and their complexity as well. Although this might seem a silly statement, some companies still use weak or even the default passwords on systems.

Additional information and reasons why to use an External Security Management tool to help securing your system can be found at Chapter 2, “The z/VM security management support utilities” on page 3.

7.3.2 Choosing the z/VM privilege class

The most basic level of security that z/VM provides, and is sometimes forgotten by Linux administrators, is the user’s privilege classes. The CP commands that any user is able to execute on a z/VM system depends on the privilege class they have defined. The list of classes and their meanings can be found at Table 2-1 on page 24.

As a starting point, a best practice is to 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 Linux virtual guest should not have additional privileges to define system-wide parameters of the z/VM system nor other virtual guests 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 are not discussed in this book, we have described the process to activate and use some of them within the book. For specific information about the classes and which resources each one of them controls, see *z/VM: RACF Security Server Security Administrator’s Guide* (for V5R4.0), SC24-6142.

7.3.3 z/VM network connection

Many alternative ways exist to connect to a z/VM system: through the HMC, with the service elements, through a physical console connected to a terminal controller, with the OSA-Express integrated console controller, or your through 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, requiring 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, therefore if someone was listening on your network, sniffing for security vulnerabilities would be easy.

A strong recommendation is to have all Telnet communication, 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 2.6, “Securing network access to z/VM” on page 14.

7.4 Securing the data

Ensuring the protection and security of an organization's information can be considered the foundation for today's successful businesses. Although 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 requires.

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 do 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 on ESM see 2.7, "Securing z/VM resources" on page 24.

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.

7.4.1 Securing your minidisks

The z/VM system was designed in such way that unless otherwise granted, the 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 the user's MDISK directory statements are being processed, or when the user issues a LINK command for a self-owned minidisk.

If the VMMDISK class had not previously been 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 on Example 7-1.

Example 7-1 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, see Example 7-2.

Example 7-2 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 7-3 has the command to create the RACF profile. We create the profile for the minidisk by using virtual address 201 for user LNXSU1.

Example 7-3 Creating a VMMDISK profile

```
RAC RDEFINE VMMDISK LNXSU1.201 UACC(NONE)
```

Note: 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 more restrictive access possible, READ) would be able to copy the data files to another minidisk, for which 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 smaller number of users and groups as possible.

To change the UACC of a minidisk if the profile was not defined this way, you may execute the command in Example 7-4.

Example 7-4 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 Example 7-5.

Example 7-5 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 on your user IDs minidisks profiles is important.

To check the privileges that are granted on the LNXSU1.201 profile, you can execute the command shown in Example 7-6 on page 202.

Example 7-6 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 authority for minidisks are NONE, READ, UPDATE, CONTROL, and ALTER.

Note: For detailed information about the access authority for minidisks, refer to the *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 recommended 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 7-7.

Example 7-7 Linking a minidisk without proper privileges

```
CP LINK LNXSU1 201 201 RR
RPIMGRO32E 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 7-8.

Example 7-8 Granting permission to access a minidisk

```
RAC PERMIT LNXSU1.201 CLASS(VMMDISK) ID(LNXSU2) ACCESS(READ)
```

Note: You may use the same command to grant privileges to groups.

If the user LNXSU2 no longer needs access to user LNXSU1 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 7-9.

Example 7-9 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.

7.4.2 Reducing the intrusion points with shared disks

A golden rule on 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) or similar. 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 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 subject to discussion 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 on “Protecting access to minidisks” on page 26, 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 7-1.

```
[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
[root@lnxrh1 /]#
```

Figure 7-1 Linking a minidisk with the wrong password

When ESM is not used, the manageability is seriously compromised. Although the access to the user directory is protected and restricted to users with privileged access, it is however a clear text file, so the use of an ESM is strongly recommended. 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, and we had chosen RACF for this book, is considered a best practice for securing Linux servers on the system z platform. Granting access to a minidisk, using RACF, can be accomplished as shown in Example 7-10.

Example 7-10 Allowing read access to a minidisk

```
RAC PERMIT LNXSU1.300 CLASS(VMMDISK) ID(LNXRH1) ACCESS(READ)
```

After you have granted permission for LNXRH1 link, the LNXSU1's 300 disk, you can proceed with a 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 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 the following figures, you only have to bring the disk online in Read Only mode and make it available.

Figure 7-2 shows an example on LNXSU1.

```
lnxsu1:/ # df -h /mnt/
Filesystem      Size  Used Avail Use% Mounted on
/dev/dasdc1     69M   40K   65M   1% /mnt
lnxsu1:/ #
```

Figure 7-2 Minidisk mounted R/W on the source node

Figure 7-3 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
[root@lnxrh1 ~]#
```

Figure 7-3 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.

7.4.3 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 terms, *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.

Our advice here is to 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 129.

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 recommend 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 132.

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.

7.5 Securing the network

The System z platform leverages a great advantage from the security point of view with the 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 can at the same time provide auditing and troubleshooting functionality. A possibility is provide Linux guests with connectivity and dramatically reduce the 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 are not detailed within this chapter; for specific configuration details and system requirements, see *z/VM V5R4.0 Connectivity*, SC24-6080-07. 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 25.

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 7-4. 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.

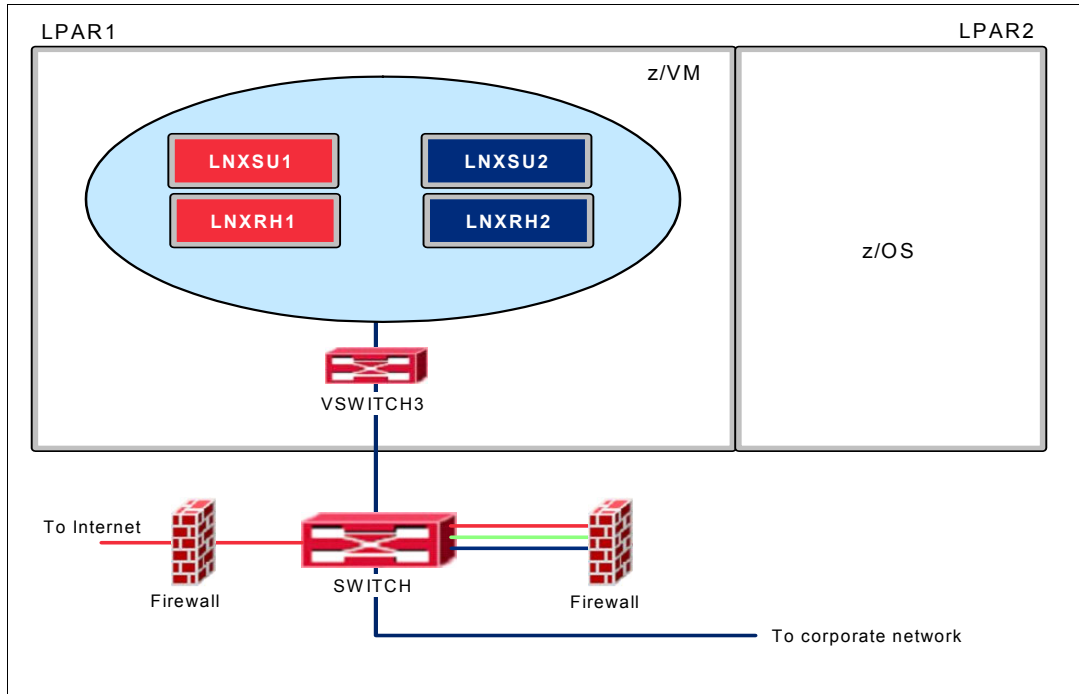


Figure 7-4 VSWITCH infrastructure

7.5.1 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 7-11.

Example 7-11 Creating a Layer2 VSWITCH

```
DEFINE VSWITCH VSWITCH3 RDEV 3083 30A3 ETHERNET CONTROLLER *
```

This command produces the output shown in Figure 7-5 on page 207.

```

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 7-5 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.

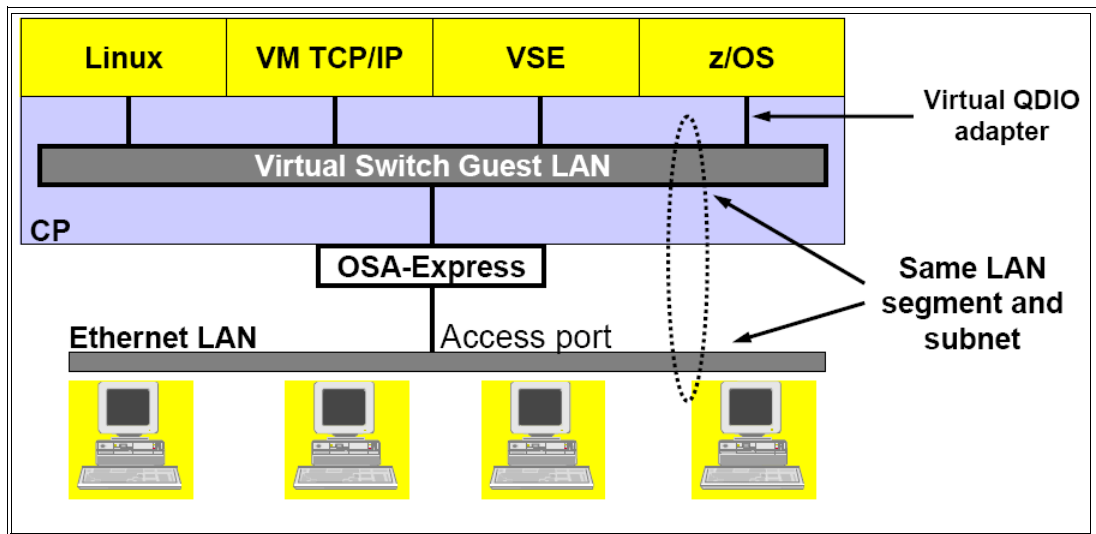


Figure 7-6 VLAN-Unaware VSWITCH infrastructure

Without any previous permission, if you try to couple to the VSWITCH from your Linux server, an error occurs, as shown in Figure 7-7. 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
lnxsu1:~ #

```

Figure 7-7 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 7-12.

Example 7-12 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 7-13.

Example 7-13 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 7-14 shows the syntax to use.

Example 7-14 Listing a VSWITCH's access list

```
QUERY VSWITCH VSWITCH3 ACCESSLIST
```

This command produces output shown in Figure 7-8. 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 7-8 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 7-9.

```
lnxsu1:~ # vmcp couple 700 system VSWITCH3
NIC 0700 is connected to VSWITCH SYSTEM VSWITCH3
lnxsu1:~ #
```

Figure 7-9 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 V5R4.0 Connectivity*, SC24-6080-07.

RACF can improve the security requirements, managing policies to grant appropriate access to your systems 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 7-15 shows the command to use to activate the class.

Example 7-15 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 7-10.

```
RAC SEARCH CLASS(VMLAN) MASK(SYSTEM.VSWITCH3)
SYSTEM.VSWITCH3
```

Figure 7-10 Listing the base profile for a VSWITCH

If a RACF profile is not created for your VSWITCH, you may create it by using the command in Example 7-16.

Example 7-16 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 the 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 7-17 shows the command that should be executed to grant access to our LNXSU1 user to the VSWITCH3 VSWITCH, currently managed by RACF.

Example 7-17 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 one 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 7-18.

Example 7-18 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 7-19, we created a RACF group and named it VSW3GRP.

Example 7-19 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 7-20. The syntax is the same used in Example 7-17 on page 209, using the group VSW3GRP in the ID field.

Example 7-20 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 7-21.

Example 7-21 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.

7.5.2 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 7-11 on page 211.

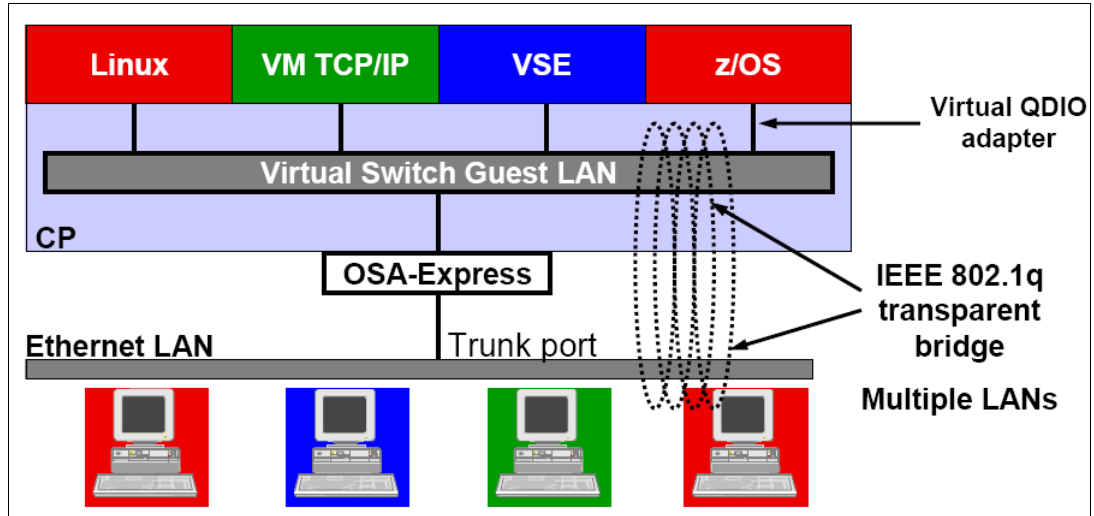


Figure 7-11 VLAN-aware VSWITCH infrastructure

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 recommended 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 7-1 lists options to be used with VLAN-aware virtual switches.

Table 7-1 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	VLANID 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 the available options for the VSWITCH definition, refer to *z/VM V5R4.0 Connectivity*, SC24-6080-07.

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 security administrators. The scenario is shown on Figure 7-12 on page 212.

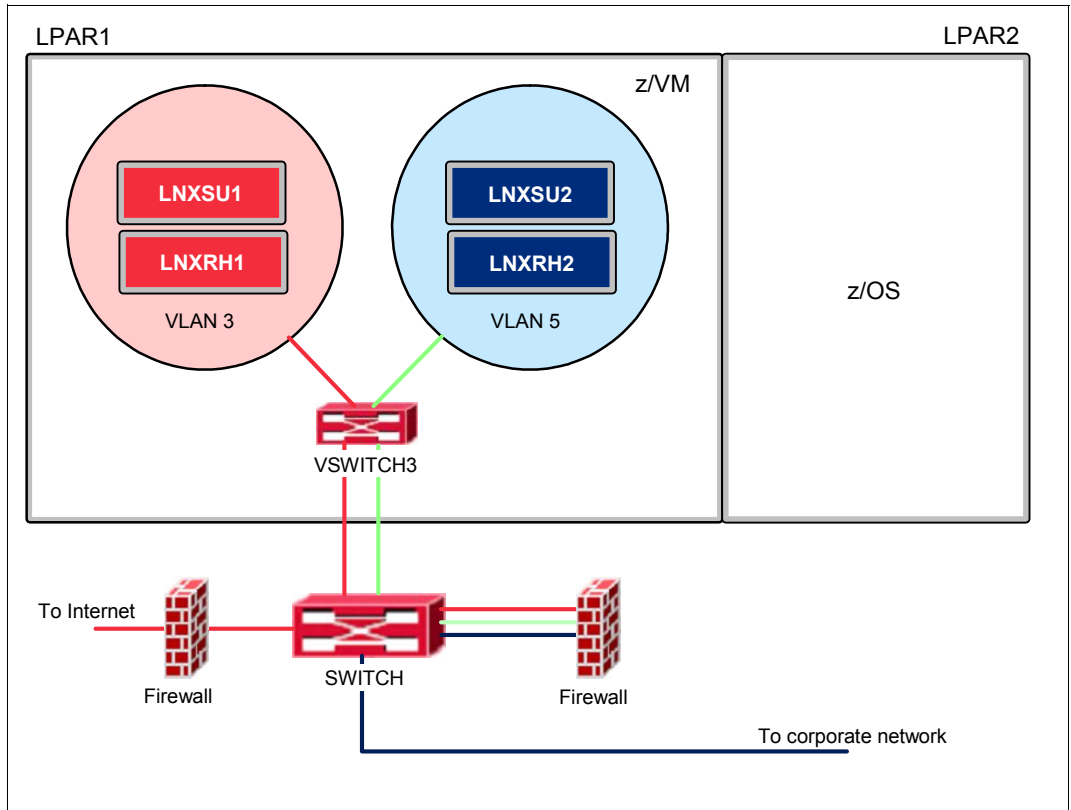


Figure 7-12 VLAN-aware VSWITCH infrastructure

Example 7-22 shows how to create the VLAN-aware VSWITCH for the scenario we have discussed.

Example 7-22 Creating a VLAN-aware VSWITCH

```
DEFINE VSWITCH VSWITCH3 RDEV 3083 30A3 VLAN 3 NATIVE 1 ETHERNET CONTROLLER *
PORTTYPE ACCESS
```

Figure 7-13 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 7-13 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 defaults 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 whether the guest is stripping VLAN IDs from the frames that flow through its interfaces. In Example 7-23, we grant privileges to LNXSU1 to couple to VSWITCH3 using VLAN ID 0003 and ACCESS port type.

Example 7-23 Updating CP access list granting access to a virtual guest to a VSWITCH VLAN-aware

```
SET VSWITCH VSWITCH3 GRANT LNXSU1 VLAN 3 PORTTYPE ACCESS
```

Note: If not specified, the user ID for whom 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 rules 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 7-14.

```
RAC SEARCH CLASS(VMLAN) MASK(SYSTEM.VSWITCH3.0003)  
SYSTEM.VSWITCH3.0003
```

Figure 7-14 Listing the IEEE VLAN-aware VSWITCH profile

If the profile does not exist, you can create it with the command in Example 7-24. Remember to always have the universal access authority (UACC) defined to NONE to avoid unwanted access to the device.

Example 7-24 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 to guarantee 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 7-20 on page 210, replacing the group VSW3GRP with VLAN0003, and then giving this new group the same UPDATE privilege on the RACF profile for the VLAN 0003 (SYSTEM.VSWITCH3.0003). Use the same command as in Example 7-21 on page 210.

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 7-22 on page 212.

All new guests coupling to the VSWITCH3 using VLAN 0003 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 7-15 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 7-12 on page 212.

```
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 7-15 Summary of commands

With this example we assume 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: When 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 striping the frames for the VLANs it is supposed to communicate with.

7.5.3 Virtual switch port isolation

A virtual switch and an OSA-Express card port can be shared by multiple TCP/IP stacks. What does it mean? It 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 7-16 on page 215.

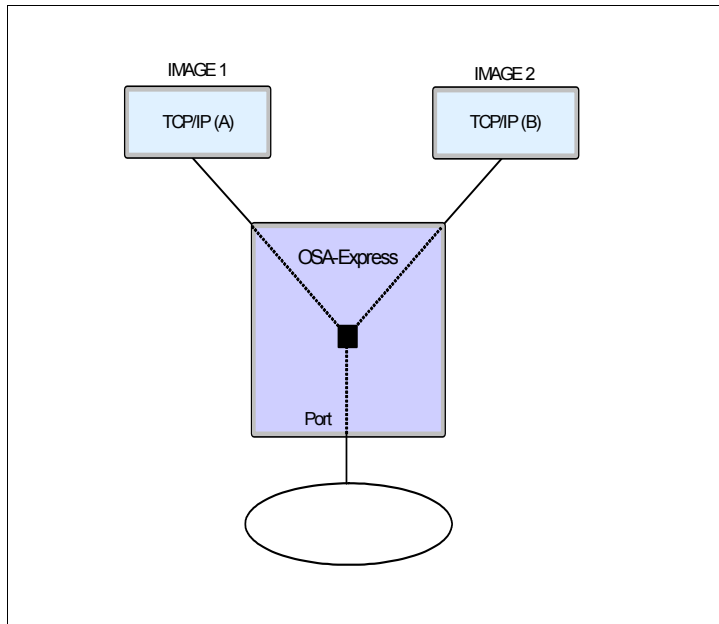


Figure 7-16 OSA-Express port sharing

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 virtual switch. The communications take place within the boundaries of the central electronic complex (CEC).

Although the ability to do image-to-image communication can represent a great advantage from the performance stand point, 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 to be able to reach others without going outside the OSA-Express port. From a security perspective, this isolation feature can help 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 the need would not exist 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 7-25.

Example 7-25 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 the other 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.

7.5.4 Network diagnostics

Another feature of z/VM has 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.

A 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 VSWITCHes 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 7-26, we allow the virtual guest LNXSU1 to couple to the virtual switch VSWITCH3 using promiscuous mode.

Example 7-26 Allowing promiscuous mode

```
SET VSWITCH VSWITCH3 GRANT LNXSU1 PRO
```

A recommended 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 7-27.

Example 7-27 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 the Example 7-28, we activate the network interface card by using virtual address 700 from the Linux console, using vmcp tool.

Example 7-28 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 the network data capturing, you can check the condition of your network card with the following command. We assumed for Example 7-29 that your network card virtual address is 700.

Example 7-29 Querying network interface details

```
vmcp query nic 700 detail
```

This example produces the output shown in Figure 7-17.

```
Inxsul:~ # 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 7-17 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 7-18 on page 218.

```

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 7-18 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 a HiperSockets check how to use TRSOURCE on *z/VM V5R4.0 Connectivity*, SC24-6080-07.

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 recommend the use of groups when possible.

You can create the RACF group by using the same command as shown in Example 7-20 on page 210, and then connect the user that you want to this new group. In Example 7-30, we assume that a new group named SNIFF was created and the virtual user id LNXSU1 was connected to this group.

Example 7-30 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 simply removing them from the special SNIFF group that was created for this purpose. See Example 7-31 on page 219.

Example 7-31 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.

7.5.5 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 7-32.

Example 7-32 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 recommended 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 in Example 7-33.

Example 7-33 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 7-34.

Example 7-34 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 recommendation that should be implemented on each virtual users 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 unauthorized access, and traffic control are concerned.

7.5.6 Implementing mandatory access control

Until now, 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's security, they are all discretionary rules that must be carefully maintained to avoid security breaches. Imagine what might happen in a hypothetical scenario, where hundreds of servers need to be supported, if for 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 in jeopardy 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 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 would guarantee that only subjects previously authorized to use a resource within a security zone, would have the privilege to do so, as shown on Figure 7-19.

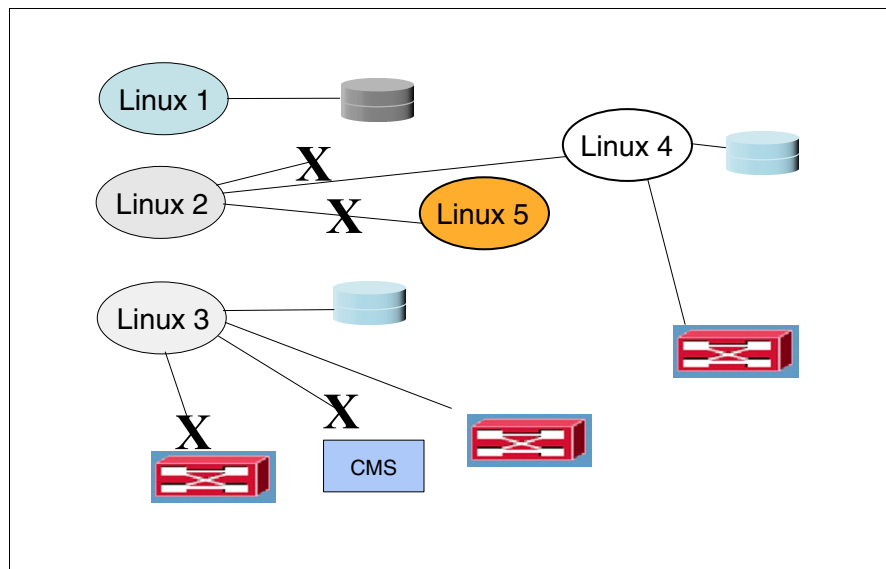


Figure 7-19 Separating resources in security zones

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's 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 7-35.

Example 7-35 Creating a default security level and data partition

```
RAC RDEFINE SECDATA SECLEVEL ADDMEM(DEFAULT/100)
RAC RDEFINE SECDATA CATEGORY ADDMEM(EXTRANET INTRANET)
```

In the 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 depends 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 7-36 shows how to group the security levels and categories together.

Example 7-36 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 7-12 on page 212 would have to be adapted to what is shown in Figure 7-20 on page 222. Two virtual switches are required, because the same VSWITCH cannot be on different security zones.

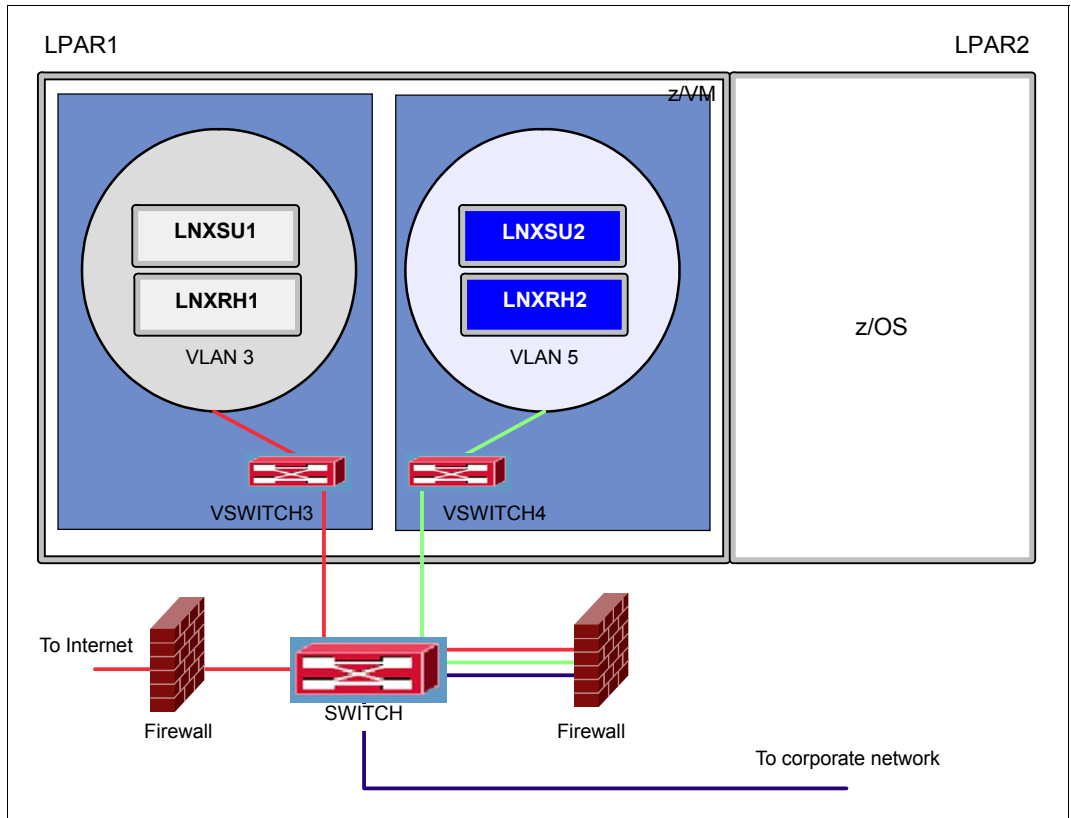


Figure 7-20 VLAN-aware infrastructure with SECLABEL active

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 7-37 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 7-38, is important.

Example 7-38 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 responsibility to keep the 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 on Example 7-39 on page 223.

Example 7-39 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 7-40 to activate the SECLABEL and VMMAC classes.

Example 7-40 Activating the SECLABEL and VMMAC RACF classes

```
RAC SETROPTS CLASSACT(SECLABEL VMMAC)
RAC SETROPTS RACLIST(SECLABEL)
```

Then, you can activate 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, Example 7-41.

Example 7-41 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 7-42.

Example 7-42 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. The opposite is true from the object's perspective also. The possible combinations between the user's security labels and 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 7-21 on page 224 occurs.

```

1 lnrxh1 lnrxh1
  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 7-21 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 on its attempt to couple to VSWITCH3, because it does not have enough security clearance. To grant access to LNXRH1, we can simply repeat Example 7-37 on page 222 and Example 7-38 on page 222 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 7-22.

```

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 7-22 Successful log on 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 recommend a detailed study of the *z/VM: Secure Configuration Guide* (for V5R4.0), SC24-6158.

7.6 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. However, 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 had been 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 became 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 strengthen 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 at Chapter 4, "Authentication and access control" on page 97.

7.7 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 enough proof of their identity.

Linux offers an extremely 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 an extremely powerful instrument to reinforce the compliance to 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 109 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 the authentication requirements vary from one installation to the other. 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 the overall security.

To implement the RSA key pair method you can simply:

1. Create a key pair on the server you are connecting from, as shown in Figure 7-23.

```
[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
[root@lnxrh1 ~]#
```

Figure 7-23 Creating a RSA key pair

2. Copy the public key you had just created. See Figure 7-24.

```
[root@lnxrh1 ~]# cat /root/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAssER1/xik1gvSGLiEvvYSWR9DCnYFcCh0LnGb5ETVAzqQ/R3
LKDSvqrp05QU2phT8Ggq2R0mkg2i5g8ygJHwsoD1cpoKxD4zuSJG7//kz6q3xzv3dG0bzyoC2s1u
I9BBznThZzncKmuJkCHZWMh1QJwzdK199Ugf19LfIEc2/ok= root@lnxrh1.itso.ibm.com
[root@lnxrh1 ~]#
```

Figure 7-24 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 7-25, the user ID is `ricardo`.

```
[root@lnxrh2 ~]# echo "ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAssERl/xik1gvSGLiEvvYSWR9DCnYFcCh0LnGb5ETVAzqQ/R3
LKDSvqrp05QU2phT8Ggq2R0mkg2i5g8ygJHWsoD1cpoKxD4zuSJG7//kz6q3xzv3dG0bzyoC2s1u
I9BBznThZzncKmuJkCzHWMh1QJwzdK199Ugf19LfIEc2/ok= root@lnxrh1.itso.ibm.com"
>> ~ricardo/.ssh/authorized_keys
[root@lnxrh2 ~]#
```

Figure 7-25 Copying the public key into `authorized_keys`

4. Then, log on the target system by using your RSA key pair, as shown on Figure 7-26.

```
[root@lnxrh1 .ssh]# ssh ricardo@lnxrh2.itso.ibm.com
Enter passphrase for key '/root/.ssh/id_rsa':
[ricardo@lnxrh2 ~]$
```

Figure 7-26 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 the `ssh-agent` as an option only to provide the passphrase to open the key once, as shown in Figure 7-27.

```
[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 ricardo@lnxrh2.itso.ibm.com
Last login: Tue Sep 29 13:47:34 2009 from 9.12.5.93
[ricardo@lnxrh2 ~]$
```

Figure 7-27 Using the `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 better suit your needs.

7.8 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.

7.8.1 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 different servers.

As mentioned in 7.7, "Authentication" on page 225, 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 one 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 could potentially be in jeopardy.

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 z/VM LDAP server" on page 49 has instructions for how to set up and correctly configure the z/VM LDAP server, and with it, use an SDBM as the back end database and make use of existing RACF user definitions for Linux users. In addition, you can extend to 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 recommend that your Linux user IDs are on a centralized repository, keeping the super user (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 115 for a further discussion on 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 a LDAP server can cause on your external (outside the System z server) network segment.

7.8.2 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 recommend to use a secure, encrypted channel, which this 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 z/VM LDAP server” on page 49 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 recommend those applications to 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).

7.9 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.

The existence of a well defined security policy, although vital for the maintenance of your system’s security, 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 2.7.5, “Centralized audit” on page 29 has information for 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, you have to control the security to your physical resources, with the advantage of being able to control everything from a central location. Using the methods described in 2.7.5, “Centralized audit” on page 29, 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.

7.10 Separation of duties

As far as IT security is concerned, separation of duties, although laborious and in some cases expensive, is definitely a best practice. It avoids conflict of interests 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 recommended by defining the policies, and having strong control of who in the organization has access to the super user 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 other 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.

For numerous organizations, the same person might have various roles to play. We strongly recommend that each role has 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 z/OS features in a Linux environment

In this appendix, we discuss several z/OS features that can be used to enhance security in a Linux on System z environment. In addition, Linux for System z can provide a number of security functions for greater flexibility in a z/OS environment.

Authentication using IBM Tivoli Access Manager

IBM Tivoli® Access Manager for e-Business forms the basis for the IBM Tivoli Access Manager product suite. Delivered as an integrated solution, this suite of products provides access control management solutions to centralize the network and application security policies for distributed applications.

In this section, we discuss using Tivoli Access Manager for Linux user authentication against users who are defined to RACF, running in a z/OS LPAR. The scenario is illustrated in Figure A-1.

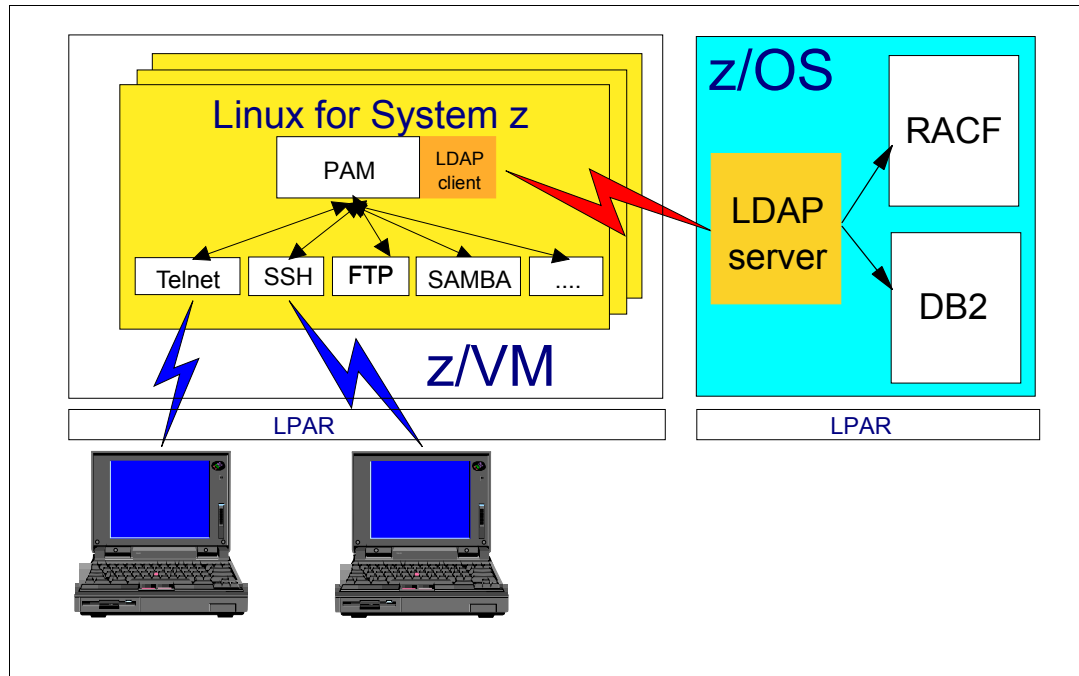


Figure A-1 User authentication using Tivoli Access Manager

In this scenario, network clients (such as Telnet and SSH) are authenticated against users defined to an LDAP directory running on z/OS. Linux users are defined in the LDAP directory (stored in DB2®) on z/OS. However, no user passwords are stored in the LDAP directory. Instead, password authentication is performed against user passwords that are stored in RACF running on z/OS. Using the capabilities provided by PAM are described in 4.3, “Pluggable Authentication Modules” on page 109.

Using the LDAP server on z/OS

The current LDAP server used on z/VM is a direct migration of the LDAP server that became available on z/OS 1.10. If your organization is already using a z/OS server, using that same server also for your z/VM Linux environment might be easier. Details of how to customize an LDAP server for this purpose is in 3.1, “The z/VM LDAP server” on page 50.

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 back-end Web application server resources into its security policy.

In addition to authenticating Linux users, we can use WebSEAL in conjunction with LDAP running on z/OS to protect Web-based resources. This scenario is depicted in Figure A-2.

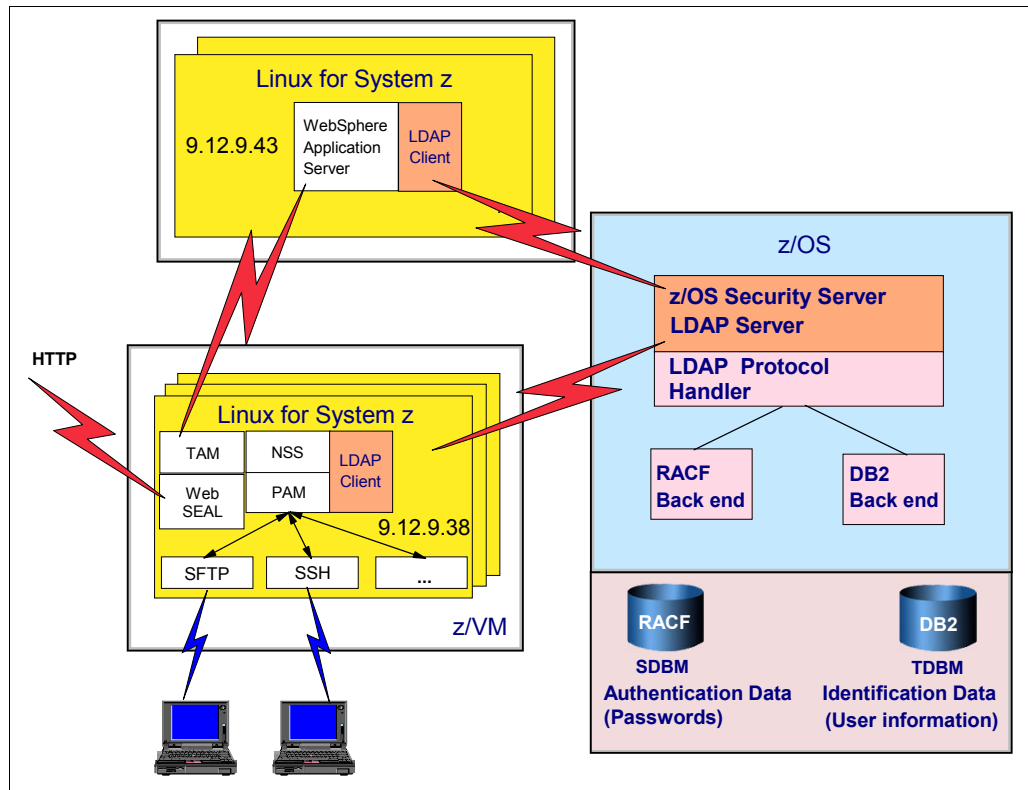


Figure A-2 Authenticating users against a z/OS RACF database

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.



z/VSE Security and Linux on System z

z/VSE is designed to provide robust, cost-effective solutions for customers with a wide range of processor capacity requirements. This is especially important to customers that have lower processor capacity requirements and value the low cost of operation and administration.

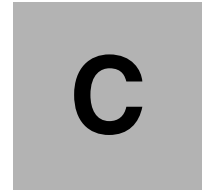
As with all IBM System z operating systems, z/VSE can run as a single operating system or share the mainframe with others by using the LPAR capabilities of System z. z/VSE can also run as a guest on the IBM z/VM system to use a customized environment that might contain emulated hardware features that are not necessarily installed on that particular model of server.

Transmission Control Protocol/Internet Protocol (TCP/IP) in z/VSE allows interconnection to other platforms. With TCP/IP, z/VSE became accessible through the Internet. Special security functions are provided to make those connections secure.

TCP/IP enables z/VSE to support single sign-on with Lightweight Directory Access Protocol (LDAP) and secured integration in a multiplatform environment by using z/VSE connectors.

z/VSE supports the System z cryptographic solutions. They are used within the connection security and also within the tape encryption.

For more information and examples of how z/VM and Linux on System z can be used to secure the z/VSE environment, see *Security on IBM z/VSE*, SG24-7691.



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 Web site 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.

Abbreviations and acronyms

AES	Advanced Encryption Standard	SCSI	Small Computer System Interface
AP	Adjunct Processor	SHA	Secure Hash Algorithm
CA	certificate authority	SSM	Software Security Module
CEX2	Crypto Express2	TDES	triple DES
CEX3	Crypto Express3	TPM	Trusted Platform Module
CEX2A	Crypto Express2 in accelerator mode	VFS	virtual file system
CEX2C	Crypto Express2 in coprocessor mode		
CPACF	Central Processor Assist for Cryptographic Function		
CPC	central processor complex		
DAC	discretionary access control		
DASD	direct access storage device		
DES	Data Encryption Standard		
ESCON®	Enterprise Systems Connection		
FCP	Fibre Channel Protocol		
FICON	fibre channel connection		
GSKit	Global Security Toolkit		
IBM	International Business Machines Corporation		
IBMPKCS11Impl	Java IBM PKCS#11 implementation		
ITSO	International Technical Support Organization		
JRE	Java Runtime Environment		
HSM	Hardware Security Module		
LDAP	Lightweight Directory Access Protocol		
LPAR	logical partition		
LUKS	Linux Unified Key Setup		
LVM	Logical Volume Manager		
MAC	mandatory access control		
NAT	Network Address Translation		
NSS	Network Security Services		
PAM	Pluggable Authentication Module		
PCI DSS	Payment Card Industry Data Security Standard		
PRNG	pseudorandom number generator		
PU	processing unit		
RACF	Resource Access Control Facility		
SDBM	secure database manager		

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

For information about ordering these publications, see “How to get Redbooks” on page 242. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *Security on z/VM*, SG24-7471
- ▶ *IBM System z10 Enterprise Class Technical Guide*, SG24-7516
- ▶ *IBM System z10 Business Class Technical Overview*, SG24-7632
- ▶ *IBM System z10 Enterprise Class Configuration Setup*, SG24-7571
- ▶ *Linux on IBM eServer zSeries and S/390: Best Security Practices*, SG24-7023
- ▶ *Security on z/VM*, SG24-7471

Other publications

These publications are also relevant as further information sources:

- ▶ *z/VM Planning and Administration*, SC24-5995
- ▶ *Device Drivers, Features, and Commands as available with SUSE Linux Enterprise Server 11*, SC34-2595
- ▶ *System z10 Support Element Operations Guide*, SC28-6979
- ▶ *z/VM: RACF Security Server System Programmer's Guide* (for V5R4), SC24-6149
- ▶ *z/VM: RACF Security Server Security Administrator's Guide* (for V5R4.0), SC24-6142

Online resources

These Web sites are also relevant as further information sources:

- ▶ z/VM Security and Integrity
<http://www-07.ibm.com/systems/includes/content/z/security/pdf/gm130145.pdf>
- ▶ OpenLDAP Suite
<http://www.openldap.org/>
- ▶ Using phpLDAPadmin
<http://phpldapadmin.sourceforge.net>
- ▶ The audispd-zos-remote dispatcher man page
<http://linux.die.net/man/8/audispd-zos-remote>

- ▶ PAM documentation
<http://www.kernel.org/pub/linux/libs/pam/index.html>
- ▶ Linux PAM library
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html
- ▶ Modules for Novell SUSE Linux on System z
<http://www.novell.com/products/linuxpackages/server10/s390x/pam-modules-32bit.html>
- ▶ Running IPsec with Linux
<http://www.ipsec-howto.org>
- ▶ Using openCryptoki on Linux
<http://www.ibm.com/developerworks/linux/library/s-pkcs/>
- ▶ GSKit
<http://www.ibm.com/developerworks/tivoli/library/t-gsk7/>
- ▶ IBM Java PKCS#11 implementation
<http://www.ibm.com/developerworks/java/jdk/security/50/secguides/pkcs11implDocs/IBMJavaPKCS11ImplementationProvider.html>
- ▶ IBM Proventia Server Intrusion Prevention System (IPS) for Linux
http://www-935.ibm.com/services/us/iss/pdf/proventia_server_ips_for_linux_datasheet.pdf
- ▶ Using fwbuilder tool
<http://www.fwbuilder.org>
- ▶ Using shorewall tool
<http://www.shorewall.net>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

3083.P00 VDEV (3083 Controller) 207
30A3.P00 VDEV (30A3 Controller) 207

A

access control 97, 197, 200, 211
administrator-definable policy 104
allocating ID numbers problem 67
AppArmor 105, 108, 225
 access 107
 Control Panel 106
 disable 106
 event 108
 Profile Dialog window 109
 profile entry 109
 tool 108
application programming interface (API) 109
audit 29
AUDITOR attribute 31
auditor-controlled logging 31
auth 110
authentication 97, 225
Authorized usersids 208

B

backchannel communication 219
binary policy 99
boot.apparmor 105

C

CAPP x
central electronic complex (CEC) 215
central processor
 See CP
Central Processor Assist for Cryptographic Function
 See CPACF
centralized user repository 228
certificate authority (CA) 14–15
CEX2A, Crypto Express2, accelerator mode 118
CEX3A, Crypto Express3, accelerator mode 118
chains 167
class G user 219
clear key cryptography, introduction 118
Common Information Model (CIM) 105
Compatible Disk Layout (CDL) 178
configuration file 98–99, 112
 application entry 110
continuous development 205
control_flag field 111
 valid values 111
CP 199
 native 211

 privilege classes 24
CPACF 2, 205
 features 118
Crypto Express, accelerator mode 120
Crypto Express2, accelerator mode, CEX2A 118
Crypto Express3, accelerator mode, CEX3A 118
CTC interface 219

D

DAC 98, 220, 223, 225
DASD 0300 204
data-link layer 211
Deep Thought ix
default configuration 114, 206
default VLAN ID 213
devices C200-C202 224
Directory Information Tree (DIT) 56
DirMaint 6
discretionary access control
 See DAC
disk-based encryption 205
 additional information 205
dm-crypt 205
 introduction and setup 132
 subsystem 205
Douglas Adams ix
DSA key pair 226

E

e2label command 102
EAL5 x
EAL-5 Common Criteria x
e-Business form 232
eCryptfs, introduction and setup 129
encrypted file systems 204
encryption
 deadlock 204
 general definition 204
error
 cannot add object to LDAP 78
ESDM 199
ESM 4
 database 209
etc/pam.d directory 111
etc/pam.d/other file 114
etc/pam.d/passwd file 112
etc/pam.d/sshd file 115
etc/pam.d/system-auth file 113
etc/passwd file 115
etc/security/selinux directory 104
etc/selinux/config file 101
etc/selinux/targeted/policy directory 101
etc/shadow file 228
etc/zipl.conf file 104

- event LDAP
 - configuration problem 115
- example change 113
- example LNXSU1 208
- existence 229
- ext2 file system 204
- external ACL 215
- external security manager
 - See ESM
- EXTRANET INTRANET
 - security level data partition 221

F

- FBA Emulation 179
- file system relabel 102
- file_contexts 101
- firewall
 - reducing load on 215
 - where to host 161
- Forty-two ix
- ftp 0, sample login 103
- ftp_home_dir boolean 103
- ftp_home_dir, enable 103
- fwbuilder, introduction 170

G

- GDBM 50
- getenforce command 100
- grep 0.0.0300 204
- groups 7
 - problem adding 75
- gskkyman 15

H

- HiperSockets 162, 205
- HMC 199
- hypervisor 198
- hypothetical scenario 220

I

- I/O device 99
- I/O lock 203
- IBM ISS Proventia Server Intrusion Prevention System (IPS) 164
- IBM Tivoli Access Manager
 - WebSEAL 233
- ICTX 50
- ID numbers
 - problem allocating 67
- image =/boot/vmlinuz-2.6.18-164.el5 104
- image-to-image
 - traffic 215
- image-to-imagime
 - communication 215
- industry-standard management 105
- intruder 228
- IP address 214
- IP layer 211

- IPSec
 - introduction and setup 134
 - sample transport mode configuration 135
- iptables tool 166
- IUCV
 - allow 26
 - authorization 219
 - recommendation 219

J

- journaling file systems 204

L

- last login time
 - ssh-agent 227
- Layer2 VSWITCH 206
- LDAP 235
 - client 97
 - server 115, 228, 232
 - server, direct port 232
- LDAP Data Interchange Format (LDIF) 56
- LDBM 50
 - extending schema 54
- libica package
 - icainfo output 119
- libselinux
 - enable SELinux 105
- Lightweight Directory Access Protocol
 - See LDAP
- Linux 97, 197, 231
 - access data from inside 200
 - administrator 199
 - client 229
 - configure system for authentication 115
 - controlling I/O lock on system 203
 - distribution 98, 205
 - environment 232
 - firewall, tools 169
 - function 205
 - guest 199
 - kernel 98
 - accelerated functions 126
 - level 198
 - login 116
 - server 105, 198
 - high security clearance 219
 - user 216, 232
 - ID 228
 - RACF user definitions 228
- Linux 5.4 99
- LNXRH1 Group 214
- LNXRH1 link 203
- LNXRH2 Group 214
- LNXSU1 Group 210
- LNXSU1.201 Class 202
- LNXSU1.201 profile 201
- LNXSU1.300 Class 203
- LNXSU2 Group 214
- logging

- owner-controlled 30
- login
 - access 113
 - device 110
 - disable 114
 - process 112
 - program 110
 - session 113
- LPAR capability 235
- LSPP x
- lvm group 203

M

- main assets 229
- MAINT 230
- mandatory access control (MAC) 98, 220
- Maxconn
 - VSWITCH configuration output 207
- MDISK event 200
- mingetty program 116
- misconfiguration 215
- multilevel security 162
- multiple organizations
 - separation of duties 230
- multiple servers
 - physical security 198
- multitier 161
- multizone 161

N

- Name Service Switch
 - See NSS
- NATive attribute 213
- native CP 211
- native VLAN 212
- netfilter packet filtering 165
- network daemons 98
- network diagnostics 216
- Network File System (NFS) 202
- network interface card
 - See NIC
- next reboot 104
- NIC 162, 208, 216
 - NIC 0702 216
 - NIC C200 224
 - QDIO 218
- non-tagged traffic 211
- Novell AppArmor
 - See AppArmor
- N-Port ID Virtualization (NPIV) 183
- NSS 139, 228
 - using openCryptoki as PKCS#11 key store 139
- nullok try_first_pass 113

O

- OpenLDAP 56
- operating system 198
- owner-controlled logging 30

P

- package manager 105
- PAM 109, 226
 - API 110
 - configuration files 112
 - documentation 112
 - enabling applications to use 110
 - environment 111
 - framework 109
 - function 112
 - library 109
 - transaction 110
- passwd program 112
- path name based system 105
- path names 105
- PDF reader
 - required for additional materials 237
- permanent loss 204
- physical security 198
- PKCS#11
 - generic token initialization 138
 - introduction 137
- plain text
 - data 204
 - file 209
- Please login 103
- Pluggable Authentication Module
 - See PAM
- policy directory 101
- policy.conf file 104
- PORTTYPE Access 212–213
- prandom, device node 127
- predecessor cryptoloop 205
- privilege class 199
- privilege escalation 115
- problem
 - adding group 75
 - allocating ID numbers 67
- program login 112
- promiscuous mode 216
 - network interface 216
 - QDIO NIC 218
 - virtual device 216
- protocol z/VM 199
- PRT 224
- pseudorandom number generator (PRNG) 127
 - CPACP support 127
- public key 226

Q

- QDIO NIC 218
- QUERY CRYPTO AP, CP command 124
- QueueStorage 207

R

- R/O node 204
- R/W 204
- RACF class 229
- RACF group 210

- access 210
 - list checking 210
- RACF SMF Unload utility 34–35
- RACFADU 35
- ramdisk=/boot/initrd-2.6.18-164.el5.img selinux 104
- RDR 224
- read-only mode 203
- read-write mode 203
 - same disks 203
- reboot command 101
- Redbooks Web site 242
 - Contact us xiv
- region 7
- regulatory compliance 105, 205
- RHEL
 - revision 101
 - version 100
 - select appropriate directory 101
- RHEL 4 99
- RHEL 5 99
 - binary policy modules 99
 - distribution 116
 - system 99
- root account 115
- root user ID 230
- root@lnxrh1 pam 112
- RPM Package Manager 105
- rpmbuild, sample usage 170
- RPMs 105
- RSA key pair 226
- rules, definition of 168
- run levels
 - AppArmor 108
- run sealert 103

S

- same passphrase entry 226
- same subnet sharing 216
- same VSWITCH 214
 - port types 214
- same z/VM 204
- schema 54
 - converting OpenLDAP to LDIF 56
- SDBM 50, 228
- SECLABEL VMMAC 223
- Secure Sockets Layer (SSL) 14
- security
 - label 220–221
 - level 102, 221
 - policy 98, 200, 233
 - requirement 197, 200, 205
 - application servers 206
 - wide variety 226
 - zones 220
- Security-Enhanced Linux
 - See SELinux
- seed value 68
- self-owned minidisk 200
 - LINK command 200
- SELinux 98, 105, 225

- context 100
 - disabled 104
 - disabling 103
 - entry 104
 - error 103
 - expert 104
 - policy 101
 - prints warning 98
 - security policy 100
 - status 99
 - subdirectory 101
 - support 104
 - system 105
- separation of duties 230
- Service Provider Interface (SPI) 110
- sestatus command 99
- SET SECUSER 219
- setenforce 1 example 102
- setsebool command 103
- shadow nullok library 110
- shared disks 202
- shorewall, introduction 174
- SLES 11 109
- software emulation 205
- SSH access
 - limit by user 114
- sshd file 115
- ssh-rsa
 - key example 226
- SSL 14
- static file system 203
- subnets 206
- subsequent IPL 219
- SYSADMIN 230
- system administrator 109, 220
 - account 115
 - responsibility 222
- system programmer 198
- system resource 97, 209
- System Service 106, 108
- SYSTEM VSWITCH3 207
- System z 97, 99, 197, 231
 - cryptographic solution 235
 - environment 231
 - Novell SUSE Linux 110, 242
 - physical and infrastructure security 198
 - platform 198, 200
 - server 227–228
- SYSTEM.VSWITCH3
 - Class 214
 - ID 209
 - profile 222
- system-auth file 113
- system-wide parameter 199

T

- T=0.01/0.01 time 224
- table
 - filter 167
 - mangle 167

- NAT 167
- raw 167
- security 167
- target =/boot 104
- TCP/IP 214
- Telnet communication 199
- Tivoli Access Manager 232
 - authentication using RACF 232
 - WebSEAL 233
- Tivoli Access Manager for Linux
 - user authentication 232
- Transport Layer Security (TLS) 14

U

- udev, sample rule for PRNG device 128
- UID 0 root account 115
- user
 - login 112
 - management 228
 - password 112, 232
- user directory
 - defining cryptographic devices 123
- user LNXSU1
 - minidisk 202
 - privilege 207
- userland 130

V

- virtual guest 204, 207
 - diminished authority 219
- virtual machine 216
- virtual switch 205
 - base profile 218–219
 - detailed status 218
 - discrete profile 224
 - guest ports 215
 - isolation 228
 - port 215
 - port isolation 214
 - profile 209
 - VLAN-ID qualified profiles 213
 - VSWITCH3 214, 216
- VLAN
 - 0003 213
 - assign multiple 209
 - counter 212
 - native 212
 - tagging 210
- VLAN0003 group 214
- VLAN-aware 211
 - VSWITCH 211
- VLAN-ID qualified profile 213
- VLAN-unaware 207
 - guest 211
 - VSWITCH 207
- VM level 202
 - unauthorized access 202
- VMLAN 209
 - class 209

- limit 219
- VMMAC class 223
- VMMDISK
 - RACF class 200
- VMMDISK class 200
 - authorization check 200
- VNC server, sample command line 171
- VSFTPD
 - access 103
 - interact 103
 - service 103
 - vsFTPd 2.0.5 sample FTP login 103
- VSWITCH 206–207
 - attributes for defining 211
 - creation 213
 - definition 211
 - available options 211
 - name 213
 - VLAN-aware 213
 - z/VM virtual switch 206

W

- Web material 237
- Web-based resource 233
 - security manager 233
- WebSEAL
 - Protecting Web-based resources 233

Y

- Yet another Setup Tool (YaST) 83

Z

- z/OS
 - features 231
 - LPAR 232
 - server 232
- z/OS 1.10 232
- z/VM 115
 - certification authority 15
 - further information 200
 - integrity statement 160
 - LDAP server 115, 228, 232
 - minidisk 200
 - privilege class 199
 - Telnet Server 14
 - users passwords 198
 - V5.4.0 224
- z/VSE 235
 - connector 235
 - environment 235
- z90crypt
 - driver statistics 150
 - loading 124
- zipl menu 104



Redbooks

Security for Linux on System z

(0.5" spine)
0.475" x 0.873"
250 <-> 459 pages



Security for Linux on System z



Securing the System z infrastructure

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.

Securing z/VM

Securing Linux guests

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.

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.

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.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7728-00

ISBN 0738433713