Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security

- Understand CS for z/OS TCP/IP security and policy capabilities
- See CS for z/OS security and policy implementation examples
- Protect your z/OS networking environment

Bill White
Adi Horowitz
Rudi Van Niekerk

Redbooks

ibm.com/redbooks

International Technical Support Organization

Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security

January 2007

SG24-7342-00

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

First Edition (January 2007)

This edition applies to Version 1, Release 8, of z/OS Communications Server.

© Copyright International Business Machines Corporation 2007. 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 ix
Trademarks x

Preface xi
Our implementation environment xii
The environment used for all four books xii
Our focus for this book xiv
The team that wrote this redbook xiv
Become a published author xv
Comments welcome xv

Part 1. Policy-based networking 1

Chapter 1. Policy Agent 3
1.1 Policy Agent description 4
1.1.1 Basic concepts 4
1.1.2 The policy model 6
1.2 Implementing PAGENT on z/OS 9
1.2.1 Starting PAGENT as started task 9
1.2.2 Starting PAGENT from UNIX 13
1.2.3 Stopping PAGENT 14
1.2.4 How to disable PAGENT policies for IPSec 14
1.2.5 Basic configuration 14
1.2.6 Coding policy definitions in a configuration file 17
1.2.7 Refreshing policies 18
1.2.8 Verification 18
1.2.9 For additional information 19
1.3 Setting up TRMD 19

Chapter 2. IP filtering 21
2.1 Defining IP filtering 22
2.1.1 Basic concepts 22
2.1.2 For additional information 24
2.2 Why IP filtering is important 24
2.3 How IP filtering is implemented 25
2.3.1 z/OS IP filtering implementation 25
2.3.2 FTP and Telnet IP filtering scenario 29
2.3.3 Verification 72

Chapter 3. IP Security 75
3.1 IPSec definition 76
3.2 Key IPSec components 76
3.2.1 IP Authentication Header protocol 77
3.2.2 IP Encapsulating Security Payload protocol 77
3.2.3 Internet Key Exchange protocol 77
3.3 How IPSec is implemented 78
3.3.1 Set up the Internet Key Exchange daemon 78
3.3.2 Set up the system logging daemon to log IKED messages 85
3.3.3 Start IKE daemon and verify it initializes 85

© Copyright IBM Corp. 2007. All rights reserved.

3.3.4 Set up Traffic Regulation Manager daemon 85
3.3.5 Update the TCP/IP stack to activate IPSec. 86
3.3.6 Restrict the use of the ipsec command 86
3.3.7 Install the Policy Agent 86
3.3.8 Define the IPSec policies to PAGENT 87
3.3.9 Commands used to administer IP security 87
3.3.10 Using the IBM configuration Assistant for z/OS 87
3.4 Configuring quick dynamic tunnels 88
3.5 Implementing IPSec between two z/OS systems 95
3.5.1 Setting up the policy using IBM Configuration Assistant. 95
3.5.2 Verification of IPSec between two z/OS images 126
3.6 Implementing IPSec between z/OS and Windows 143
3.6.1 Setting up the IKE daemon 143
3.6.2 Setting up the certificates 144
3.6.3 Setting up the z/OS IPSec policy 146
3.6.4 Setting up a Windows IPSec policy. 162
3.6.5 Verifying that things are working 176
3.7 AES cryptographic support for integrated IPSec/VPN 180
3.7.1 Coexistence requirements 180
3.7.2 Enabling AES encryption for IP security tunnels. 180
3.7.3 Verifying that AES is used in IP security 184
3.8 IPv6 support for integrated IPSec/VPN 186
3.8.1 Restrictions 187
3.8.2 Implementing IP Security over IPv6 187
3.8.3 Enabling IP Security for IPv4 188
3.8.4 Defining IPSec policies for IPv6 188
3.8.5 Displaying IPSec IPv6 policies from Policy Agent. 197
3.8.6 Configuring TCP/IP profile for IP Security for IPv6 197
3.8.7 Displaying whether IP Security for IPv6 is active in the TCP/IP stack 198
3.8.8 Displaying the security class configured for each IPv6 interface 198
3.8.9 Displaying IPv6 filter rules. 199
3.8.10 Displaying IPv6 tunnels. 199
3.8.11 Displaying filters that apply to traffic between two IPv6 addresses. 201
3.8.12 Retrieving the new SNMP MIB objects 203
3.9 Further information 203

Chapter 4. Network Address Translation support 205
4.1 Network Address Translation 206
4.2 IPSec and NAT incompatibles 208
4.3 Network address port translation traversal support for integrated IPSec/VPN 208
4.3.1 Enabling NAPT traversal support for IPSec 209
4.3.2 Testing and verification 218

Chapter 5. Application Transparent Transport Layer Security 223
5.1 AT-TLS definition 224
5.1.1 Basic concepts 224
5.1.2 AT-TLS application types 225
5.1.3 For additional information 226
5.2 Why AT-TLS is important 226
5.3 Recommendations 226
5.4 Restrictions 227
5.5 How AT-TLS is implemented 227
5.5.1 Rexx socket application scenario 228

Chapter 6. Intrusion Detection Services 249
6.1 What IDS is 250
6.2 Basic concepts 251
6.2.1 Scan policies 251
6.2.2 Attack policies 255
6.2.3 Attack policy tracing 257
6.2.4 Traffic Regulation policies 257
6.3 How IDS is implemented 259
6.3.1 Installing the Policy Agent 260
6.3.2 The IBM Configuration Assistant for z/OS Communication Server 260
6.3.3 Requirements and download instructions 261
6.3.4 Configuring IDS policy using the GUI 261
6.3.5 Installing the IDS policy 275
6.3.6 Checking that things are working 277
6.3.7 Additional information 278

Chapter 7. Quality of Service 279
7.1 QoS definition 280
7.1.1 Differentiated Services 280
7.1.2 QoS with z/OS Communications Server 282
7.1.3 PAGENT QoS policies 283
7.1.4 Configuring QoS in the z/OS Communications Server 284
7.1.5 For additional information 286
7.2 Why QoS is important 286
7.3 Install the Policy Agent 286
7.4 Using the IBM Configuration Assistant for z/OS Communication Server 286
7.4.1 z/OS host information 288
7.4.2 QoS policy rules 293
7.4.3 Problem determination 301

Part 2. SAF-based security 303

Chapter 8. RACF demystified 305
8.1 Basic concepts 306
8.2 How to protect your network resources 308
8.3 How to protect your programs 309
8.4 How to associate a user ID with a started task 310
8.5 How to set up security for daemons in z/OS UNIX 311
8.6 RACF multilevel security for network resources 311
8.6.1 Basic concepts of MLS 311
8.7 Digital certificates in RACF 312
8.8 Further information 312

Chapter 9. Protecting network resources 313
9.1 The SERVAUTH resource class 314
9.2 Protecting your TCP/IP stack 314
9.2.1 Stack access overview 314
9.2.2 Example setup 314
9.3 Protecting your network access 315
9.3.1 Network access control overview 315
9.3.2 Server considerations 316
9.3.3 Using NETSTAT for network access control 316
9.3.4 Working example of network access control 317
9.4 Protecting your network ports 318

9.4.1 The PORT/PORTRANGE SAF keyword . . . 319
9.4.2 Using NETSTAT to display Port Access control . . . 320
9.5 Protecting the use of socket options . . . 320
9.5.1 SO_BROADCAST Socket option access control . . . 320
9.5.2 IPv6 advanced socket API options . . . 321
9.6 Protecting sensitive network commands . . . 321
9.6.1 z/OS VARY TCPIP command security . . . 322
9.6.2 TSO NETSTAT and UNIX onetstat command security . . . 325
9.6.3 Policy Agent command security . . . 327
9.6.4 IPSec command access control . . . 328
9.6.5 For more information . . . 328
9.7 Protecting FTP-related resources . . . 328
9.7.1 FTP SITE command control . . . 328
9.7.2 FTP server access control . . . 328
9.7.3 FTP z/OS UNIX access control . . . 328
9.7.4 RACF-delegation of cryptographic resources . . . 329
9.8 Protecting network management resources . . . 329
9.8.1 SNMP agent control . . . 329
9.8.2 TCP connection information service access control . . . 329
9.8.3 CIM provider access control . . . 329
9.9 Protecting miscellaneous resources . . . 329
9.9.1 Digital Certificate Access Server access control . . . 329
9.9.2 MODDVIPA utility program control . . . 330
9.9.3 Fast Response Cache Accelerator access control . . . 330
9.9.4 Real-time SMF information service access control . . . 330
9.9.5 TCP/IP packet trace service access control . . . 330
9.9.6 TCP/IP stack initialization access control . . . 330

Part 3. Appendixes . . . 331

Appendix A. Basic cryptography . . . 333
Potential problems with electronic message exchange . . . 334
The request is not really from your client . . . 335
The order could have been intercepted and read . . . 335
The order could have been intercepted and altered . . . 336
An order is received from your client, but he denies sending it . . . 337
Secret key cryptography . . . 337
Public key cryptography . . . 339
Encryption . . . 339
Authentication . . . 340
Public key algorithms . . . 340
Digital certificates . . . 341
Performance issues of cryptosystems . . . 345
Message integrity . . . 346
Message digest (or hash) . . . 346
Message authentication codes . . . 347
Digital signatures . . . 348

Appendix B. Tools for application security . . . 351
Secure Sockets Layer . . . 352
SSL protocol description . . . 352
Certificates for SSL . . . 354
System SSL . . . 356
TLS protocol . . . 357

Kerberos-based security system 358
Kerberos protocol overview. 358
Inter-realm operation. 363
Some assumptions 363
Kerberos implementation in z/OS 364

Appendix C. Certificate management in z/OS 369
Digital certificates 370
How to generate digital certificates in z/OS 371
Digital certificate field formats 372
RACF RACDCERT command use 374
RACF key rings 375
RACDCERT command security 376
RACDCERT command format 376
gskkyman command use 377
Client certificates 379
Server certificates 379
Self-signed certificates 380
Obtaining certificates 380
Self-signed certificates 380
Internal Certificate Authority 396
External Certificate Authority. 401
Certificate locations example 413
RACF certificates 413
gskkyman z/OS UNIX certificates 416

Appendix D. IPSec scenario policies 421

Related publications 431
IBM Redbooks 431
Other publications 431
Online resources 433
How to get IBM Redbooks 433
Help from IBM 433

Index 435

Notices

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

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

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

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

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

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

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

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

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

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

© Copyright IBM Corp. 2007. All rights reserved.

Trademarks

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

CICS®
DB2®
eServer™
HiperSockets™
IBM®
MVS™
NetView®
OS/390®
Parallel Sysplex®
RACF®
Redbooks (logo) ™
Redbooks™
SecureWay®
System z9™
System z™
System/360™
Tivoli®
z/OS®
z9™
zSeries®

The following terms are trademarks of other companies:

Internet Explorer, Microsoft, Windows, 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

For more than 40 years, IBM® mainframes have supported an extraordinary portion of the world's computing work, providing centralized corporate databases and mission-critical enterprise-wide applications. The IBM System z9™, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360™ heritage. Likewise, its IBM z/OS® operating system is far superior to its predecessors—providing, among many other capabilities, world-class, state-of-the-art, support for the TCP/IP Internet protocol suite.

TCP/IP is a large and evolving collection of communication protocols managed by the Internet Engineering Task Force (IETF), an open, volunteer organization. Because of its openness, the TCP/IP protocol suite has become the foundation for the set of technologies that form the basis of the Internet. The convergence of IBM mainframe capabilities with Internet technology, connectivity, and standards (particularly TCP/IP) is dramatically changing the face of information technology and driving requirements for even more secure, scalable, and highly available mainframe TCP/IP implementations.

This Communications Server (CS) for z/OS TCP/IP Implementation series provides easy-to-understand, step-by-step how-to guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP. *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-based Network Security*, SG24-7342, explains how to set up security for your z/OS networking environment. With the advent of TCP/IP and the Internet, network security requirements have become more stringent and complex. Because many transactions come from untrusted networks such as the Internet, and from unknown users, careful attention must be given to host and user authentication, data privacy, data origin authentication, and data integrity. In addition, there are certain applications shipped with TCP/IP such as File Transfer Protocol (FTP) that, without proper configuration and access controls in place, could allow unauthorized users access to system resources and data. The z/OS V1R8.0 Communications Server, along with other elements of z/OS, provides integrated IP security functions to address these TCP/IP security concerns, including:

- Protecting data privacy and integrity while in the network

  The Communications Server protects data in the network using secure protocols based on cryptography, such as IP Security (IPSec), Secure Sockets Layer (SSL), and Transport Layer Security (TLS). IPSec and TLS implementations with z/OS Communications Server are shown in Chapter 3, “IP Security” on page 75, and Chapter 5, “Application Transparent Transport Layer Security” on page 223, respectively.

> **Note:** This book is focused on providing CS for z/OS TCP/IP implementation guidance. Just in case you are unfamiliar with cryptographic technologies, we have also included technology overviews for cryptography, IPSec, SSL, and TLS in Appendix A, “Basic cryptography” on page 333, and Appendix B, “Tools for application security” on page 351.

© Copyright IBM Corp. 2007. All rights reserved.

- Protecting the system from the network

  The Communications Server is responsible for protecting the system against denial-of-service attacks from the network. It has built-in defenses and also provides several services that an installation can optionally deploy to protect against these attacks. Chapter 2, “IP filtering” on page 21, and Chapter 6, “Intrusion Detection Services” on page 249, provide implementation examples for those technologies.

- Protecting system resources and data from unauthorized access

  Communications Server applications and the TCP/IP protocol stack protect data and resources on the system using standard Security Access Facility (SAF)-based services. Part 2, “SAF-based security” on page 303, provides high-level discussion and implementation details regarding the IBM Resource Access Control Facility (RACF®) product.

Finally, given that security technologies are complex and can be confusing, we have included helpful tutorial information in the appendixes of this book.

For more specific information about CS for z/OS base functions, standard applications, and high availability, see the other volumes in the series. These are:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7340
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341

Our implementation environment

We updated the four books in the *Communications Server for z/OS V1R8 Implementation* guides at the same time. Given the complexity of our test environment, we needed to be somewhat creative in organizing the environment so that each team could work with minimal coordination with (and interference from) the other teams.

The environment used for all four books

To enable concurrent work on each of the four books, we set up and shared the test environment illustrated in Figure 1.

Figure 1 Our implementation environment

Our books were updated (and implementation scenarios executed) using three logical partitions (LPARs) on an IBM System z9-109 (referred to as A23, A24, and A25). Because we were working on four books at the same time, we implemented *four* TCP/IP stacks on each LPAR (admittedly, a configuration not recommended for a production environment, but convenient for our purposes). Each LPAR shared:

- IBM HiperSockets™ connectivity
- Coupling Facility connectivity (CF38 and CF39) for parallel sysplex scenarios
- Four OSA-Express2 1000BASE-T Ethernet ports cross-connected to a pair of Cisco 6509 switches

Finally, we shared 10 workstations, representing corporate network access to the z/OS networking environment, for scenario verification (using applications such as TN3270 and FTP).

Our IP addressing convention is as follows:

- The first octet is the network (always 10 for our environment).
- The second octet is the virtual local area network (VLAN (10,20,30,40)) assigned to the stack. (Essentially, except when required by a specific implementation scenario, each team's stacks shared a common VLAN.)
- The third octet refers to the device:
  - The addresses with the third octet of 2 or 3 are defined to the OSA devices.
  - The addresses with the third octet of 4, 5, or 6 are defined to the Hipersocket devices.

Important: Our use of multiple TCP/IP stacks on each LPAR and our TCP/IP addressing were set up for our convenience and are not recommended approaches for your environment.

Our focus for this book

Given the above actual environment, the (simplified) environment that we had to work with on this book is illustrated in Figure 2.

Figure 2 Our environment for this book

For the purposes of this book, then, we viewed the environment as three LPARs leveraging coupling facilities, HiperSockets, and OSA connectivity as required for our implementation scenarios.

The team that wrote this redbook

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

Bill White is a Project Leader and Senior Networking Specialist at the ITSO, Poughkeepsie Center.

Adi Horowitz is an IT specialist who has worked for the past eight years in enterprises that require intensive computing resources, such as the Israel Defence Forces Data Center and Tel-Aviv University. Her areas of expertise include z/OS, Linux® on IBM System z™, and IBM Communications Server for z/OS. She is well-versed with other operating systems and products, specializing in communication products and information systems.

Rudi Van Niekerk is a Consulting IT Specialist and IBM Certified Professional who works for Network and Connectivity Services in South Africa. He has worked with IBM mainframes

since 1985. His areas of expertise include IBM Communications Server (VTAM/APPN, TCP/IP), IBM Parallel Sysplex®, OSA-Express, and SNA-IP migration. He also co-authored other IBM Redbooks™, such as *SNA in a Parallel Sysplex Environment*, SG24-2113 and *OSA-Express Implementation Guide*, SG24-5948.

Thanks to the following people from the ITSO, Poughkeepsie Center, for their contributions to this project: David Bennin, Rich Conway, Mike Connolly, and most importantly, Bob Haimowitz and Greg Geiselhart.

As is always the case with any complex technical effort, success would not have been possible without the advice, support, and review of many outstanding technical professionals. We are especially grateful for the significant expertise and contributions of content to this book from the Communications Server for z/OS Development team, especially from Alfred Christensen, Mike Fox, Hain Shin, and Mark Wright.

Finally, we would like to thank the authors of the previous *Communications Server (CS) for z/OS TCP/IP Implementation* series of IBM Redbooks for creating the groundwork for this series of CS redbooks: Rama Ayyar, Valirio Braga, Octavio Ferreira, Michael Jensen, Sherwin Lake, Bob Louden, Garth Madella, Joel Porterie, Marc Price, Larry Templeton, and Thomas Wienert.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll 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'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Part 1

Policy-based networking

For as long as Information Technology (IT) components have been shared, we have needed to make (and enforce) choices as to who is allowed to access specific IT applications and which applications get priority over others. Historically, such choices have been implemented individually for each component, for example:

- Setting up application access and task priorities in each computer system (such as using ftp.data and telnetglobals for security-specific configuration of File Transfer Protocol (FTP) and Telnet, respectively)
- Defining network traffic priorities in each piece of network equipment

Ultimately, however, those application and traffic decisions must be determined by business priorities, also called *business policies*.

As the complexity of IT environments has increased, it has become increasingly difficult to configure each system and network component individually, and yet still ensure that the overall system usage (and especially the network) matches the required business policies. Consequently, policy-based networking has emerged as a standards-based approach for defining policies in one place and applying them broadly across the entire IT environment.

In this part, we discuss the z/OS Communications Server Policy Agent (“Policy Agent” on page 3) and its use in defining:

- “IP filtering” on page 21
- “IP Security” on page 75
- “Application Transparent Transport Layer Security” on page 223
- “Intrusion Detection Services” on page 249
- “Quality of Service” on page 279

© Copyright IBM Corp. 2007. All rights reserved.

Also starting with z/OS Communications Server V1R8, a graphical interface (GUI) called the IBM Configuration Assistant (also referred to as Configuration Assistant) is provided.

The Configuration Assistant GUI provides an alternative to manually editing configuration files, and it assists you in the configuration of the following z/OS Communications Server TCP/IP functions:

- Quality of Service (QoS)
- Intrusion Detection Services (IDS)
- Application Transparent TLS (AT-TLS)
- IP Security (IPSec)

The Configuration Assistant GUI replace the GUIs that were previously provided as Web downloads for configuring the following functions:

- zQoS Manager for configuring QoS
- zIDS Manager for configuring IDS
- Network Security Configuration Assistant (NCSA) for configuring AT-TLS and IPSec.

Configuration files created by these previously existing GUIs can be imported into the IBM Configuration Assistant GUI, which makes migration easy.

To download the GUI and for additional information, see the IBM Communications Server product support Web site:

http://www.ibm.com/software/network/commserver/zos/support/

1

Policy Agent

The Policy Agent (PAGENT) is a component within server platforms that is responsible for implementing policy decisions. This chapter focuses on the z/OS Communications Server Policy Agent and its related security functions. Policy Agent enforces a set of rules and policies that dictate how users, applications, and organizations can access and use IT resources. We show how policy-based networking introduces a centralized policy storage approach, and how it interacts with other security functions.

Attention: The current release of PAGENT allows policies to be read from configuration files, Lightweight Directory Access Protocol (LDAP) server, or both. For the purpose of this book and the fact that z/OS Communications Server V1R8 is planned to be the last release to support LDAP version 2, we have exclusively made use of the graphical user interface (GUI) called IBM Configuration Assistant for z/OS Communications Server to create our configuration files.

This chapter discusses the topics listed in Table 1-1.

Table 1-1 Topics discussed in this chapter

Section	Topics
1.1, “Policy Agent description” on page 4	We discuss the idea behind using Policy Agent as a central repository for policies. We also discuss basic concepts of the Policy Agent.
1.2, “Implementing PAGENT on z/OS” on page 9	In this section, we show a basic Policy Agent installation.
1.3, “Setting up TRMD” on page 19	This section describes how to set up Traffic Regulation Monitoring daemon (TRMD) to work in conjunction with Intrusion Detection Services (IDS) to handle Policy Agent messages.

© Copyright IBM Corp. 2007. All rights reserved.

1.1 Policy Agent description

As is illustrated in Figure 1-1, the z/OS Communications Server PAGENT implements policy-based networking for the z/OS environment.

Figure 1-1 Policy-based networking and PAGENT

Policy definitions are contained in local configuration flat files. The policies are used to control network security, traffic prioritization, bandwidth management, network behavior, and resource balancing for the z/OS environment. The Policy Agent reads these configuration files, parses the policies, and stores the policy definitions in the TCP/IP stack. The policies are then acted on by a TCP/IP stack.

The policies supported by PAGENT are discussed in detail in the subsequent chapters as follows:

- Chapter 2, “IP filtering” on page 21
- Chapter 3, “IP Security” on page 75
- Chapter 5, “Application Transparent Transport Layer Security” on page 223
- Chapter 6, “Intrusion Detection Services” on page 249
- Chapter 7, “Quality of Service” on page 279

1.1.1 Basic concepts

Architecturally, policy-based networking typically involves the following components:

- A policy management service: A graphical user interface (GUI) for specifying, editing, and administering policies, such as the new Configuration Assistant.
- Policy repository: A place to store and retrieve policy information, such as configuration file.
- Policy decision point (PDP): A resource manager or policy server that is responsible for handling events and making decisions based on those events, and updating the Policy Enforcement Point configurations appropriately. Policy Agent (PAGENT) is our PDP, as shown in Figure 1-2 on page 5.
- Policy enforcement point (PEP): A PEP exists in network nodes such as routers, firewalls, and hosts. It enforces the policies based on the “if condition then action” rule sets it has received from the PDP. In Figure 1-2 on page 5, the TCP/IP stack serves as our PEP.

Figure 1-2 Policy system model

Where and how we define policies

Policies are defined in the Policy Agent configuration file, as shown in Figure 1-3. You should keep policy names unique, as policy objects with duplicate names run the risk of being deleted by PAGENT.

Figure 1-3 Configuring Policy Agent

The following PAGENT policies can be stored in a configuration text file format:

- Quality of Service (QoS) policies
- IDS policies
- IP Security (IPSec) Virtual Private Network (VPN) policies
- IP filter policies
- Application Transparent TLS (AT-TLS) policies
- Sysplex Distributor policies
- Traffic regulation policies

To aid in setting up policies, use the IBM Configuration Assistant for z/OS Communications Server. It can build the configuration flat file for IDS, QoS, IPSec, and AT-TLS policies.

This GUI is available for download from the Web at:

http://www-1.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=D400&uid=swg24013160&loc=en_US&cs=UTF-8&lang=en

1.1.2 The policy model

Service policies consist of policy rules and policy actions. When the policy rule is true, one set of actions is initiated, and when false, a different set of actions is initiated. The policy rule is the condition. Conditions can involve both time specifications and traffic filters; however, if both are used, then both have to match for the condition to be true. The policy action is the action to be performed. To learn more about Policy Agent rules, refer to the *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.

An example of a policy rule and action statement

The example policy definition in Figure 1-4 on page 7 causes the stack to discard all Telnet requests from subnet 10.12.4.224 to 10.12.4.254 on Port 23. Note how we have restricted the range to a single port (23). Yet there might be other Telnet servers at other ports available on the IP stack, including a UNIX® Telnet server. Perhaps, it is not important to block access to those Telnet servers, because you have other security mechanisms in place for them. However, if it is critical to inhibit all attempts to reach any Telnet server on the z/OS systems, you need to include a policy rule for each of them. Policy Agent blocking is not meant to be a replacement for firewall filtering. PAGENT should be considered only as a second or third line of defense for certain types of actions.

Rules that intend to block traffic apply to both inbound and outbound traffic, whether or not the traffic originates at the z/OS Communications Server - TCP/IP component.

For actions specifying a permission of blocked, TCP traffic is handled differently from UDP or RAW traffic depending on whether the connection request is inbound or outbound. When an inbound request is received, an inbound rule is checked to see if the SYN should be accepted. If it is, then the outbound rule is checked to see if the connection is allowed. If both the inbound and outbound rules indicate that it is all right to accept the connection, then a tcb (TCP connection control block) is built. Any other QoS rules, for example, TOS settings or similar, can now be applied to the outbound connection. If the inbound rule permits the connection but the outbound connection does not, the tcb is not built and the connection request is rejected.

If an outbound TCP SYN request is generated and there is no outbound blocking rule in effect, the tcb is created and any outbound QoS rules are applied. Inbound blocking rules are ignored. When the SYN/ACK arrives back at the z/OS Communications Server IP server, a tcb with an assigned outbound QoS already exists, and there is no further checking to see if an inbound blocking rule is in effect.

With a flat file policy definition, the *source* indicates the source of the data flow with *destination* signifying the target of the data traffic. Therefore, in some cases the source is z/OS Communications Server IP and in others the source is the remote host.

Figure 1-4 Policy rule and action statement

The TCP/IP stack receives policies from the following two sources:

- Policy Agent, which has its policies stored in a flat file
- The PROFILE.TCPIP statements:
  - IPCONFIG IPSECURITY SOURCEVIPA
  - IPSEC
  - ipsec rules
  - ENDIPSEC

If using the PROFILE.TCPIP statements without Policy Agent IPSec policies, note the following:

- The IPSEC/ENDIPSEC block statement is ignored, if IPSECURITY is not specified on the IPCONFIG statement.
- Only one IPSEC/ENDIPSEC block statement block should appear in the profile. Any subsequent statement blocks are ignored.
- If you code IPSECURITY IPCONFIG with no IPSEC/ENDIPSEC block statement, only local traffic will flow through the stack (that is, loopback addresses).
- No routing functions are supported by the stack through these statements even with DATAGRAMFWD coded in the PROFILE.TCPIP. Routing function can only be performed based on the policies read in from PAGENT.

Example 1-1 shows the sample shipped in member SAMPPOF of the SEZAINST data set.

Example 1-1 Example IPCONFIG for IPSECURITY support

```
; IPCONFIG IPSECURITY
; ---------------------------------------------------------------------
; Configure IPSECURITY default filter rules
; ---------------------------------------------------------------------
;
; Example IPSEC default filter rule. This rule permits
; outbound TCP traffic from local IP address 1.1.1.1 port 23 to
; remote IP address 2.2.2.2. The same rule also permits
; inbound TCP traffic from remote IP address 2.2.2.2 to local
; IP address 1.1.1.1 port 23.
;
```

```
; IPSEC LOGDISABLE NOLOGIMPLICIT
; Rule   SrcIp   DestIp  Log   Prot      SrcPort    DestPort  Secclass
; IPSECR 1.1.1.1 2.2.2.2 NOLOG PROTO TCP SRCPORT 23 DSTPORT *
; ENDIPSEC
```

Note: TCP/IP does not use both sets of policies simultaneously. It uses the IPSEC policies from the profile when PAGENT is not active and swaps to the PAGENT policies when PAGENT is active.

PAGENT installs two default policy rules, which deny all inbound and outbound traffic. The active policies can be displayed using the **pasearch** command. We displayed our active policies and used the **pasearch** command to establish the presence of the default rules, which can be seen in Example 1-2.

Example 1-2 pasearch command showing the default policies

```
policyRule:             DenyAllRule_Generated___________Inbnd
  Rule Type:            IpFilter
  Version:              3                 Status:            Active
  Weight:               102               ForLoadDist:       False
  Priority:             2                 Sequence Actions:  Don't Care
  No. Policy Action:    0
  IpSecType:            policyIpFilter
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
   Fr TimeOfDay UTC:    04:00             To TimeOfDay UTC:  04:00
   TimeZone:            Local
  IpSec Condition Summary:                NegativeIndicator: Off
   IpFilter Condition:
     FromAddr:          All
     ToAddr:            All
    Destination Address:
     FromAddr:          All
     ToAddr:            All
    Service Condition:
     Protocol:          All
     Direction:         Inbound
     RouteType:         Either            SecurityClass:     0

policyRule:             DenyAllRule_Generated___________Outbnd
  Rule Type:            IpFilter
  Version:              3                 Status:            Active
  Weight:               101               ForLoadDist:       False
  Priority:             1                 Sequence Actions:  Don't Care
  No. Policy Action:    0
  IpSecType:            policyIpFilter
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
```

```
  Start Date Time:     None
  End Date Time:       None
  Fr TimeOfDay:        00:00            To TimeOfDay:      24:00
  Fr TimeOfDay UTC:    04:00            To TimeOfDay UTC:  04:00
  TimeZone:            Local
 IpSec Condition Summary:               NegativeIndicator: Off
  IpFilter Condition:
```

Important: When dealing with policies that deny all traffic, it is imperative to permit traffic to some essential services.

The services listed in Table 1-2 have to be available to the stack.

Table 1-2 Stack services

Service	Direction	Source port	Destination port	Protocol
Resolver	Outbound	53	Any	TCP or UDP
Resolver	Inbound	53	Any	TCP or UDP
Omproute - RIPV1	Outbound	520	520	UDP
Omproute - RIPV1	Inbound	520	520	UDP
Omproute - RIPV2	Outbound	520	520	UDP and IGMP
Omproute	Inbound	520	520	UDP and IGMP
Omproute - OSPF	Outbound			IP and IGMP
Omproute - OSPF	Inbound			IP and IGMP

Even though not listed under essential services, PING using protocol Internet Control Message Protocol (ICMP) can be useful during problem determination.

1.2 Implementing PAGENT on z/OS

On z/OS V1R8.0 Communications Server, the Policy Agent runs as a UNIX process. As such, it can be started either from the UNIX System Services shell or as a started task. We used a started task procedure to start Policy Agent.

1.2.1 Starting PAGENT as started task

The sample started task procedure for PAGENT can be found in TCPIP.SEZAINST(EZAPAGSP). We used the following PAGENT started task procedure on our system (Example 1-3).

Example 1-3 PAGENT started task procedure

```
//PAGENT   PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZAPAGSP
```

```
//*
//* 5694-A01 (C) Copyright IBM Corp. 1998, 2005
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV1R8
//*
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*      PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /
//*            etc/pagent3.conf -l SYSLOGD'
//*
//* Provide environment variables to run with the desired
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   PAGENT_CONFIG_FILE=/etc/pagent2.conf
//*   PAGENT_LOG_FILE=/tmp/pagent2.log
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//*STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.PAGENT.ENV(PAGENT),DISP=SHR
//* Sample z/OS UNIX file containing environment variables:
//STDENV   DD PATH='/etc/pagent.sc30.env',PATHOPTS=(ORDONLY)1
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, PAGENT doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the PAGENT_LOG_FILE environment variable, and defaults to
//* /tmp/pagent.log.  When the -d parameter is specified, however,
//* output is also written to stdout.
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

You can use environment variables either configured in an IBM MVS™ data set or z/OS UNIX file specified by the STDENV DD to run with the required configuration. We have configured our environment variables in a z/OS UNIX file, /etc/pagent.sc30.env, as shown in Example 1-4.

Example 1-4 STDENV data set contents

```
LIBPATH=/lib:/usr/lib:/usr/lpp/ldapclient/lib:.  1
PAGENT_CONFIG_FILE=/SC30/etc/pagent.sc30.conf 2
PAGENT_LOG_FILE=/SC30/tmp/pagent.sc30.log     3
PAGENT_LOG_FILE_CONTROL=300,3                 4
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA
TZ=EST5EDT
```

Note: Ensure that the z/OS UNIX file pointed to by STDENV, as well as the files contained in this STDENV file, has the correct permission bits set to allow PAGENT access to these files.

We configured the following environment variables for the Policy Agent:

- **1** LIBPATH enables PAGENT to search the dynamic link libraries needed to act as an LDAP client.
- **2** PAGENT_CONFIG_FILE specifies the specific PAGENT configuration file to use.
- **3** PAGENT_LOG_FILE specifies the log file name used by PAGENT.
- **4** PAGENT_LOG_FILE_CONTROL defines the number of PAGENT log files and their size in kilobytes. In our case, we requested three log files, each 300 KB in size. These are used in a round-robin fashion. To configure PAGENT to use SYSLOGD to log messages, you can define PAGENT_LOG_FILE=SYSLOGD. In this case, PAGENT_LOG_FILE_CONTROL has no meaning.
- In our case, although we do not have the RESOLVER_CONFIG variable configured, PAGENT establishes an affinity to the proper TCP/IP stack through BPXK_SETIBMOPT_TRANSPORT=TCPIPA. The TcpImage statement in the Policy Agent configuration file also determines to which TCP/IP stack PAGENT will install policies.
- For the Policy Agent to run in your local time zone, you might have to specify the time zone in your working location using the TZ environment variable, even if you have the TZ environment variable configured in /etc/profile.

Note: Most z/OS UNIX applications that start as MVS started tasks cannot use environment variables that have been configured in /etc/profile.

Before we started PAGENT, we defined it with the correct security authorizations.

Defining the security product authorization for PAGENT

Because the Policy Agent can affect system operation significantly, security product authority (for example, RACF) is required to start the Policy Agent from a z/OS procedure library.

To set up the security definitions for PAGENT, perform the following steps:

1. Define the PAGENT started task to RACF.
2. Define a user ID for the PAGENT started task.
3. Associate this user ID with the PAGENT started task.
4. Give authorized users access to start and stop PAGENT.
5. Restrict access to the pasearch command to authorized users.
6. Set up TTLS Stack Initialization access control.

Define the PAGENT started task to RACF

To set up a started task, you have to define a profile for it in the resource class called STARTED. First, you have to activate this class if it is not already active. This resource class is RACLISted so that the profiles are kept in RACF data space for improved performance. It is also defined as a GENERIC class to allow generic profiles to be created in this class for more efficient searches. In most installations, this might already have been done, therefore you might not have to issue commands to define the STARTED class RACLIST and GENERIC. We just mention it here for completeness.

You must specify two qualifiers for the profile names in STARTED class. We defined PAGENT.*, and then we refreshed RACLIST and GENLIST to update the in-storage profiles with this new information. Example 1-5 shows the RACF commands for this.

Example 1-5 Define the PAGENT started task to RACF

```
SETROPTS CLASSACT(STARTED)
SETROPTS RACLIST(STARTED)
SETROPTS GENERIC(STARTED)
RDEFINE  STARTED  PAGENT.*
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED)  REFRESH
```

Define a user ID for the PAGENT started task

We defined a PAGENT user ID with default group TCPGRP and with an OMVS segment.

This user ID has to be defined with UID=0. But only one user ID can have UID=0 in the system and it is normally already assigned to user BPXROOT in most installations. Therefore, you have use the “SHARED” parameter in the definition. A home directory is also assigned to this user ID. Example 1-6 shows the command that we used.

Example 1-6 Define a user ID for the PAGENT started task

```
ADDUSER  PAGENT DFLTGRP(TCPGRP) OMVS(UID(0) SHARED HOME('/'))
```

Associate this user ID with the PAGENT started task

We used the RALTER command to associate the PAGENT user ID and its group TCPGRP to the PAGENT started task. RACF stores this information in the STDATA field of the profile. Example 1-7 shows the command that we used.

Example 1-7 Associate user ID with the PAGENT started task

```
RALTER STARTED PAGENT.* STDATA(USER(PAGENT) GROUP(TCPGRP))
```

Give authorized users access to start and stop PAGENT

To control which users can start PAGENT and thus reduce the risk of an unauthorized user starting and affecting policy-based networking, we define a profile named MVS.SERVMGR.PAGENT in resource class OPERCMDS and give authorized users access to this facility. Activate the OPERCMDS class and RACLIST it if not already done in your installation. Example 1-8 shows the commands that we used.

Example 1-8 Give authorized users access to start and stop PAGENT

```
SETROPTS CLASSACT(OPERCMDS)
SETROPTS RACLIST (OPERCMDS)
RDEFINE  OPERCMDS (MVS.SERVMGR.PAGENT) UACC(NONE)
PERMIT  MVS.SERVMGR.PAGENT CLASS(OPERCMDS) ACCESS(CONTROL) ID(PAGENT,CS01,CS02)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Restrict access to the pasearch command to authorized users

The **`pasearch`** command is used to obtain details of the security policies on your system. You can also enable and disable policies using this command. This is a very sensitive command and needs to be protected. The profile to protect this resource is of the form EZB.PAGENT.sysname.tcpprocname.*, where:

EZB	Constant
PAGENT	Constant for this resource type

sysname	This is the system name
tcpprocname	This is the TCP/IP proc name
*****	This is for all policy type options

The profile is defined in the SERVAUTH class. Example 1-9 shows the commands that we used.

Example 1-9 Restrict access to the pasearch command to authorized users

```
RDEFINE  SERVAUTH EZB.PAGENT.SC30.TCPIPA.* UACC(NONE)
PERMIT   EZB.PAGENT.SC30.TCPIPA.* CLASS(SERVAUTH) ID(PAGENT,CS01,CS02) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

Set up TTLS Stack Initialization access control

If you are using Application Transparent Transport Layer Security (AT-TLS), z/OS will not allow any socket-based applications to start before PAGENT is up and running so as to make sure that all the security policies are enforced. But some essential applications need to start before PAGENT. To allow this, you have to define a resource profile EZB.INITSTACK.sysname.tcpprocname in the SERVAUTH class. The resource name consists of the following parts:

EZB	Constant
INITSTACK	Constant for this resource type
sysname	This is the system name
tcpprocname	This is the TCP/IP proc name

The RACF commands that we used for this are shown in Example 1-10.

Example 1-10 Set up TTLS Stack Initialization access control

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE  SERVAUTH EZB.INITSTACK.SC30.TCPIPA UACC(NONE)
PERMIT   EZB.INITSTACK.SC30.TCPIPA CLASS(SERVAUTH) ID(*) ACCESS(READ) -
          WHEN(PROGRAM(PAGENT,EZAPAGEN))
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

1.2.2 Starting PAGENT from UNIX

The PAGENT executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure that your PATH statement contains either /usr/sbin or /usr/lpp/tcpip/sbin. To start PAGENT in the z/OS UNIX System Services shell, issue the following command:

```
pagent -c /etc/pagent.sc30.conf SYSLOGD &
```

The Policy Agent uses the configuration file /etc/pagent.sc30.conf and logs output to the syslog daemon (SYSLOGD). To run PAGENT in the background, the start command is suffixed with “&”.

Consult the *z/OS V1R8.0 Communications Server: IP Diagnosis Guide*, GC31-8782, to resolve any EZZ errors encountered at PAGENT startup time.

1.2.3 Stopping PAGENT

You can stop the Policy Agent as follows:

- Using the operator command P PAGENT from System Display and Search Facility (SDSF) or the system console.
- Using the **kill** command in the z/OS UNIX shell. Example 1-11 shows how to find the process ID for PAGENT, which is then killed with the **kill -s** command. The pid can also be found in /tmp/pagent.pid.

Example 1-11 Stopping pagent from UNIX

```
CS02 @ SC30:/u/cs02>ps -ef | grep PAGENT
 BPXROOT   16842831   83951672  - 16:00:27 ttyp0001  0:00 grep PAGENT
 BPXROOT   67174676          1  -   Oct 14 ?         1:13 PAGENT
CS02 @ SC30:/u/cs02>kill -s TERM 67174676
CS02 @ SC30:/u/cs02>ps -ef | grep PAGENT
 BPXROOT   33620047   83951672  - 16:01:10 ttyp0001  0:00 grep PAGENT
CS02 @ SC30:/u/cs02>
```

When the Policy Agent is shut down normally (KILL or STOP), if the PURGE option is configured, all QoS, IDS, and AT-TLS policies are purged from this stack. IPSec policies are not automatically purged.

1.2.4 How to disable PAGENT policies for IPSec

PAGENT policies can be disabled by using the following command:

```
ipsec -f default
```

This reverts back to the TCP/IP configuration file policies. To convert back to PAGENT policies, issue the following command:

```
ipsec -f reload
```

1.2.5 Basic configuration

Before defining policies, some basic operational characteristics of the Policy Agent need to be configured in the PAGENT configuration file. In this section, we detail the following configuration steps:

- Define the TcpImage statements.
- Define the appropriate logging level.

Define the TcpImage statements

The Policy Agent can be configured to install policies on one or more TCP/IP stacks or images. Each TCP/IP stack is configured using a TcpImage statement in the main configuration file. A secondary configuration file can be defined for each stack, a set of stacks can share configuration information in the main configuration file, or a combination of these techniques can be used.

To install different sets of policies to different stacks, configure each image with a different secondary configuration file. In this case, each image can be configured with a different policy refresh interval, if required. The refresh interval used for the main configuration file will be the smallest of the values specified for the different stacks.

In Figure 1-5, we show PAGENT's configuration file identifying, through TcpImage statements, the names of the TCP/IP stacks on which policies are to be installed and the different configuration files that should be used by each.

Figure 1-5 Multiple stacks, multiple policy definitions

Note: When the main configuration file is an MVS data set, it is reread at each refresh interval (which is the smallest of the individual stack refresh intervals), regardless of whether it has actually been changed. Because PAGENT restarts all stack-related processing when the main configuration file is reread, this effectively makes the refresh interval for all stacks the same as this smallest configured interval.

To install a common set of policies to a set of stacks on the same LPAR, do not specify secondary configuration files for each image. In this case, there is only one configuration file (the main one) and the policy information contained in it is installed to all of the configured stacks. Different refresh intervals can also be configured for each image, but might probably not be useful in this case.

In Figure 1-6, we show PAGENT's configuration file identifying, through TcpImage statements, the names of the TCP/IP stacks on which policies are to be installed, but in this case installing the same policies into each.

Figure 1-6 Multiple stacks, single policy definition

It is possible that TCP/IP stacks that are configured to the Policy Agent are not started or even defined. The Policy Agent will fail when trying to connect to those stacks and log appropriate error messages.

The Policy Agent does not end when any (or all) stacks end. When the stacks are restarted, active policies are automatically reinstalled. When the Policy Agent is shut down normally (that is, using KILL or STOP commands), and the TcpImage statement option PURGE is coded, all policies will be purged from this stack. The TcpImage statement specifies a TCP/IP image and its associated configuration file to be installed to that image. The following example installs the policy control file /etc/pagent.sc30.conf to the TCPIPA TCP/IP image, after flushing the existing policy control data:

```
TcpImage TCPIPA /etc/pagent.sc30.conf FLUSH
```

Define the appropriate logging level

The LogLevel statement is used to define the amount of information to be logged by the Policy Agent. The default is to log only event, error, console, and warning messages. This might be appropriate for a stable policy configuration, but more information might be required to understand policy processing or debug problems when first setting up policies or when making significant changes. Specify the LogLevel statement with the appropriate logging level in the main configuration file.

Note: The maximum logging level (511) can produce a significant amount of output, especially with large LDAP configurations. This is not of concern if z/OS UNIX log files are used, because Policy Agent will round-robin (circulate) a set of finite size files. (The environmental variable PAGENT_LOG_FILE_CONTROL identifies the number and size of these files.) However, when using the syslog daemon as the log file, the amount of log output produced should be taken into consideration.

Considerations when defining policy rules

When you define and code the policy rules direction, source, and destination, you should consider when policy rules are applied:

- For TCP, the policies are applied at TCP connection set up.
- For UDP, a policy rule is applied every time a UDP datagram is being received or sent.
- For other protocols, such as ICMP, Open Shortest Path First (OSPF), and so on, every time an IP datagram is being received or sent, the policy rules are applied.
- The policies are re-mapped when the policy definitions are being updated or refreshed. The rules are re-mapped for every ACK segment in a TCP flow to adjust for time-of-day related policies.

1.2.6 Coding policy definitions in a configuration file

The example configuration shown in Example 1-12 is based upon the “Multiple stacks, multiple policy definitions” scenario shown in Figure 1-5 on page 15. In this scenario, the policy definitions have been configured in the PAGENT configuration file and we use a two-level PAGENT configuration file to define the policy in a multiple IP stacks environment, as shown in Example 1-12.

Example 1-12 PAGENT configuration file

```
#
#  IBM Communications Server for z/OS
#  SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZAPAGCO
#
#  Licensed Materials - Property of IBM
#  5694-A01
#  (C) Copyright IBM Corp. 1998, 2005
#  Status = CSV1R8
# LogLevel Statement
Loglevel 255 1
# TcpImage Statement 2
TcpImage TCPIPA /etc/sc30.tcpipa_image.conf FLUSH
TcpImage TCPIPB /etc/sc30.tcpipb_image.conf FLUSH
TcpImage TCPIPC /etc/sc30.tcpipc_image.conf FLUSH
```

The log level 1 is set with the integer that specified the level of logging/tracing. We are using LogLevel 255, which means all messages except trace messages are captured. The supported levels are:

- 1 - SYSERR - System error messages
- 2 - OBJERR - Object error messages
- 4 - PROTERR - Protocol error messages
- 16 - EVENT - Event messages

- 32 - ACTION - Action messages
- 64 - INFO - Informational messages
- 128 - ACNTING - Accounting messages
- 256 - TRACE - Trace messages

The TcpImage statement **2** defined the TCP/IP stacks to be policed (TCPIPA, TCPIPB, and TCPIPC). Up to four parameters can be configured for this statement. The first parameter specifies the TCP/IP stack name on which the policy must be installed. The next one is the path of the image configuration file for the associated TCP/IP stack. The third parameter, you can specify whether the Policy Agent deletes all the policies existing in the TCP/IP stack when it is started.

> **Note:** The policies installed in the TCP/IP stack will be deleted at PAGENT startup time *only* if the FLUSH parameter is specified. This prevents the policies from being deleted unexpectedly, if PAGENT terminates abnormally.

If you want to remove policies when you cancel PAGENT, you can restart PAGENT afterward, pointing to a configuration file with FLUSH specified but no policies defined. The last TcpImage statement parameter (not specified in our example) specifies the time interval in seconds for checking the creation or modification time of the configuration files and for refreshing policies from the LDAP server. The default value is 1800 seconds (30 minutes).

> **Note:** Dynamic monitoring for the configuration file is only supported for z/OS UNIX files. MVS data sets are not monitored for changes.

Policy Agent log file

When you start the Policy Agent as a started task, the output messages written to stdout and stderr go to the data set or file specified with SYSPRINT or SYSOUT DD. But, normally, PAGENT does not write output to stdout or stderr. Instead, output is written to the log file, which can be specified by the PAGENT_LOG_FILE environment variable and defaults to /tmp/pagent.sc30.log. When the -d parameter is specified, however, output is also written to stdout. The log file is created when the Policy Agent is activated, if it does not already exist.

1.2.7 Refreshing policies

There are two commands used to refresh policies in PAGENT:

- The F PAGENT,REFRESH command triggers Policy Agent to reread its config files and, if requested, downloads policy objects from the LDAP server. If the FLUSH parameter is specified on the TcpImage configuration statement, policy statistics being collected in the TCP/IP stack are reset, because FLUSH deletes and reinstalls all policies.
- The F PAGENT,UPDATE command is different from the REFRESH command, because PAGENT only installs or removes from the stack (as appropriate) any new, changed, or deleted policies.

1.2.8 Verification

Use the z/OS UNIX **pasearch** command to query information from the z/OS UNIX Policy Agent. The command is issued from the UNIX System Services shell. We used the **pasearch** command to display all the policy entries for our TCP/IP stack named TCPIPA using the following command:

```
pasearch -p TCPIPA
```

The default is to return all policy entries for all TCP/IP stacks. The value used for TcpImage, in our example TCPIPA, must match one of the values specified on the TcpImage statement in the Policy Agent configuration file.

1.2.9 For additional information

Refer to the *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775, for additional information regarding Policy Agent.

1.3 Setting up TRMD

The Traffic Regulation Monitoring daemon, or TRMD, can be viewed as a syslog daemon message writer. TRMD handles syslogd event recording for Intrusion Detection Services (IDS), IPSec services, and traffic regulation.

Setting up the started task procedure

A sample TRMD procedure can be found in TCPIP.SEZAINST(EZATRMDP). To associate the TRMD procedure with our TCP/IP job name, we set the RESOLVER_CONFIG environment variable to point to our TCPIP.DATA file, as shown in Example 1-13. TRMD can be started from the z/OS UNIX shell or as a started task.

Example 1-13 TRMD procedure parameters

```
//TRMD   EXEC PGM=EZATRMD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//  'ENVAR("RESOLVER_CONFIG=//''TCPIP.TCPPARMS(DATAA30)''"',
//  '"LIBPATH=/usr/lib)"/-d 1')
```

To start TRMD as a started task, use the S TRMD command from the MVS console or SDSF. To automatically start TRMD when the TCP/IP stack is started, add TRMD to the AUTOLOG statement in the TCP/IP profile, as shown in Example 1-14.

Example 1-14 Autologging TRMD from TCP/IP

```
AUTOLOG
   TRMD JOBNAME TRMD
ENDAUTOLOG
```

Starting TRMD from z/OS UNIX

Only a superuser can run TRMD from the z/OS UNIX shell. Ensure that the following environment variables are correctly set before starting TRMD:

```
RESOLVER_CONFIG - To determine which stack TRMD will use
TZ - To ensure that the syslogd records are correctly timestamped
```

We set the environment variables **1**, and started **2** and stopped **3** TRMD with the **kill** command, as shown in Example 1-15.

Example 1-15 Starting and stopping TRMD

```
CS02 @ SC30:/u/cs02>su
CS02 @ SC30:/u/cs02>export TZ="EST5EDT" 1
CS02 @ SC30:/u/cs02>echo $TZ
EST5EDT
CS02 @ SC30:/u/cs02>
CS02 @ SC30:/u/cs02>export RESOLVER_CONFIG="//'TCPIPA.TCPPARMS(DATAA30)'" 1
```

```
CS02 @ SC30:/u/cs02>echo $RESOLVER_CONFIG
//'TCPIPA.TCPPARMS(DATAA30)'
CS02 @ SC30:/u/cs02>
CS02 @ SC30:/u/cs02>trmd                                                    2
CS02 @ SC30:/u/cs02>ps -ef | grep trmd
 BPXROOT      65581   83951684  - 11:58:18 ttyp0001  0:00 grep trmd
 BPXROOT      65601          1  - 11:57:58 ttyp0001  0:00 trmd
CS02 @ SC30:/u/cs02>kill -s TERM 65601 3
CS02 @ SC30:/u/cs02>ps -ef | grep trmd
 BPXROOT   16842817   83951684  - 11:59:03 ttyp0001  0:00 grep trmd
CS02 @ SC30:/u/cs02>
```

Defining the security product authorization for TRMD

RACF is required to start the Policy Agent from a z/OS procedure library. First, define a user ID for TRMD with a UID of 0 and define it to the RACF started class list (as shown in Example 1-16).

Example 1-16 RACF definitions for TRMD

```
//RACFDEF EXEC PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
 ADDUSER  TRMD     DFLTGRP(TCPGRP)  OMVS(UID(0) SHARED HOME('/'))
 RDEFINE  STARTED  TRMD.* STDATA(USER(TRMD))
 SETROPTS RACLIST(STARTED) REFRESH
 SETROPTS GENERIC(STARTED) REFRESH
```

TRMDSTAT

Trmdstat is a utility that produces reports from IDS syslog records (summary and detailed). It reads a log file and analyses the log records from TRMD. The following reports are available:

- Overall summary of logged connection events
- IDS summary of logged events
- Reports of logged connection events
- Reports of logged intrusions defined in the ATTACK policy
- Reports of logged intrusions defined in the TCP policy
- Reports of logged intrusions defined in the UDP policy
- Reports of statistics events

2

IP filtering

IP filtering provides a means of *permitting* or *denying* IP messages into and out of the z/OS Communications Server environment at a very early stage in message handling (and therefore, very efficiently).

Note: One thing that can be confusing is that z/OS Communications Server IP filtering support has been packaged together with IP Security (IPSec) and is referred to as integrated IP Security. That is because there is a very close affinity between IPSec and IP filtering in the z/OS Communications Server; although you can implement IP filtering without IPSec, you cannot implement IPSec without IP filtering. Consequently, you will notice that, in order to configure IP filtering, you have to indicate that you are configuring IPSec in the IBM Configuration Assistant for z/OS Communications Server.

This chapter discusses the topics listed in Table 2-1.

Table 2-1 Topics discussed in this chapter

Section	Topic
2.1, “Defining IP filtering” on page 22	Discusses the basic concepts of IP filtering.
2.2, “Why IP filtering is important” on page 24	Discusses key characteristics of IP filtering and why it might be important in your environment.
2.3, “How IP filtering is implemented” on page 25	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions.

© Copyright IBM Corp. 2007. All rights reserved.

2.1 Defining IP filtering

IP filtering enables a z/OS system to classify any IP packet that comes across a network interface and take specific action according to a predefined set of rules. An administrator can configure IP filtering to deny or allow any given network packet into or out of a z/OS system with an IP filtering policy. IP filtering provides:

- Packet filtering and logging
- Filtering rules that determine whether IPSec encryption and authentication are required

The **ipsec** command is used to manage and monitor the IP filtering and VPN policies.

> **Note:** The z/OS V1R8.0 Communications Server introduced IPv6 support for integrated IP Security: IPv6 support for IP filtering, IP security/Virtual Private Network (IPSec/VPN), and Internet Key Exchange (IKE) dynamic key management, no longer requiring the Integrated Security Services Firewall Technologies. IP filtering, IPSec, and Application Transparent Transport Layer Security (AT-TLS) are now all under Policy Agent control. This support provides easier configuration, greater scalability, improved performance, and enhanced serviceability over the Firewall Technologies versions available prior to z/OS V1R8. Therefore, integrated IP security is the recommended way to implement packet filtering in z/OS 1.8.

Filter rules can be defined to match inbound and outbound packets based on:

- Packet information
- Network attributes
- Time of day

Possible actions that can be taken include:

- Permit (with or without manual or dynamic IPSec)
- Deny
- Log (in combination with Permit or Deny)

> **Note:** When an IP packet is blocked, the source of the packet is not informed.

2.1.1 Basic concepts

When a packet arrives over a network interface into the z/OS environment, the IP filtering function running under the TCP/IP stack searches the Security Policy Database (SPD), matching the TCP/IP header against the specified filters. Filters are rules defined to either deny or permit packets. IP filtering matches a filter rule to data traffic based on any combination of IP source or destination address (or masked address), protocol, source or destination port address, direction of flow, or time. In order to create the IP filtering policy, we have to know the resources available in the network, the resources available in a z/OS image, and how they relate to others hosts. Figure 2-1 shows the basic concept of IP filtering.

Figure 2-1 IP filtering at the z/OS data endpoint

The resources available in the z/OS image are:

- TCP/IP address space and stack (can be more than one stack)
- The network interfaces and their respective IP address
- The servers or clients, which are the address spaces running programs that will either access or be accessed by others hosts (such as TN3270 server, File Transfer Protocol (FTP) server and client, and IBM DB2®) and what interfaces they will be using
- The direction of the information flow and if routing might be needed
- Authentication and encryption requirements

The network resources that should be mapped outside the z/OS are:

- Clients and servers that need to connect to the z/OS image, their IP addresses, and the services required
- Networks and subnets

The relationship between the TCP/IP components in a z/OS image and the network resources is translated in the IP filtering implementation. We can call this relationship an IP filtering *policy*. This policy contains all the rules that permit or deny the access to a z/OS image.

Note: The IP filtering function available on the z/OS Communications Server should only be used to control and protect the resources owned by the z/OS image and not to protect resources running on another host, acting as a security gateway or an enterprise firewall.

Security Policy Database

The Security Policy Database provides two types of filter policies: The *default* IP filter policy and the *IP security filter policy*.

The default IP filters policy is intended to allow limited access while the IP security filter policy is being loaded and can be reverted to in an attack situation with an operator command. The default IP filter policy is defined in the TCP/IP profile and default is to deny all traffic. It provides a basic filtering function only (permit rules only and no VPN support).

The IP security filter policy is intended to be the primary source of filter rules and its default is to *deny* all traffic.It is defined in a Policy Agent IPSec configuration file and can be generated by the IBM Configuration Assistant for z/OS Communication Server.

The **ipsec** command is used to switch between the default and IP security filter policies.

The IPSECURITY option on the IPCONFIG statement

The IPSECURITY option enables use of the integrated IP security functions. Figure 2-2 shows a functional view of IP filter policy on z/OS.

Figure 2-2 IP filter policy

2.1.2 For additional information

For additional information, consult the following books:

- *z/OS V1R8.0 Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS V1R8.0 Communications Server: IP Diagnosis Guide*, GC31-8782

2.2 Why IP filtering is important

Depending on your security policies, IP filtering capabilities (provided as part of the z/OS V1R8.0 Communications Server Integrated IP Security support) can provide either the primary means of protecting your z/OS environment from network-based attacks or a powerful additional line of defense (when used in conjunction with layers of external firewalls and access control lists).

IP filtering is required, if you intend to use IPSec for authentication or encryption.

2.3 How IP filtering is implemented

IP filtering is implemented through the z/OS Communications Server PAGENT function. PAGENT is discussed in Chapter 1, “Policy Agent” on page 3.

2.3.1 z/OS IP filtering implementation

There are two parts to implementing the IP filtering policy:

- The *default policy* is specified using the IPSEC statement in the TCP/IP profile data set.
- The *filter policy* is specified using the PAGENT policy configuration files. Those are flat files that can be created in a z/OS UNIX file or in a sequential data set under z/OS.

The TCP/IP profile has to be configured to activate IP filtering. That is done with the IPSECURITY option defined in the IPCONFIG statement[1]. If IPSECURITY is not specified, then the IP filtering function will not be available even if they are configured either in the PAGENT filter policy configuration file or in the IPSEC statement in the TCP/IP profile. Figure 2-3 shows the structure of an IP filtering implementation.

Figure 2-3 IP security filtering structure

As is illustrated in Figure 2-3:

- PAGENT loads the filter policies into the TCP/IP stack at startup or when you make changes to it and refreshes the policy by using the **`modify`** console command.
- The default policy is always loaded by the stack at the start. If it does not have any rules, the implicit rules will be loaded. Any rule defined for the log options can be changed by the **`vary tcpip obeyfile`** console command.
- The **`ipsec`** command is used to manage and monitor the IP security filtering.

[1] Define IPSECURITY in the IPCONFIG6 statement to enable IP filtering for IPv6.

Important: The implicit rules are always created using either the default or the filter policy. Just by using the IPSECURITY option in the IPCONFIG statement, the implicit rules are created and deny all the inbound and outbound TCP/IP traffic in that z/OS image.

The major differences between the default policy and the filter policy are:

- In the default policy, there are only permit rules. The implicit rules implement the deny all functions. In the pagent policy, you have the option to create deny rules.
- The default policy only applies to local packets. If you want to apply filter policies to messages that are being routed between a z/OS image and other z/OS images, then the filter policy must be used.
- There is no option to group similar resources in the default policy. That capability is only available in the filter policy.

The default policy is always used in the absence of a filter policy. If a filter policy is defined, both are loaded into the TCP/IP stack and the filter policy is used unless you specify with the **ipsec** command that the default policy should be used. Using the **ipsec** command, you can switch between the default and the filter policy whenever necessary.

The IPSECURITY option is activated only at the TCP/IP startup. If you want to remove the IP filtering function, you must restart the stack without it.

Example 2-1 shows our IP filtering definitions in the TCP/IP profile for our z/OS test system image A23.

Example 2-1 IPSec configuration statements on profile for z/OS image A23

```
IPCONFIG DATAGRAMFWD
   IPSECURITY       1

 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 40 SECCLASS 1 2
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 40 SECCLASS 1
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 41 SECCLASS 1
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 41 SECCLASS 1

IPSEC          3
   LOGENABLE   4
   LOGIMPLICIT 5
ENDIPSEC
```

1 The IPSECURITY option is specified on the IPCONFIG statement, indicating that we want IP Security activated on the stack. This option automatically installs the default security policy that contains only the implicit rules, meaning that all the IP traffic is blocked. If PAGENT is running and there is an IP filtering policy defined, it is installed and activated.

Important: If the IPCONFIG IPSECURITY statement is coded in the TCP/IP profile, the default IP filter policy is to effectively deny all network traffic, with the exception of some selected Internet Control Message Protocol (ICMP) messages that are necessary for the internal stack function.

2 The SECCLASS option on the LINK statements enables you to either uniquely identify an interface, or to group interfaces with similar security requirements, based on site policy. Then you can configure a single IP filter rule that matches all of the IP traffic from interfaces that share a common security class without explicitly identifying any attributes of the IP packets. In our scenario, we have only one security class for the network outside the z/OS image, the security class 1.

Each non-virtual interface on a z/OS system is assigned a security class. The security class of an interface is determined by the SECCLASS parameter that is coded on either the LINK statement or the DYNAMICXCF parameter of the IPCONFIG statement in the TCP/IP profile. The value of SECCLASS is a number in the range 1 - 255. If SECCLASS is not specified for an interface, the interface is assigned the default security class of 255.

Each IP packet entering or leaving the system inherits the security class of the interface that it traverses:

- For inbound traffic, this is the interface on which the packet arrived.
- For outbound traffic, this is the interface over which the packet is sent.

Security classes can only be assigned to physical interfaces, not VIPA devices. Networks connected to the same network interface (for example, Alpha network and Beta network in Figure 2-4) cannot be distinguished into different security classes.

Figure 2-4 Interface security class example

Note: Although it is possible to configure different security classes for different interfaces into the *same* network (for example, Gamma network in Figure 2-4), all interfaces into the same network should be configured with the same security class to avoid unnecessarily complicating security policies.

Security classes can be used in conjunction with IP address information to create filter rules to block packets having spoofed source IP addresses. For example, if a packet enters the stack from the Delta network (in Figure 2-4), but its source IP address is not from the address space of the Delta network, then the packet is probably spoofed and should be denied.

3 The IPSEC statement is used to define the default policy. The default policy is activated and applied if the filter policy is not defined. Additional rules have to be defined in order to gain access to services running on this TCP/IP stack. In our scenario, our default policy contains only the implicit rules, and all of our real policy rules are defined in the filter policy

configuration. Filter rules can be defined in the IPSEC statement by using the IPSECRULE statement. In our example, we do not have any rules defined on the IPSEC statement.

4 The LOGENABLE option activates packet filter logging. All the log messages are sent to the syslogd by the Traffic Regulation Monitoring daemon (TRMD). You can disable it by using the LOGDISABLE option. In each of the filter rules, there is an option to generate (or not) a log record when the rule is applied to a packet.

5 The LOGIMPLICIT specifies that we want to log all packets that get blocked by the implicit rules in the default policy.

In our implementation, we have all the configuration files stored in a partitioned data set (PDS). This PDS is shared by all of the components: PAGENT, TRMD, SYSLOGD, and TCP/IP address spaces. Figure 2-5 shows the members flow to customize IP filtering from IPSec and PAGENT.

Figure 2-5 Members flow

The IpSecConfig statement in the pagent configuration file is:

```
IpsecConfig //'TCPIPA.TCPPARMS(IPSEC0A)'
```

The IpsecConfig statement specifies the path of an IPSec policy file that contains common IPSec policy statements. If no path name is specified, then the common IPSec policy file specified on the CommonIpSecConfig statement is used.

You can manually create the IP security policy configuration files by coding all of the required statements in a z/OS UNIX file or MVS data set. There are a large number of powerful configuration options provided by IP security policy statements that permit advanced users to

carefully fine-tune the IP security policy. However, IBM also provides a configuration graphical user interface (GUI) that you can use to generate the Policy Agent and IKE daemon configuration files. The IBM Configuration Assistant for z/OS Communication Server is a stand-alone application that runs under the Windows® operating system and requires no network connectivity or setup.

The IBM Configuration Assistant for z/OS Communication Server GUI tool can be downloaded from:

http://www-306.ibm.com/software/network/commserver/zos/support/

For a complete explanation of the IP filtering statements and options, see:

- *z/OS V1R8.0 Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775

2.3.2 FTP and Telnet IP filtering scenario

In the following IP filtering implementation scenario (see Figure 2-6 on page 30), we use the IBM Configuration Assistant for z/OS Communication Server to define IP filter rules for LPAR A23 as follows:

- Permit Telnet traffic between remote workstation (10.12.4.224) and LPAR A23 (10.10.1.230)
- Permit FTP traffic between IP address 10.10.1.241 (LPAR A24) and IP address 10.10.1.230 (LPAR A23)
- Permit *basic services* traffic between all data endpoints and A23

> **Important:** When the IPSECURITY option is activated, the implicit rules are in effect. Therefore, all inbound and outbound traffic traversing the TCP/IP stack is denied.
>
> In our scenario, we allow only specific point-to-point connections; however, there is also some basic services traffic that must be permitted. These include:
>
> - resolver and DNS (if implemented in the z/OS image)
> - Outbound: srcport=any, destport=53, proto=TCP or UDP
> - Inbound: srcport=53, destport=any, proto=TCP or UDP
> - omproute (dynamic routing)
> - RIP: Inbound and outbound: srcport=520, destport=520, proto=UDP; for RIPv2 also need IGMP
> - OSPF: Inbound and outbound: IP protocol 89 (OSPF) and IGMP
>
> We also recommend permitting ping traffic. This allows you to verify connectivity between data endpoints.

- Deny all other application traffic for LPAR A23

Figure 2-6 IP filtering scenario

Implementation steps

The steps are as follows:

1. Start the IBM Configuration Assistant for z/OS Communication Server.

   This presents you with the main configuration window (shown in Figure 2-7), from where you can create and implement policies. Select **IPSec only** (recall that z/OS Communications Server IP filtering support is packaged with the IPSec support), then click **Add New z/OS Image**.

Figure 2-7 Configuration Assistant start panel

2. Figure 2-8 shows that our LPAR name is A23 and we are not using IPSec VPNs with dynamic tunnels, because we are only applying IP filtering in this scenario. Click **Next**.

Figure 2-8 z/OS image information

3. The window shown in Figure 2-9 opens. Click **Finish**.

Figure 2-9 Save new image

4. You are asked to add a TCP/IP Stack, as shown in Figure 2-10. Click **Yes**.

Figure 2-10 Proceed to configuring TCP/IP stack

5. The Welcome wizard window is displayed, as shown in Figure 2-11. Click **Next**.

Figure 2-11 New Stack Wizard

6. The window shown in Figure 2-12 opens.

   Our stack name is TCPIPA and, as with the z/OS image, we are not using IPSec VPNs with dynamic tunnels, because we are only applying IP filtering in this scenario.

   Click **Next**.

Figure 2-12 IP stack information

7. In Figure 2-13, configure the stack-level settings. We chose to:
   - Enable Filter Logging Policy.
   - De-encapsulate and then filter IPSec payloads rather than filter IPSec headers.
   - Allow IP V6 link activation.

   Click **Next**.

Figure 2-13 Filter Logging Policy

8. You are asked to save your Filter settings (Figure 2-14). Click **Finish**.

Figure 2-14 Save Filter settings

9. You are asked to configure the connectivity rules for the new stack (see Figure 2-15). Click **Yes**.

Figure 2-15 Proceed to configuring connectivity rules

10. Indicate your connectivity rule type in the Connectivity Rule: Welcome window shown in Figure 2-16.

Figure 2-16 Connectivity Rule Welcome window

11. Select **No topology information is required** and click **Next** (see Figure 2-17).

Figure 2-17 Network topology

12. In the panel shown in Figure 2-18, specify the IP addresses of the data endpoints. It can be a single IP address, an IP address range, an IP subnet, or all IPv4 or all IPv6 addresses.

    The name of the Connectivity Rule must be a string from 1 to 25 characters. The configuration assistant supplies a “0” in this field, which you can change to something more meaningful.

    Important: The *Source data endpoint* must be the IP address of the stack that you are protecting.

    Click **Next**.

Figure 2-18 Connectivity rules

13. The Connectivity Rule: Select Requirement Map window in Figure 2-19 opens. There are IBM-provided maps; however, in our example, we want to configure a new rule, therefore we click **Add for Beginners**.

Figure 2-19 Map for data endpoints

14. This opens the Requirement Map Introduction window shown in Figure 2-20. Click **Next**.

Figure 2-20 Requirement Map Introduction window

15. Figure 2-21 shows a listing of likely traffic types. Select **TN3270 Server traffic** to allow Telnet connectivity between remote workstation (10.12.4.224) and LPAR A23 (10.10.1.230). Click **Next**.

Figure 2-21 Traffic descriptors

16. The window shown in Figure 2-22 gives you the opportunity to add or remove traffic descriptors.

    Note that the IPSECURITY implicit rules (deny all traffic) are in effect. Therefore, the all_other_traffic descriptor is not needed in the requirement map. If required, you can remove it by selecting **All_other_traffic** and clicking **Remove**.

    Click **Next**.

Figure 2-22 Add or remove traffic descriptors

17. Set the security levels. Each entry in the Requirement Map is a mapping between a Traffic Descriptor and either Deny, Permit, or IPSec Security Level (see Figure 2-23).

    Click the I**PSec - Security Level** tab for TN3270-Server and select **Permit**. Click **OK**.

Figure 2-23 Select security level

18. This takes you back to the Connectivity Rule Requirement Map window, which contains the newly created Telnet map (Figure 2-24). Click **Next**.

Figure 2-24 Connectivity Rule: Requirement Map

19. The Requirement Map: Additional Settings window shown in Figure 2-25 opens. Click **Finish**.

Figure 2-25 Connectivity Rule: Additional Settings

The window shown in Figure 2-26 shows the result of our definitions.

Figure 2-26 Stack settings

We have completed the connectivity rule for Telnet, and now we need to add another rule to allow FTP traffic between LPAR A23 (10.10.1.230) and LPAR A24 (10.10.1.241).

1. Click **Add** and the Connectivity Rule; Welcome window (Figure 2-16 on page 36) opens. We select the **Typical** (Default) rule type and click **Next**.
2. The window shown in Figure 2-27 opens. Select **No topology information is required** and click **Next**.

Figure 2-27 Network topology

3. In the window shown in Figure 2-28, we specify the IP addresses of the data endpoints (A23 and A24). It can be a single IP address, an IP address range, an IP subnet, or all IPv4 or all IPv6 addresses. Click **Next**.

Figure 2-28 Connectivity rule

4. This takes you to the Connectivity Rule: Select Requirement Map window. Click **Add for Beginners**.
5. This opens the Requirement Map: Introduction window. Click **Next** to determine which traffic types to allow.

6. The window shown in Figure 2-29 opens. Select **FTP Client traffic** and **FTP Server traffic**. Click **Next**.

Figure 2-29 Traffic types

7. Figure 2-30 shows that three lines of Traffic Descriptors are generated with the following values:
   - FTP-Server
   - FTP-Client
   - All_other_traffic

   Note that the IPSECURITY implicit rules (deny all traffic) are in effect. Therefore, the all_other_traffic descriptor is not needed in the requirement map. If required, you can remove it by selecting **all_other_traffic** and clicking **Remove**.

   Click **Next**.

Figure 2-30 Add traffic descriptor

8. Set the security levels. Each entry in the Requirement Map is a mapping between a Traffic Descriptor and either Deny, Permit, or IPSec Security Level (see Figure 2-31). Click the **IPSec - Security Level** tab for FTP-Server and select **Permit**. Do the same for FTP-Client traffic descriptor. Click **OK**.

Figure 2-31 Select security level

9. This returns you back to the Connectivity Rule: Select Requirement Map window (see Figure 2-32). Click **Next**.

Figure 2-32 Connectivity Rule: Requirement Map

10. The Requirement Map: Additional Settings window shown in Figure 2-33 opens. Click **Finish**.

Figure 2-33 Connectivity Rule: Advanced Settings

Figure 2-34 shows the results of our definitions.

Figure 2-34 Stack settings

Next we need to add a requirement map for the basic services to permit ping, resolver, DNS, and omproute traffic between A23 and all data endpoints.

1. In the Configuration Assistant Navigation Tree (on the left), click **Requirement Maps** (under **Work with Reusable Objects**). See Figure 2-35. Click **Add**.

Figure 2-35 Adding a requirement map

2. Provide a name and description, as shown in Figure 2-36.

   Remove All_other_traffic by selecting **All_other_traffic** and clicking **Remove**.

   Add ping, resolver, DNS, and omproute by selecting the appropriate traffic descriptors from the Objects list and clicking **Add**.

Figure 2-36 Removing and adding traffic descriptors

3. Click the I**PSec - Security Level** tab and select **Permit** for each traffic descriptor (see Figure 2-37). Click **OK**.

Figure 2-37 Change security level

4. The list of requirement maps opens (see Figure 2-38).

Figure 2-38 Requirement Maps

In the Configuration Assistant Navigation Tree (on the left), click the required TCP/IP stack (under **Work with z/OS Images**). See Figure 2-39. Click **Add**.

Figure 2-39 z/OS image list

5. The Connectivity Rule: Welcome window opens. We select the **Typical** (Default) rule type and click **Next**.

6. In the window shown in Figure 2-40, select **No topology information is required** and click **Next**.

Figure 2-40 Network topology

7. In Figure 2-41, we specify **All IP V4 addresses**, because this rule applies to all data endpoints in our network. Click **Next**.

Figure 2-41 Connectivity rule

8. This opens the Select Requirement Map (see Figure 2-42). Select a requirement map (in our case it was BasicServices), and click **Next**.

Figure 2-42 Requirement map list

9. The TCP/IP stack settings opens (see Figure 2-43). Because we want BasicServices traffic to flow between endpoints in all cases, it must be moved to the top of the list. This is done by clicking **Move Up**.

> **Note:** The IP addresses in a packet are compared with the IP addresses of the data endpoints of the connectivity rules in the order that those rules appear in the table.

Click **OK**.

Figure 2-43 TCP/IP stack settings

10. This takes you back to the Connectivity Rule Requirement Map window, which contain the newly created basic services map. Click **Health Check...** to determine if the TCP/IP stack contains configuration errors or inappropriate combinations of rules.

With our IP filtering rules complete, we install the configuration files as follows:

1. Select **Edit** in the top left corner, or right-click the stack name, and click **Install Configuration Files**. In Figure 2-44, we see that there are two files to be installed:
   - A sample TCP/IP profile to insert in the profile TCP/IP
   - An IPSec configuration file

Figure 2-44 Installation stack

We have two options:

– **Show Configuration File**, save the file on your computer using **Save As**, and then upload the file to z/OS later.

– Directly **FTP** the file to z/OS.

Below, we show each of the configuration files and then an example of the FTP.

2. We select **IPSec: Policy Agent Stack Configuration**, and click **Show Configuration File**. The file in Example 2-2 is generated by the Configuration Assistant.

Example 2-2 IPSec configuration file

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC30
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Thu Dec 07 15:53:28 EST 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}

IpGenericFilterAction          Deny~LogNo
{
  IpFilterAction               Deny
  IpFilterLogging              No
}

IpService                      OMPROUTE-IP_V4
{
  Protocol                     OSPF
  Type                         Any
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~1
{
  Protocol                     IGMP
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~2
{
  Protocol                     UDP
  SourcePortRange              520
  DestinationPortRange         1024 65535
  Direction                    BiDirectional
  Routing                      Either
}
```

```
IpService                       OMPROUTE-IP_V4~3
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~4
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS
{
  Protocol                      UDP
  SourcePortRange               53
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS~5
{
  Protocol                      UDP
  SourcePortRange               53
  DestinationPortRange          53
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS~6
{
  Protocol                      TCP
  SourcePortRange               53
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS~7
{
  Protocol                      TCP
  SourcePortRange               53
  DestinationPortRange          53
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       Resolver
{
  Protocol                      TCP
  SourcePortRange               1024 65535
  DestinationPortRange          53
  Direction                     BiDirectional OutboundConnect
```

```
  Routing                       Local
}

IpService                       Resolver~8
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          53
  Direction                     BiDirectional
  Routing                       Local
}

IpService                       Ping-IP_V4
{
  Protocol                      ICMP
  Type                          8
  Code                          Any
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       Ping-IP_V4~9
{
  Protocol                      ICMP
  Type                          0
  Code                          Any
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       TN3270-Server
{
  Protocol                      TCP
  SourcePortRange               23
  DestinationPortRange          1024 65535
  Direction                     BiDirectional InboundConnect
  Routing                       Either
}

IpService                       All_other_traffic
{
  Protocol                      All
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       FTP-Client
{
  Protocol                      TCP
  SourcePortRange               1024 65535
  DestinationPortRange          21
  Direction                     BiDirectional OutboundConnect
  Routing                       Either
}

IpService                       FTP-Client~10
{
  Protocol                      TCP
  SourcePortRange               1024 65535
  DestinationPortRange          20
```

```
  Direction                     BiDirectional InboundConnect
  Routing                       Either
}

IpService                       FTP-Client~11
{
  Protocol                      TCP
  SourcePortRange               1024 65535
  DestinationPortRange          50000 50200
  Direction                     BiDirectional OutboundConnect
  Routing                       Either
}

IpService                       FTP-Server
{
  Protocol                      TCP
  SourcePortRange               21
  DestinationPortRange          1024 65535
  Direction                     BiDirectional InboundConnect
  Routing                       Either
}

IpService                       FTP-Server~12
{
  Protocol                      TCP
  SourcePortRange               20
  DestinationPortRange          1024 65535
  Direction                     BiDirectional OutboundConnect
  Routing                       Either
}

IpService                       FTP-Server~13
{
  Protocol                      TCP
  SourcePortRange               50000 50200
  DestinationPortRange          1024 65535
  Direction                     BiDirectional InboundConnect
  Routing                       Either
}
##
## Connectivity Rule BasicServices combines the following items:
##   Local data endpoint        All4
##   Remote data endpoint       All4
##   Topology                   None (Permit/Deny only)
##   Requirement Map            BasicServices
##     OMPROUTE-IP_V4             => Permit
##     DNS                        => Permit
##     Resolver                   => Permit
##     Ping-IP_V4                 => Permit

IpFilterRule                    BasicServices~1
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    BasicServices~2
{
```

```
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~1
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~3
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~2
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~4
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~3
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~5
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~4
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~6
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~7
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS~5
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~8
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS~6
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~9
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS~7
  IpGenericFilterActionRef     Permit~LogNo
```

```
}

IpFilterRule                    BasicServices~10
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  Resolver
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    BasicServices~11
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  Resolver~8
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    BasicServices~12
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  Ping-IP_V4
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    BasicServices~13
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  Ping-IP_V4~9
  IpGenericFilterActionRef      Permit~LogNo
}
##
## Connectivity Rule Telnet combines the following items:
##   Local data endpoint        Telnet~ADR~1
##   Remote data endpoint       Telnet~ADR~2
##   Topology                   None (Permit/Deny only)
##   Requirement Map            Telnet
##     TN3270-Server              => Permit
##     All_other_traffic          => Deny

IpAddr                          Telnet~ADR~1
{
  Addr                          10.10.1.230
}

IpAddr                          Telnet~ADR~2
{
  Addr                          10.12.4.224
}

IpFilterRule                    Telnet~3
{
  IpSourceAddrRef               Telnet~ADR~1
  IpDestAddrRef                 Telnet~ADR~2
  IpServiceRef                  TN3270-Server
  IpGenericFilterActionRef      Permit~LogNo
}
```

```
IpFilterRule                    Telnet~4
{
  IpSourceAddrRef               Telnet~ADR~1
  IpDestAddrRef                 Telnet~ADR~2
  IpServiceRef                  All_other_traffic
  IpGenericFilterActionRef      Deny~LogNo
}
##
## Connectivity Rule FTP combines the following items:
##   Local data endpoint        FTP~ADR~1
##   Remote data endpoint       FTP~ADR~2
##   Topology                   None (Permit/Deny only)
##   Requirement Map            FTP
##     FTP-Client                 => Permit
##     FTP-Server                 => Permit
##     All_other_traffic          => Deny

IpAddr                          FTP~ADR~1
{
  Addr                          10.10.1.230
}

IpAddr                          FTP~ADR~2
{
  Addr                          10.10.1.241
}

IpFilterRule                    FTP~3
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Client
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~4
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Client~10
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~5
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Client~11
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~6
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Server
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~7
```

```
{
  IpSourceAddrRef              FTP~ADR~1
  IpDestAddrRef                FTP~ADR~2
  IpServiceRef                 FTP-Server~12
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   FTP~8
{
  IpSourceAddrRef              FTP~ADR~1
  IpDestAddrRef                FTP~ADR~2
  IpServiceRef                 FTP-Server~13
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   FTP~9
{
  IpSourceAddrRef              FTP~ADR~1
  IpDestAddrRef                FTP~ADR~2
  IpServiceRef                 All_other_traffic
  IpGenericFilterActionRef     Deny~LogNo
}

IpFilterPolicy
{
  PreDecap                     OFF
  FilterLogging                ON
  IpFilterLogImplicit          Yes
  AllowOnDemand                Yes
  IpFilterRuleRef              BasicServices~1
  IpFilterRuleRef              BasicServices~2
  IpFilterRuleRef              BasicServices~3
  IpFilterRuleRef              BasicServices~4
  IpFilterRuleRef              BasicServices~5
  IpFilterRuleRef              BasicServices~6
  IpFilterRuleRef              BasicServices~7
  IpFilterRuleRef              BasicServices~8
  IpFilterRuleRef              BasicServices~9
  IpFilterRuleRef              BasicServices~10
  IpFilterRuleRef              BasicServices~11
  IpFilterRuleRef              BasicServices~12
  IpFilterRuleRef              BasicServices~13
  IpFilterRuleRef              Telnet~3
  IpFilterRuleRef              Telnet~4
  IpFilterRuleRef              FTP~3
  IpFilterRuleRef              FTP~4
  IpFilterRuleRef              FTP~5
  IpFilterRuleRef              FTP~6
  IpFilterRuleRef              FTP~7
  IpFilterRuleRef              FTP~8
  IpFilterRuleRef              FTP~9
}
```

Figure 2-45 shows the FTP process for transferring the configuration files.

Figure 2-45 File transfer

2.3.3 Verification

We did the following tasks to verify our IP filtering configurations:

- Log on to LPAR A23 using TN3270
- FTP to LPAR A23 from a remote workstation (should fail)
- FTP from LPAR A24 to LPAR A23

Log on to LPAR A23 using TN3270

From our workstation, we create a TELNET session to TCPIPA (10.10.1.230) on A23 (SC30). See Figure 2-46.

Figure 2-46 TN3270 Host definition

Figure 2-47 shows that the connection between the remote workstation (10.12.4.224) and LPAR A23 (SC30) was established.

Figure 2-47 TN3270 connection

FTP to LPAR A23 from a remote workstation (should fail)

From the remote workstation, we are not allowed to FTP into LPAR A23, as shown in Figure 2-48.

```
C:\>ftp 10.10.1.230
> ftp: connect :Unknown error number
ftp> ls
Not connected.
ftp>
```

Figure 2-48 Workstation FTP fails

FTP from LPAR A24 to LPAR A23

From our workstation, we can log on to z/OS image A24 (SC31), because we have not set up any IP filtering rules for A24 (see Figure 2-6 on page 30). From there, we can demonstrate that the host-to-host FTP works, because we set up our filter rules to permit FTP traffic between LPARs A24 and A23.

Using TSO on LPAR A24, we then issued the following command:

```
ftp -p tcpipa 10.10.1.230
```

Figure 2-49 shows the result of the successful TSO FTP command.

```
 220-FTPMVS1 IBM FTP CS V1R8 at WTSC30A, 17:19:29 on 2006-10-13.
 220 Connection will close if idle for more than 5 minutes.
 NAME (10.10.1.230:CS10):
CS10
 >>> USER CS10
 331 Send password please.
 PASSWORD:


 >>> PASS
 230 CS10 is logged on. Working directory is "CS10.".
 Command:
```

Figure 2-49 TSO output

Problem determination

The following documentation can aid in problem determination:

- TCP/IP CTRACE output: The CTRACE facility has flexibility such as filtering, combining multiple concurrent applications and traces, and using an external writer. (For additional information, see *z/OS V1R8.0 Communications Server: IP Diagnosis Guide*, GC31-8782.)
- PASEARCH command: Use the **pasearch** command to display policy rules with complex conditions.
- TCP/IP profile output
- POLICY config file output

3

IP Security

IP Security (IPSec) is a suite of protocols and standards defined by the Internet Engineering Task Force (IETF) to provide an open architecture for security at the IP networking layer of TCP/IP. IPSec provides the framework to define and implement network security based on policies defined by your organization.

This chapter discusses the implementation of IPSec on z/OS. Table 3-1 lists the topics covered in this chapter.

Table 3-1 Topics discussed in this chapter

Section	Topic
3.1, “IPSec definition” on page 76	The basic concepts of IPSec and Virtual Private Network (VPN) are discussed in this section.
3.2, “Key IPSec components” on page 76	The key IPSec components are discussed in this section.
3.3, “How IPSec is implemented” on page 78	The implementation procedures for installing IPSec are discussed in this section.
3.4, “Configuring quick dynamic tunnels” on page 88	This section explains how to configure dynamic tunnels using the option of quick dynamic tunnels.
3.5, “Implementing IPSec between two z/OS systems” on page 95	This section shows how to implement IP security between two z/OS images.
3.6, “Implementing IPSec between z/OS and Windows” on page 143	This section shows how to implement IP security between z/OS image and Windows XP.
3.7, “AES cryptographic support for integrated IPSec/VPN” on page 180	This section shows how to implement IP security between two z/OS images using AES encryption.
3.8, “IPv6 support for integrated IPSec/VPN” on page 186	This section shows how to implement IP security between two z/OS images over IPv6.
3.9, “Further information” on page 203	This section refers the readers for more reading.

© Copyright IBM Corp. 2007. All rights reserved.

3.1 IPSec definition

IPSec is a suite of protocols and standards defined by the IETF to provide an open architecture for security at the IP networking layer of TCP/IP. Because IPSec works at the IP networking layer, it can be used to provide security for any TCP/IP application without modification. If necessary, applications can have their own additional security features on top of the underlying IPSec security. Also, unlike TCP-layer-based security implementations (such as SSL/TLS), IPSec can be used to protect both TCP and UDP applications. The IPSec standards have also been structured so that they can accommodate newer, more powerful, algorithms as they become available in the future.

IPSec is often referred to as a Virtual Private Networking (VPN) technology, because it enables an enterprise to extend its network across an untrusted network (such as the Internet) without compromising security. Using IPSec protocols, each host can encrypt and authenticate individual IP packets between itself and other communicating hosts. Companies can therefore securely, and cost effectively, extend the reach of their applications and data across the world by replacing leased lines to remote sites with VPN connections. Because Internet access is increasingly available worldwide, companies can now use VPN technologies to reach places where other connectivity alternatives such as leased lines are expensive or not available.

The z/OS V1R8.0 Communications Server implements the following IPSec RFCs:

- RFC 2401: Security Architecture for the Internet Protocol
- RFC 2402: IP Authentication Header
- RFC 2403: The Use of HMAC-MD5-96 within ESP and AH
- RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH
- RFC 2406: IP Encapsulating Security Payload (ESP)
- RFC 2407: The Internet IP Security Domain of Interpretation for ISAKMP
- RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP)
- RFC 2409: The Internet Key Exchange (IKE)
- RFC 2410: The NULL Encryption Algorithm and Its Use with IPSec
- RFC 2451: The ESP CBC-Mode Cipher Algorithms
- RFC 3602: The AES-CBC Cipher Algorithm And Its Use with IPSec
- RFC 3715: IPSec-Network Address Translation (NAT) Compatibility Requirements
- RFC 3947: Negotiation of NAT-Traversal in the IKE
- RFC 3948: UDP Encapsulation of IPSec ESP Packets

Note: One thing that can be confusing about z/OS V1R8.0 Communications Server IPSec support is that it has been packaged together with IP filtering support and is referred to as *integrated IP Security*. That is because there is a very close affinity between IPSec and IP filtering in the z/OS Communications Server; although you can implement IP filtering without IPSec, you cannot implement IPSec without IP filtering (discussed in Chapter 2, “IP filtering” on page 21).

3.2 Key IPSec components

Three of the most important IPSec components implemented by the z/OS V1R8.0 Communications Server include:

- RFC 2402: IP Authentication Header (AH) protocol, which provides for data authentication, IP header authentication, and data origin authentication
- RFC 2406: IP Encapsulating Security Payload (ESP), which provides for data authentication, data origin authentication, and data privacy (encryption)

- RFC 2409: The Internet Key Exchange (IKE), which provides protocols for automated encryption key management

3.2.1 IP Authentication Header protocol

As the name suggests, IPSec Authentication Header (AH) authenticates IP packets, ensuring that they came from a legitimate origin host and that they have not been changed. IPSec AH provides:

- Data integrity by authenticating the *entire IP packet* using a message digest that is generated by algorithms such as HMAC-MD5 or HMAC-SHA
- Data origin authentication by using a shared secret key to create the message digest
- Replay protection by using a sequence number field within the AH header

3.2.2 IP Encapsulating Security Payload protocol

Encapsulating Security Payload (ESP) provides additional protection beyond (or in addition to) AH, including:

- Encapsulating and encrypting the IP packet
- Authenticating the IP datagram portion of the IP packet

In ESP, before leaving a host, outbound packets are rebuilt with additional IPSec headers using a cryptographic key that is known to both communicating hosts. This is called *encapsulation*. On the receiving side, the inbound packets are stripped of their IPSec headers (decapsulated) using the same cryptographic key, thereby recovering the original packet. Any packet that is intercepted on the IP network is unreadable to anyone without the encryption key. Any modifications to the IP packet while in transit are detected by authentication processing at the receiving host and is discarded.

3.2.3 Internet Key Exchange protocol

The IKE protocol (implemented in the z/OS V1R8.0 Communications Server by the IKE daemon) manages the transfer and periodic changing of security keys between senders and receivers. Key exchange, defined in IKE, is normally a two-step process:

1. The partners establish a logical connection, called a *Security Association (SA)*, and decide on security parameters such as encryption and hashing algorithms and authentication methods (IKE SA).
2. After the appropriate security parameters have been negotiated, they set up a second Security Association for the actual data transfer (IPSec SA).

Such secure logical connections between pairs of endpoints are often called *tunnels*. The z/OS V1R8.0 Communications Server IPSec implementation refers to two types of tunnels:

- Manual tunnels: The security parameters and encryption keys are statically configured and are manually managed by a security administrator.
- Dynamic tunnels: The security parameters are negotiated and the encryption keys are generated dynamically using IKE.

The IKE daemon uses the IP security policies defined by you in the Policy Agent (PAGENT) and dynamically manages the keys that are associated with dynamic IPSec VPNs.

3.3 How IPSec is implemented

IPSec uses the services of a number of z/OS Communications Server components (illustrated in Figure 3-1) to provide policy-based security.

Figure 3-1 z/OS Communications Server components involved with IPSec

The steps to implement IPSec are:

1. Set up the Internet Key Exchange daemon.
2. Set up the system logging daemon to log IKED messages.
3. Start IKE daemon and verify it initializes.
4. Set up Traffic Regulation Manager daemon.
5. Update the TCP/IP stack to activate IPSec.
6. Restrict the use of the ipsec command.
7. Install the Policy Agent.
8. Define the IPSec policies to PAGENT.

In the following sections, we briefly describe each of these steps.

3.3.1 Set up the Internet Key Exchange daemon

The Internet Key Exchange daemon (IKED) is responsible for retrieving the IP security policy from the Policy Agent, and dynamically managing keys that are associated with dynamic tunnels. The IKE daemon implements the protocols to dynamically establish IKE SAs with peers that also support these protocols. It can provide automatic management of cryptographic keys and remove the administrative burden associated with key creation, distribution, and maintenance.

IKE provides the following services:

- Host authentication (ensuring that each hosts is certain of the other's identity)
- The negotiation of a security association as follows:
  - Agreeing on the type of traffic to be protected
  - Agreeing on the authentication and encryption algorithms to be used
  - Generating cryptographic keys
- Nondisruptive periodic refresh of keys
- The deletion of security associations whose lifetimes have expired

IKE operates at the application layer and communicates between two IKE peers using a series of UDP messages.

Only one instance of the IKE daemon can run on a single z/OS image. The IKE daemon obtains operational parameters from the configuration file and the IP security policy from the Policy Agent.

Attention: When the IKE daemon has obtained the IP Security policy, the Policy Agent may be stopped without impacting the IKE daemon. However, any changes to the IP Security policy are not detected until the Policy Agent is restarted. The IKE daemon reconnects to the Policy Agent when it is restarted.

Note: The IKE daemon is required only if IpDynVpnAction statements are utilized in the policy.

The steps to set up the IKE daemon are:

1. Set up the IKE daemon cataloged procedure.
2. Create the IKE daemon configuration file.
3. Reserve the TCP/IP ports for IKE daemon.
4. Associate a RACF user ID and group with the IKE daemon.
5. Define profiles to control access to the RACDCERT command.
6. Create a RACF key ring.
7. Install an X509 digital certificate for the IKE daemon.
8. (Optionally) Authorize use of hardware cryptographic encryption.

These steps are explained in the following sections.

Set up the IKE daemon cataloged procedure

A sample of the procedure can be obtained from the z/OS Communications Server installation file TCPIP.SEZAINST(IKED). Example 3-1 shows the procedure we used. Copy this procedure into your SYS1.PROCLIB library.

Example 3-1 IKE daemon cataloged procedure

```
//IKED     EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD DUMMY
//STDENV   DD DSN=TCPIP.IKED.ENV(IKED),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

TCPIP.IKED.ENV(IKED) contains the following:

```
IKED_FILE=/etc/security/iked.conf
IKED_CTRACE_MEMBER=CTIIKE00
```

The file /etc/security/iked.conf is the IKE daemon configuration file shown in Example 3-2 on page 80.

IKED_CTRACE_MEMBER is the name of a parmlib member that contains default CTRACE settings for IKE daemon.

Create the IKE daemon configuration file

A sample configuration is supplied in the z/OS Communications Server installation file /usr/lpp/tcpip/samples/IBM/EZAIKCFG. The configuration we used is shown in Example 3-2.

Example 3-2 IKE daemon configuration file

```
IkeConfig
{
 IkeSyslogLevel    255
 PagentSyslogLevel 128
 Keyring           IKED/IKED_keyring
 KeyRetries        10
 KeyWait           30
 DataRetries       10
 DataWait          15
 Echo              no
 PagentWait        0
}
```

These parameters control the workings of the IKE daemon, and each is explained as follows:

- IkeSyslogLevel: Level of logging from the IKE daemon. We left it at the highest level to get all messages during testing. On the production system, this can be set to 1.
- PagentSyslogLevel: Level of logging from pagent. We set it to the highest level of 128 for testing.
- Keyring: Owning user ID and ringname for RSA Signature Mode of authentication. We set this up later in “Create a RACF key ring” on page 82.
- KeyRetries: Number of times the IKE daemon retransmits a key negotiation before aborting.
- KeyWait: Number of seconds between retransmissions of key negotiations.
- DataRetries: Number of times the IKE daemon retransmits a data negotiation before aborting.
- DataWait: Number of seconds between retransmissions of data negotiations.
- Echo: Option to echo all IKE daemon log messages to the IKEDOUT DD file.
- PagentWait: The time limit in seconds to wait for connection to the Policy Agent. A value of 0 means retry forever.

Attention: The IkeSyslogLevel and PagentSyslogLevel have been set for the maximum level of tracing for our testing. In the production environment, you should set them to low levels to avoid a performance impact from excessive logging.

We edited our IKE daemon configuration file as follows:

1. We issued the **su** command in UNIX to get superuser authority.
2. We copied the file /usr/lpp/tcpip/samples/IBM/EZAIKCFG to /etc/security/iked.conf using the UNIX command:

   ```
   cp /usr/lpp/tcpip/samples/IBM/EZAIKCFG /etc/security/iked.conf
   ```

3. We updated this file using the UNIX command:

   ```
   oedit /etc/security/iked.conf
   ```

Tip: Newly created z/OS UNIX files might not have write access. Issue the **`ls -al`** command for the file to verify this and issue the **`chmod 700`** command to change the access, if necessary.

Reserve the TCP/IP ports for IKE daemon

Update the PORT statement in PROFILE.TCPIP to reserve ports 500 and 4500 for the IKE daemon:

```
PORT
500 UDP IKED
4500 UDP IKED
```

Make sure to specify the correct name of the IKE daemon that you are using here. We used the name IKED.

Associate a RACF user ID and group with the IKE daemon

We defined a user ID IKED with default group TCPGRP and with an OMVS segment. This user ID needs to be defined with UID=0. But only one user ID can have UID=0 in the system and it is normally already assigned to user BPXROOT in most installations. Therefore, you have to use the 'SHARED' parameter in the definition. A home directory was also assigned to this user ID.

We then defined the started task IKED to RACF and associated the user IKED and group TCPGRP using the RDEFINE command. We refreshed the RACLIST and GENERIC for the STARTED class to update the profiles in storage with this new information.

Example 3-3 shows the commands we used. Note that IKE daemon user ID IKED requires read access to RACF profile BPX.DAEMON in the FACILITY resource class to work as a daemon.

Example 3-3 Associate a RACF user ID and group with IKE daemon

```
ADDUSER IKED DFLTGRP(TCPGRP) OMVS(UID(0) SHARED HOME('/'))
RDEFINE STARTED IKED.* STDATA(USER(IKED) GROUP(TCPGRP))
PERMIT BPX.DAEMON CLASS(FACILITY) ID(IKED) ACCESS(READ)
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED) REFRESH
```

Attention: If you are a network programmer, you might not have the necessary RACF authority to issue many of these commands. You might need to work with your RACF administrator who will have the necessary "SPECIAL" authority to issue them.

Define profiles to control access to the RACDCERT command

The RACDCERT command is used to generate keys and key rings and to connect the keys to key rings. This facility should be protected and only authorized users such as the IKE daemon should have access to it. Example 3-4 shows the commands we used.

Example 3-4 Control access to the RACDCERT command

```
RDEFINE FACILITY IRR.DIGTCERT.ADD UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ADDRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.CONNECT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENREQ UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
```

```
PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(IKED) ACC(UPDATE)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(IKED) ACC(UPDATE)
```

Create a RACF key ring

Digital certificates are made available to the IKE server by connecting them to a key ring that is owned by the IKE server. To create a key ring for the IKE server, issue the TSO command:

```
RACDCERT ID(IKED) ADDRING(IKED_keyring)
```

Note: The value used for key ring is case-sensitive.

Install an X509 digital certificate for the IKE daemon

You can install an X509 digital certificate in the following ways:

- Generate an X509 digital certificate for the IKE server and have it signed by a certificate authority.
- Generate a self-signed X509 digital certificate for the IKE server.
- Migrate an existing key database to a RACF key ring.

We generated a self-signed X509 digital certificate. The steps that we performed are explained below:

1. Activate the RACF classes DIGTCERT and DIGTNMAP if not already active.
2. Generate a self-signed certificate to represent the local certificate authority.
3. Create a certificate for the server.
4. Connect the certificates to IKED's key ring.
5. Tell IKE daemon where to find the key ring.
6. Verify certificate creation.

These steps are explained in the following sections.

Activate the RACF classes DIGTCERT and DIGTNMAP if not already active

The DIGTCERT (contains digital certificates and information related to them) and DIGTNMAP (mapping class for certificate name filters) classes should be active for RACF certificate creation. The command to do this is:

```
SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)
```

Generate a self-signed certificate to represent the local certificate authority

We created a certificate to act as local certificate-issuing authority. The label for our certificate was *My Local Certificate Authority*. This label will be used to refer to the certificate in the following steps. Example 3-5 shows the command to do this.

Example 3-5 Generate a self-signed certificate

```
RACDCERT ID(IKED) CERTAUTH GENCERT SUBJECTSDN( O('I.B.M Corporation')
-
            CN('itso.ibm.com')                          -
            C('US'))                                    -
            WITHLABEL('My Local Certificate Authority') -
            KEYUSAGE(certsign)
```

Create a certificate for the server

We created a certificate for the IKED daemon and signed the new certificate with authority of My Local Certificate Authority, which was created to represent the local certificate authority. Example 3-6 shows the command we used.

Example 3-6 Create a certificate for the server

```
RACDCERT ID(IKED) GENCERT
         SUBJECTSDN  (CN('IKE Daemon on SC30')
                      OU('ITSO')
                      C('US'))
                      WITHLABEL('IKE Daemon on SC30')
                      SIGNWITH(CERTAUTH
                      label('My Local Certificate Authority'))
```

Connect the certificates to IKED's key ring

The certificates created in the previous two steps need to be connected to IKED's key ring. The commands shown in Example 3-7 accomplish this.

Example 3-7 Connect the certificate to IKED's existing key ring

```
RACDCERT ID(IKED) CONNECT(ID(IKED)                      -
                  LABEL('IKE Daemon on SC30')           -
                  RING(IKED_keyring)                    -
                  USAGE(personal))
RACDCERT ID(IKED) CONNECT(ID(IKED) CERTAUTH             -
                  LABEL('My Local Certificate Authority') -
                  RING(IKED_keyring)                    -
                  USAGE(certauth))
```

Tell IKE daemon where to find the key ring

We then added the following statement to the IKE daemon configuration file, etc/security/iked.conf, which we defined earlier:

```
Keyring IKED_keyring
```

Verify certificate creation

You can verify that the certificates that you have created are connected to the key ring associated with user ID IKED by using the RACDCERT command and examining the output of the Ring Associations field. Example 3-8 shows the commands to do the verification.

Example 3-8 Verify certificate creation

```
RACDCERT ID(iked) LIST(LABEL('IKE Daemon on SC30'))
RACDCERT ID(IKED) CERTAUTH   -
                  LIST(LABEL('My Local Certificate Authority'))
RACDCERT id(IKED) LISTRING(IKED_keyring)
```

Example 3-9 shows the output of these commands.

Example 3-9 Verify certificate creation

```
Digital certificate information for user IKED:
Label: IKE DAEMON ON SC30
Certificate ID: 2QTJOsXEydLFQMTBxdTW1UDW1UDiw/Pw
Status: TRUST
Start Date: 2006/07/27 00:00:00
End Date:   2007/07/27 23:59:59
```

```
Serial Number:
     >01<
Issuer's Name:
     >CN=ITSO.IBM.COM.O=I.B.M.C=US<
Subject's Name:
     >CN=IKE DAEMON ON SC30.OU=ITSO.C=US<
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
  Ring Owner: IKED
  Ring:
     >IKE_KEYRING<
  Ring Owner: IKED
  Ring:
     >IKED_KEYRING<
 Digital certificate information for CERTAUTH:

   Label: NEW LOCAL CERTIFICATE AUTHORITY
   Certificate ID: 2QiJmZmDhZmjgdXF5kDT1sPBOODDxdnjycbJw8HjxUDB5OPI1tnJ4+hA
   Status: TRUST
   Start Date: 2006/07/27 00:00:00
   End Date:   2007/07/27 23:59:59
   Serial Number:
        >00<
 Issuer's Name:
      >CN=ITSO.IBM.COM.O=I.B.M CORPORATION)      CN(.C=US<
 Subject's Name:
      >CN=ITSO.IBM.COM.O=I.B.M CORPORATION)      CN(.C=US<
 Key Usage: CERTSIGN
 Private Key Type: Non-ICSF
 Private Key Size: 1024
 Ring Associations:
 *** No rings associated ***

 Digital ring information for user IKED:

   Ring:
        >IKED_KEYRING<
   Certificate Label Name             Cert Owner     USAGE      DEFAULT
   -------------------------------    ------------   --------   -------
   IKE DAEMON ON SC30                 ID(IKED)       PERSONAL     NO
   MY LOCAL CERTIFICATE AUTHORITY     CERTAUTH       CERTAUTH     NO
```

Authorize use of hardware cryptographic encryption

This step is optional and is required only if you are going to use the IBM eServer™ zSeries® hardware cryptographic feature to encrypt or decrypt TCP/IP packets and digital signatures.

To authorize the use of this feature, define the appropriate profiles in the CSFSERV class and give access to authorized users and daemons. The commands required are shown in Example 3-10.

Example 3-10 Authorize use of hardware cryptographic encryption

```
RDEFINE CSFSERV service-name UACC(NONE)
PERMIT service-name CLASS(CSFSERV) ID(stackname) ACCESS(READ)
PERMIT service-name CLASS(CSFSERV) ID (userid)
SETROPTS CLASSACT(CSFSERV) SETROPTS RACLIST(CSFSERV) REFRESH
```

In our setup, we did not use this feature.

For more information about RACF

Refer to the following manuals:

- *z/OS V1R8.0 Security Server RACF Security Administrator's Guide*, SA22-7683, for use of RACDCERT command
- *z/OS V1R8.0 Security Server RACF Command Language Reference*, SA22-7687, for other RACF commands

For more information about zSeries hardware cryptography

Refer to the following manuals:

- *z/OS V1R8.0 Cryptographic Services ICSF Overview*, SA22-7519
- *z/OS V1R8.0 Cryptographic Services ICSF Administrator's Guide*, SA22-7521

3.3.2 Set up the system logging daemon to log IKED messages

The system logging daemon (syslogd) manages the logging of messages and events for all of the other components, including where the log messages are written. We added the following line in our SYSLOGD configuration file /etc/syslogd.conf to route all IKED daemon logs:

```
*.IKED*.*.* /tmp/iked-sc30.log
```

3.3.3 Start IKE daemon and verify it initializes

Start IKED and make sure it comes up correctly. Example 3-11 shows the startup messages of IKED.

Example 3-11 Starting IKED

```
S IKED
$HASP100 IKED     ON STCINRDR
IEF695I START IKED     WITH JOBNAME IKED     IS ASSIGNED TO USER IKED
   , GROUP TCPGRP
$HASP373 IKED     STARTED
IEE252I MEMBER CTIIKE00 FOUND IN SYS1.PARMLIB
EZD0967I IKE RELEASE CS V1R8 SERVICE LEVEL CS060608 CREATED ON Jun  8
2006
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/security/iked.
conf
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPA   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPA
EZD1046I IKE INITIALIZATION COMPLETE
```

3.3.4 Set up Traffic Regulation Manager daemon

The Traffic Regulation Manager daemon (TRMD) is responsible for logging IP security events that are detected by the stack, including IP filter events, updates to the IP security policy, and the creation, deletion, and refresh of IPSec security associations.

For a detailed example of implementing TRMD, see 1.3, "Setting up TRMD" on page 19.

3.3.5 Update the TCP/IP stack to activate IPSec

To activate IPSec, you need to add the IPSECURITY and SOURCEVIPA option in the IPCONFIG statement in the TCP/IP Profile. In addition, you need to add IPSec rules, which are also added to the stack's profile. See Example 3-12.

Example 3-12 Our definitions for IPSEC in TCP/IP profile of stack TCPIPA in image SC30

```
IPCONFIG DATAGRAMFWD SYSPLEXROUTING IPSECURITY SOURCEVIPA
DYNAMICXCF 10.20.10.100 255.255.255.0 8
IPSEC  LOGENABLE
   ;;  OSPF protocol used by Omproute
   IPSECRULE *  *  NOLOG  PROTOCOL OSPF
   ;;  IGMP protocol used by Omproute
   IPSECRULE *  *  NOLOG  PROTOCOL 2
   ;;  DNS queries to UDP port 53
  IPSECRULE *  *  NOLOG  PROTOCOL UDP      SRCPORT *  DESTPORT 53  SECC
   ;;  Administrative access
   IPSECRULE *  9.1.1.1  LOG  PROTOCOL *
ENDIPSEC
```

3.3.6 Restrict the use of the ipsec command

The **ipsec** command is very powerful and needs to be protected from unauthorized use. We created a RACF profile for this and gave command access to the IKE daemon. Example 3-13 shows the commands to do this.

Example 3-13 Define access control for the ipsec command

```
SETROPTS GENERIC(SERVAUTH) RDEFINE SERVAUTH EZB.IPSECCMD.* UACC(NONE)
PERMIT EZB.IPSECCMD.* CLASS(SERVAUTH) ID(IKED) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

3.3.7 Install the Policy Agent

PAGENT reads the configuration files that contain the IP security policy configuration statements, checks them for errors, and installs them into the IKE daemon and the TCP/IP stack. Setting up the PAGENT is described in Chapter 1, "Policy Agent" on page 3.

Note: You need superuser authority to start PAGENT, and the PAGENT executable modules must be in an Authorized Program Facility (APF)-authorized library.

After setting it up, you need to define the IpSecConfig statement to specify the path of the policy file that contains stack-specific IPSec policy statements to PAGENT. Example 3-14 shows the IP security statements in the TCP/IP stack configuration file used by Policy Agent.

Example 3-14 The pagent /etc/sc30.tcpipa_image.conf file with IP security configured

```
# This is a file that contains statements for TCP/IP stack TCPIPA in z/OS image SC30
# This file is used by Policy Agent.
#
# IPSecConfig Statements
IPSecConfig //'TCPIP.SC30.POLICIES(IPSECA)'
```

3.3.8 Define the IPSec policies to PAGENT

IPSec provides flexible building blocks that can support a variety of configurations. You can choose from a number of protocols and encryption algorithms provided by IPSec to suit to the security requirements of your installation. You can define your IPSec security policies to PAGENT in one of two ways:

- Manually code all of the required policy statements to create a configuration file in a z/OS UNIX file or an MVS data set.
- Use the IBM Configuration Assistant for z/OS Communication Server to create the IP security configuration file.

We used the IBM Configuration Assistant for z/OS Communication Server to create the IP security configuration file for our implementation scenarios.

3.3.9 Commands used to administer IP security

The commands used to administer IP security are:

- **ipsec**: Use this z/OS UNIX System Services command to display information about active filters and security associations, and to control aspects of security association negotiation. Authority to use this command is controlled through z/OS security server (RACF).
- **pasearch**: Use this command to display Policy Agent information that is defined in the Policy Agent configuration files, including IP security and other types of policies. If the user is not a superuser, authority is controlled though RACF.
- **MODIFY**: Use this console command to make IKE daemon reread the IKED configuration file or to make Policy Agent reread the policy configuration agent files.
- **Netstat**: Use this command to display IPSECURITY enablement for a particular stack or to display the SecurityClass (SECCLASS) for a specific interface.

For more detailed information about the syntax and usage of those commands, refer to *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781.

3.3.10 Using the IBM configuration Assistant for z/OS

IBM provides a graphical user interface (GUI) called IBM Configuration Assistant for z/OS Communication Server to help you to code your security policies. This is a Windows-based interface that you can download from the IBM Web site:

`http://www.ibm.com/software/network/commserver/zos/support/`

When your policy has been coded using this software, it can then be sent via File Transfer Protocol (FTP) to your z/OS system to be used by the Policy Agent.

The IBM Configuration Assistant for z/OS Communication Server gives you the option to create policies for a simple scenario to test your IPSec for the first time. In this section, we explain how to set up a *quick dynamic tunnel* using the IBM Configuration Assistant for z/OS Communication Server.

> **Attention:** The IBM Configuration Assistant for z/OS Communication Server is constantly being updated for enhancements. Therefore, you should download the latest version before you use it.

The IBM Configuration Assistant for z/OS Communication Server provides an option to set up a quick dynamic tunnel with the following characteristics:

- All IP packets will be encrypted.
- The dynamic tunnel is activated by the outbound traffic flow without user intervention.
- Uses *transport mode* encapsulation (hence, it does not encapsulate the original IP header as would be done using *tunnel mode*).
- Uses shared key authentication for IKE peers.
- Uses AES (or DES) encryption for both phase 1 (IKE) and phase 2 (IPSec) tunnels.
- Uses ESP HMAC MD5 authentication.

The only parameters that you need to supply are the IP addresses of the communicating stacks.

3.4 Configuring quick dynamic tunnels

The option of Quick Dynamic Tunnels Configuration creates two basic configuration files for IP security between two z/OS images. The new configuration files do not contain any connectivity rules and therefore disallow any connectivity between the logical partitions or LPARs (because IP security disallows any traffic by default). You can FTP those configuration files to the z/OS images and change them manually in the z/OS environment.

To set up the quick dynamic tunnel configuration:

1. Start the IBM Configuration Assistant for z/OS Communication Server and click **File** from the top left corner of the window. Select the **Quick Dynamic Tunnel Configuration** option, as shown in Figure 3-2.

Figure 3-2 Quick Dynamic Tunnel Configuration

2. Enter the IP addresses of the source and destination endpoints (TCP/IP stacks) between which you want to set up the tunnel, as shown in Figure 3-3. Click the **Install Configuration Files** option.

Figure 3-3 Entering addresses of the tunnel endpoints

3. This shows the two configuration files the Configuration Assistant has created for you, one per stack, as shown in Figure 3-4. These files contain the security policies for each stack created by the Configuration Assistant. You can now FTP the definitions to the respective z/OS systems. For this, select the first file and then click the **FTP** option.

Figure 3-4 The configuration files created

4. This leads you to the window shown in Figure 3-5. Enter the host name (in our case, SC30) of the z/OS system where you want this file to be sent via FTP along with your user ID and password on that system.

Figure 3-5 FTP of stack number 1 configuration

5. Similarly, FTP the other file to the other system (in our case, SC31), as shown in Figure 3-6.

Figure 3-6 FTP of stack number 2 configuration

6. You can click the **Show Configuration File** option to display the policy created by the z/OS Configuration Assistant. This is shown in Example 3-15 for stack 1.

Example 3-15 Quick Dynamic Tunnel Policy for stack 1

```
#-------------------------------------------------------
# Quick-Start IP Security policy
# Created by the IBM Configuration Assistant for z/OS Communications Server
# Version 1 Release 8
# Date Created = Mon Jul 31 10:26:53 EDT 2006
#-------------------------------------------------------
IpFilterPolicy
{
   PreDecap                    off
   FilterLogging               on
   AllowOnDemand               yes

   IpFilterRule                QuickStartRule1
   {
      IpSourceAddr             10.10.1.230
      IpDestAddr               10.10.1.241
      IpService
      {
         SourcePortRange       500
         DestinationPortRange  500
         Protocol              udp
```

```
         Direction              bidirectional
         Routing                local
      }
      IpGenericFilterActionRef  permit
   }

   IpFilterRule                 QuickStartRule2
   {
      IpSourceAddr              10.10.1.230
      IpDestAddr                10.10.1.241
      IpService
      {
         Direction              bidirectional
         Routing                local
      }
      IpGenericFilterActionRef  ipsec
      IpDynVpnActionRef         TransportMode
   }
}

KeyExchangePolicy
{
   KeyExchangeRule              QuickStart_KeyExRule
   {
      LocalSecurityEndpoint
      {
         Identity               IpAddr 10.10.1.230
         Location               10.10.1.230
      }
      RemoteSecurityEndpoint
      {
         Identity               IpAddr 10.10.1.241
         Location               10.10.1.241
      }
      KeyExchangeActionRef      QuickStart_KeyExAction
      SharedKey                 Ascii TheEagleHasLanded
   }
}

#-------------------------------------------------------
# Reusable actions
#-------------------------------------------------------
IpGenericFilterAction           permit
{
   IpFilterAction               permit
}

IpGenericFilterAction           ipsec
{
   IpFilterAction               ipsec
   IpFilterLogging              yes LogDeny
}

KeyExchangeAction               QuickStart_KeyExAction
{
   KeyExchangeOffer
   {
      HowToAuthPeers            PreSharedKey
   }
}
```

```
IpDynVpnAction                  TransportMode
{
   IpDataOffer
   {
      HowToEncap                transport
   }
}
```

Example 3-16 shows the configuration file that is generated for stack 2.

Example 3-16 Quick Dynamic Tunnel Policy for stack 2

```
#-------------------------------------------------------
# Quick-Start IP Security policy
# Created by the IBM Configuration Assistant for z/OS Communications Server
# Version 1 Release 8
# Date Created = Mon Jul 31 17:47:10 EDT 2006
#-------------------------------------------------------
IpFilterPolicy
{
   PreDecap                     off
   FilterLogging                on
   AllowOnDemand                yes

   IpFilterRule                 QuickStartRule1
   {
      IpSourceAddr              10.10.1.241
      IpDestAddr                10.10.1.230
      IpService
      {
         SourcePortRange        500
         DestinationPortRange   500
         Protocol               udp
         Direction              bidirectional
         Routing                local
      }
      IpGenericFilterActionRef  permit
   }

   IpFilterRule                 QuickStartRule2
   {
      IpSourceAddr              10.10.1.241
      IpDestAddr                10.10.1.230
      IpService
      {
         Direction              bidirectional
         Routing                local
      }
      IpGenericFilterActionRef  ipsec
      IpDynVpnActionRef         TransportMode
   }
}
```

```
KeyExchangePolicy
{
   KeyExchangeRule              QuickStart_KeyExRule
   {
      LocalSecurityEndpoint
      {
         Identity               IpAddr 10.10.1.241
         Location               10.10.1.241
      }
      RemoteSecurityEndpoint
      {
         Identity               IpAddr 10.10.1.230
         Location               10.10.1.230
      }
      KeyExchangeActionRef      QuickStart_KeyExAction
      SharedKey                 Ascii TheEagleHasLanded
   }
}

#-------------------------------------------------------
# Reusable actions
#-------------------------------------------------------
IpGenericFilterAction           permit
{
   IpFilterAction               permit
}

IpGenericFilterAction           ipsec
{
   IpFilterAction               ipsec
   IpFilterLogging              yes LogDeny
}

KeyExchangeAction               QuickStart_KeyExAction
{
   KeyExchangeOffer
   {
      HowToAuthPeers            PreSharedKey
   }
}
IpDynVpnAction                  TransportMode
{
   IpDataOffer
   {
      HowToEncap                transport
   }
}
```

7. You can now load these policies in the PAGENT policy file, and then refresh PAGENT to pick up this policy using the operator command F PAGENT,REFRESH.

When the policy is activated, it sets up the tunnel for you between the systems and provides security for the traffic flowing between them.

3.5 Implementing IPSec between two z/OS systems

In this scenario, we show how to set up a VPN tunnel between two z/OS systems.

Figure 3-7 VPN traffic between two z/OS systems

We used the IBM Configuration Assistant for z/OS Communication Server to set up a dynamic tunnel between the two z/OS systems. In this section, we go through the step-by-step process of defining the policy to set up this tunnel.

3.5.1 Setting up the policy using IBM Configuration Assistant

To set up the policy using IBM Configuration Assistant, perform the following steps:

1. Start the IBM Configuration Assistant for z/OS Communication Server. Click **File** → **Open** → **Create a New Configuration**.
2. The Information window shown in Figure 3-8 opens. Click **OK**.

Figure 3-8 The Configuration Assistant Information window

3. Enter a location and file name for the new configuration. Click **Save**.

4. The window shown in Figure 3-9 opens. Select the **IPSec only** option and click **OK**.

Figure 3-9 IBM Configuration Assistant for z/OS Communication Server Welcome window

You should now be presented with a window that has the IPSec only option selected, as shown in Figure 3-10.

Figure 3-10 Current configuration window

The steps for configuring an IPSec policy between two LPARs

Perform the following steps to configure an IPSec policy between two LPARs:

1. Add requirement map objects for FTP, DNS, PING, Omproute and all the other traffic objects that you want to use in your z/OS image. These objects are used for configuring the TCP/IP stacks.
2. Add a new z/OS image.
3. Add the TCP/IP stack for the new z/OS image.
4. Add connectivity rules for the TCP/IP stack.
5. Add another new z/OS image and repeat steps 3 and 4 for the new image.

We created two z/OS images, A23 and A24. For each image, we configured one TCP/IP stack, and for each TCP/IP stack we defined two connectivity rules: one for omproute and one for all the other traffic. Use the following steps to configure IPSec between two LPARs.

Step 1: Adding requirement map objects

To add requirement map objects:

1. Select **IPSec Work with Reusable Objects** → **Requirement Maps**, as shown in Figure 3-11. Click **Add**.

Figure 3-11 Requirement Maps window

2. Select the **All_other_Traffic traffic** descriptor, as shown in Figure 3-12. Click **Remove**.

Figure 3-12 Requirement Map window: Selecting All_Other_Traffic traffic descriptor

3. Type a name and description to the requirement map. Select **OMPROUTE-IP_V4** from the Objects list, as shown in Figure 3-13. Click **Add**.

Figure 3-13 Requirement Map window: Selecting Omproute traffic descriptor for IPv4 network

4. Click the **IPSec - Security Level** list box and change the security level to **Permit**, as shown in Figure 3-14. Click **OK**, and then click **Proceed**.

   To create the rule for IPSec connectivity, click **Add**.

Figure 3-14 Requirement Map window: Changing IPSec security level to Permit

5. Type a name and description for the rule and change the IPSec security level to **IPSec_Silver**, as shown in Figure 3-15. Click **OK**, and then click **Proceed**.

Figure 3-15 Requirement Map: Adding IPSec requirement map object for all types of traffic

The objects which we created now will be used for creating the connectivity rules, in step 4.

> **Note:** IBM provides five built-in security level objects: Permit, Deny, IPSec_Bronze, IPSec_Silver, and IPSec_Gold. The first two objects are for non-secured connections, while the other three objects are used for secured connections. Each object has different definition, which can be viewed from the **IPSec** → **Work with Reusable Objects** → **Security Levels** panel. You can also define new security objects from this panel.

Step 2: Adding a new z/OS image

To add a new z/OS image:

1. Select **IPSec** → **Work with z/OS Images** from the navigation tree. Click **Add a New z/OS Image**, and then click **Next**.
2. This opens the window shown in Figure 3-16. Enter the name of the z/OS image (in our case, SC30) and its description. Specify that you want to set up dynamic tunnels for this image by selecting the **Yes, this Image will have dynamic tunnels** radio button. Click **Next**.

Figure 3-16 Adding our z/OS image for SC30

3. This opens the window shown in Figure 3-17. Click **Next**, and then click **Finish**.

Figure 3-17 Setting our CA certificate label

4. Proceed to the next step and click **Yes**, which leads you to the New TCP/IP Stack Wizard window.

Step 3: Adding the TCP/IP stack for the new z/OS image

To add the TCP/IP stack for the new z/OS image:

1. In the Welcome window, click **Next**.
2. Specify the name of the TCP/IP stack that you want to configure in the z/OS image. Our stack name is TCPIPA. Select the **Yes, this Stack will have dynamic tunnels** radio button to specify that you want to set up dynamic tunnel for this stack, as shown in Figure 3-18. Click **Next**.

Figure 3-18 New TCP/IP stack name

3. The window shown in Figure 3-19 opens. There are two options for configuring a local identity:
   - Single tunnel identity: One identity for all IP addresses on the TCP/IP stack.
   - Separate tunnel identities: One identity for each IP address on the TCP/IP stack.

   We set up a single tunnel for the stack. We selected the **I want to use a single identity for all IP addresses** radio button. You can identify the stack in four different ways, as shown by the radio buttons in the window. We identified our stack by its IP address, 10.10.1.230.

   Click **Next**.

Figure 3-19 Identifying the local dynamic tunnel

4. This opens the window shown in Figure 3-20. This window is used to specify the log levels. For our test, we selected the radio buttons to enable filter logging and to log all implicit deny events. When everything is working, you can disable the logging later. This window is also used to specify the Network Address Translation (NAT) Traversal Policy. For the IPSec test, we disallowed NAT. Also, we decided not to filter the IPSec headers.

   Click **Next**, and then **Finish**.

Figure 3-20 IPSec log level settings

5. This leads to the Connectivity wizard.

   When you are asked `Do you want to add a Connectivity Rule now?`, click **Yes**.

Step 4: Adding connectivity rules for the TCP/IP stack

To add connectivity rules for the TCP/IP stack:

1. In the Welcome window, click **Next**.
2. We created two connectivity rules:
   - Omproute: A rule that permits omproute traffic between all IPv4 addresses.
   - All_Traffic_Silver: A rule that forces all other traffic to be secured, using IP security with security level IPSec_Silver.

   In this way, we make sure that each connection between the two LPARs are encrypted and therefore secured.

First, create the Omproute connectivity rule. In the Network Topology window, select the first radio button, as shown in Figure 3-21.

Click **Next**.

Figure 3-21 Connectivity Rule: Network Topology window - Defining non-secure network topology

3. In the Connectivity Rule: Data Endpoints window, select **All V4 addresses** for Source data endpoint and for Destination data endpoint. Enter a name for the connectivity rule, as shown in Figure 3-22. Click **Next**.

Figure 3-22 Connectivity Rule : Data Endpoints panel

4. In the Connectivity Rule: Select Requirement Map window, select **Omproute** requirement map object from the list, as shown in Figure 3-23.

> **Note:** The Select Requirement Map window shows all the requirement map objects that are defined in the current configuration file, including the ones we defined (see “Step 1: Adding requirement map objects” on page 97) and the IBM-supplied samples.
>
> **Attention:** Adding a new Requirement Map object from this window affects the list of all defined requirement map objects. That is, the new object, which is added from this panel, is also added to the general list, and therefore to the other z/OS images and to the other TCP/IP stacks as well. This is also true for any modification that you make to the objects.
>
> **Our recommendation:** Use the Requirement Map window under Work with Reusable Objects to make modifications to the requirement map objects. For more information, see “Step 1: Adding requirement map objects” on page 97.

Click **Next** and then **Finish**.

Figure 3-23 Connectivity Rule: Select Requirement Map window

5. This brings you to the TCP/IP Stack Settings window shown in Figure 3-24. To add the second rule for all other traffic, click **Add** and then **Next**.

Figure 3-24 TCP/IP Stack Settings window: After adding the first connectivity rule for stack TCPIPA in SC30

6. In the Connectivity Rule: Network Topology window, select **Host to Host** topology option, under “This Connectivity Rule will contain a Security Level using IPSec tunnels”, as shown in Figure 3-25. Click **Next**.

Figure 3-25 Connectivity Rule: Network Topology window: Defining host-to-host connectivity

7. This opens the Connectivity Rule: Data Endpoints window shown in Figure 3-26. Specify the endpoints of the connection by entering the IP addresses of the z/OS images and give a name for the connection. Click **Next**.

Figure 3-26 Connectivity Rule: Data Endpoints window - Defining the data endpoints

8. In the Connectivity Rule: Select Requirement object window, select **All_Traffic_Silver** requirement map object, as shown in Figure 3-27. Click **Next**.

Figure 3-27 Select Requirement Map window: Selecting the secured traffic object

9. In the Connectivity Rule: Remote Security Endpoint Information window, select the remote IKE identity for the remote endpoint. We chose **IP address**. If you choose the same type of identity, enter the IP address of the remote z/OS image of this rule (in our case: 10.10.1.241). There are two ways to authenticate the remote IKE peers:
   - RSA signature
   - Shared key, in which you can specify a shared key as American Standard Code for Information Interchange (ASCII) string, an Extended Binary Coded Decimal Interchange Code (EBCDIC) string, or a hexadecimal string.

For our test environment, we chose **Shared key** and entered an EBCIDIC string, as shown in Figure 3-28. Click **Next**.

Figure 3-28 Connectivity Rule: Remote Security Endpoint Information window

10. Click **Finish**, as shown in Figure 3-29.

Figure 3-29 Additional Settings for a Requirement Map object in a specific stack

Repeat the same procedure to add all the other objects that you want to select for the current stack.

Note: There is an importance to the order of the connectivity rules in a stack. When the Policy Agent is running and there is an attempt to send packets from or to the z/OS image, the first rule in the list is checked first and if there is a match with the definition of this rule, it will be used. If there is no match, the second rule will be checked and so on.

11. In TCP/IP Stack Settings window, click **OK**. See Figure 3-30.

Figure 3-30 Our completed connectivity rule

This completes the creation of the policy for one endpoint of the tunnel. You can now FTP the policy to the z/OS system. Right-clicking **Image-SC30** in the Configuration Assistant Navigation Tree area gives you the option to install the configuration file. Clicking **Install Configuration Files** option gives you further options to view the policy, and then FTP it to the z/OS system, as shown in Figure 3-31.

Note: You should click the **Health Check** button from the Connectivity Rules tab before you install the configuration file to the z/OS image. The health check reports on errors in your policy and we found it helpful.

Figure 3-31 Installation: Image-SC30 panel

12. Select **TCPIPA - IPSec: Policy Agent Stack Configuration** in the configuration files list, as shown in Figure 3-32. To install the new configuration to the z/OS image, click **FTP** and provide the necessary information.

Figure 3-32 Installation panel: Selecting our configuration

The policy file we created for SC30 z/OS image is shown in Example 3-17.

Example 3-17 IPSec policy created for SC30

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC30
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Mon Aug 07 17:31:54 EDT 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}

IpGenericFilterAction          IpSec~LogNo
{
  IpFilterAction               IpSec
  IpFilterLogging              No
}

KeyExchangeOffer               KEO~5
{
  HowToEncrypt                 DES
  HowToAuthMsgs                SHA1
  HowToAuthPeers               PresharedKey
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}
```

```
IpDataOffer                     IPSec__Silver~R
{
  HowToEncap                    Transport
  HowToEncrypt                  DES
  HowToAuth                     ESP Hmac_Sha
  RefreshLifetimeProposed       240
  RefreshLifetimeAccepted       120 480
  RefreshLifesizeProposed       None
  RefreshLifesizeAccepted       None
}

IpDataOffer                     IPSec__Silver~R~6
{
  HowToEncap                    Transport
  HowToEncrypt                  3DES
  HowToAuth                     ESP Hmac_Sha
  RefreshLifetimeProposed       240
  RefreshLifetimeAccepted       120 480
  RefreshLifesizeProposed       None
  RefreshLifesizeAccepted       None
}

IpDataOffer                     IPSec__Silver~R~7
{
  HowToEncap                    Transport
  HowToEncrypt                  AES
  HowToAuth                     ESP Hmac_Sha
  RefreshLifetimeProposed       240
  RefreshLifetimeAccepted       120 480
  RefreshLifesizeProposed       None
  RefreshLifesizeAccepted       None
}

IpService                       OMPROUTE-IP_V4
{
  Protocol                      OSPF
  Type                          Any
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~1
{
  Protocol                      IGMP
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~2
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~3
{
```

```
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~4
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

## NOTE -- Generated IpService IKE~Gen
IpService                       IKE~Gen
{
  Protocol                      UDP
  SourcePortRange               500
  DestinationPortRange          500
  Direction                     BiDirectional
  Routing                       Local
}

IpService                       All_other_traffic
{
  Protocol                      All
  Direction                     BiDirectional
  Routing                       Local
}

IpDynVpnAction                  IPSec__Silver
{
  Initiation                    Either
  VpnLife                       1440
  Pfs                           None
  IpDataOfferRef                IPSec__Silver~R
  IpDataOfferRef                IPSec__Silver~R~6
  IpDataOfferRef                IPSec__Silver~R~7
}
##
## Connectivity Rule Omproute combines the following items:
##   Local data endpoint       All4
##   Remote data endpoint      All4
##   Topology                  None (Permit/Deny only)
##   Requirement Map           Omproute
##     OMPROUTE-IP_V4            => Permit

IpFilterRule                    Omproute~1
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    Omproute~2
{
```

```
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~1
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~3
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~2
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~4
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~3
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~5
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~4
  IpGenericFilterActionRef     Permit~LogNo
}
##
## Connectivity Rule All_Traffic_Silver combines the following items:
##   Local data endpoint       All_Traffic_Silver~ADR~1
##   Remote data endpoint      All_Traffic_Silver~ADR~2
##   Topology                  HH
##   Requirement Map           All_Traffic_Silver
##     All_other_traffic         => IPSec__Silver

IpAddr                         All_Traffic_Silver~ADR~1
{
  Addr                         10.10.1.230
}

IpAddr                         All_Traffic_Silver~ADR~2
{
  Addr                         10.10.1.241
}

LocalSecurityEndpoint          All_Traffic_Silver~LSE~4
{
  Identity                     IpAddr 10.10.1.230
  LocationRef                  All_Traffic_Silver~ADR~1
}

RemoteSecurityEndpoint         All_Traffic_Silver~RSE~3
{
  Identity                     IpAddr 10.10.1.241
  LocationRef                  All_Traffic_Silver~ADR~2
}

KeyExchangeRule                All_Traffic_Silver~5
```

```
{
  LocalSecurityEndpointRef     All_Traffic_Silver~LSE~4
  RemoteSecurityEndpointRef    All_Traffic_Silver~RSE~3
  KeyExchangeActionRef         All_Traffic_Silver
  SharedKey                    Ebcdic abcde
}

KeyExchangeAction              All_Traffic_Silver
{
  HowToInitiate                Main
  HowToRespond                 Either
  KeyExchangeOfferRef          KEO~5
  AllowNat                     No
}

## NOTE -- Generated IpFilterRule All_Traffic_Silver~6
IpFilterRule                   All_Traffic_Silver~6
{
  IpSourceAddrRef              All_Traffic_Silver~ADR~1
  IpDestAddrRef                All_Traffic_Silver~ADR~2
  IpServiceRef                 IKE~Gen
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   All_Traffic_Silver~7
{
  IpSourceAddrRef              All_Traffic_Silver~ADR~1
  IpDestAddrRef                All_Traffic_Silver~ADR~2
  IpServiceRef                 All_other_traffic
  IpGenericFilterActionRef     IpSec~LogNo
  IpDynVpnActionRef            IPSec__Silver
}

KeyExchangePolicy
{
  AllowNat                     No
  KeyExchangeRuleRef           All_Traffic_Silver~5
}

IpFilterPolicy
{
  PreDecap                     OFF
  FilterLogging                ON
  IpFilterLogImplicit          No
  AllowOnDemand                Yes
  IpFilterRuleRef              Omproute~1
  IpFilterRuleRef              Omproute~2
  IpFilterRuleRef              Omproute~3
  IpFilterRuleRef              Omproute~4
  IpFilterRuleRef              Omproute~5
  IpFilterRuleRef              All_Traffic_Silver~6
  IpFilterRuleRef              All_Traffic_Silver~7
}
```

Step 5: Adding another new z/OS image and repeating steps 3 - 4

Similarly, follow steps 1 - 4 (see “Step 1: Adding requirement map objects” on page 97 through “Step 4: Adding connectivity rules for the TCP/IP stack” on page 105) to create the second endpoint with TCP/IP stack and connectivity rules. FTP the policies to the respective

z/OS systems. Update the Policy Agent with the MODIFY PAGENT,REFRESH command and the tunnel will become active.

We created another z/OS image called SC31 in the configuration file with TCP/IP stack named TCPIPA and the same connectivity rules, as shown in Figure 3-33.

Figure 3-33 Our final IPSec configuration between two LPARS

The policy file we created for SC31 z/OS image is shown in Example 3-18.

Example 3-18 IPSec policy created for SC31

```
###
## IPSec Policy Agent Configuration file for:
##     Image: SC31
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Mon Aug 07 17:58:31 EDT 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}

IpGenericFilterAction          IpSec~LogYes
{
  IpFilterAction               IpSec
```

```
  IpFilterLogging              Yes
}

KeyExchangeOffer               KEO~5
{
  HowToEncrypt                 DES
  HowToAuthMsgs                SHA1
  HowToAuthPeers               PresharedKey
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~6
{
  HowToEncap                   Transport
  HowToEncrypt                 3DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~7
{
  HowToEncap                   Transport
  HowToEncrypt                 AES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpService                      OMPROUTE-IP_V4
{
  Protocol                     OSPF
  Type                         Any
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~1
{
```

```
  Protocol                      IGMP
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~2
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~3
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~4
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

## NOTE -- Generated IpService IKE~Gen
IpService                       IKE~Gen
{
  Protocol                      UDP
  SourcePortRange               500
  DestinationPortRange          500
  Direction                     BiDirectional
  Routing                       Local
}

IpService                       All_other_traffic
{
  Protocol                      All
  Direction                     BiDirectional
  Routing                       Local
}

IpDynVpnAction                  IPSec__Silver
{
  Initiation                    Either
  VpnLife                       1440
  Pfs                           None
  IpDataOfferRef                IPSec__Silver~R
  IpDataOfferRef                IPSec__Silver~R~6
  IpDataOfferRef                IPSec__Silver~R~7
}
##
## Connectivity Rule Omproute combines the following items:
```

```
##   Local data endpoint       All4
##   Remote data endpoint      All4
##   Topology                  None (Permit/Deny only)
##   Requirement Map           Omproute
##     OMPROUTE-IP_V4            => Permit

IpFilterRule                   Omproute~1
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~2
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~1
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~3
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~2
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~4
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~3
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~5
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~4
  IpGenericFilterActionRef     Permit~LogNo
}
##
## Connectivity Rule All_Traffic_Silver combines the following items:
##   Local data endpoint       All_Traffic_Silver~ADR~1
##   Remote data endpoint      All_Traffic_Silver~ADR~2
##   Topology                  HH
##   Requirement Map           All_Traffic_Silver
##     All_other_traffic         => IPSec__Silver

IpAddr                         All_Traffic_Silver~ADR~1
{
  Addr                         10.10.1.241
}

IpAddr                         All_Traffic_Silver~ADR~2
{
```

```
  Addr                          10.10.1.230
}

LocalSecurityEndpoint           All_Traffic_Silver~LSE~4
{
  Identity                      IpAddr 10.10.1.241
  LocationRef                   All_Traffic_Silver~ADR~1
}

RemoteSecurityEndpoint          All_Traffic_Silver~RSE~3
{
  Identity                      IpAddr 10.10.1.230
  LocationRef                   All_Traffic_Silver~ADR~2
}

KeyExchangeRule                 All_Traffic_Silver~5
{
  LocalSecurityEndpointRef      All_Traffic_Silver~LSE~4
  RemoteSecurityEndpointRef     All_Traffic_Silver~RSE~3
  KeyExchangeActionRef          All_Traffic_Silver
  SharedKey                     Ebcdic abcde
}

KeyExchangeAction               All_Traffic_Silver
{
  HowToInitiate                 Main
  HowToRespond                  Either
  KeyExchangeOfferRef           KEO~5
  AllowNat                      No
}

## NOTE -- Generated IpFilterRule All_Traffic_Silver~6
IpFilterRule                    All_Traffic_Silver~6
{
  IpSourceAddrRef               All_Traffic_Silver~ADR~1
  IpDestAddrRef                 All_Traffic_Silver~ADR~2
  IpServiceRef                  IKE~Gen
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    All_Traffic_Silver~7
{
  IpSourceAddrRef               All_Traffic_Silver~ADR~1
  IpDestAddrRef                 All_Traffic_Silver~ADR~2
  IpServiceRef                  All_other_traffic
  IpGenericFilterActionRef      IpSec~LogYes
  IpDynVpnActionRef             IPSec__Silver
}

KeyExchangePolicy
{
  AllowNat                      No
  KeyExchangeRuleRef            All_Traffic_Silver~5
}

IpFilterPolicy
{
  PreDecap                      OFF
  FilterLogging                 ON
  IpFilterLogImplicit           Yes
```

```
  AllowOnDemand                Yes
  IpFilterRuleRef              Omproute~1
  IpFilterRuleRef              Omproute~2
  IpFilterRuleRef              Omproute~3
  IpFilterRuleRef              Omproute~4
  IpFilterRuleRef              Omproute~5
  IpFilterRuleRef              All_Traffic_Silver~6
  IpFilterRuleRef              All_Traffic_Silver~7
}
```

3.5.2 Verification of IPSec between two z/OS images

This section describes the steps needed to verify that the IP security configuration is working properly. The steps are as follows:

- Check that IPSECURITY and SOURCEVIPA is configured to a TCP/IP stack in both z/OS images.
- Check that Policy Agent is updated with IPSEC policy.
- Display the active IP security policy.
- Try to run FTP between the z/OS images.
- Display the IKED tunnels and dynamic (or manual, depending on your configuration) tunnels.

Checking TCP/IP stack is configured for IP security

Run the following command:

```
netstat -p TCPIPA -f
```

The first parameter, -p, allows a specific stack to be queried. Check that IpSecurity field and SourceVipa field under IP Configuration Table is equal to Yes. The beginning of the **netstat** output should look something like what is shown in Example 3-19.

Example 3-19 netstat -f for stack TCPIPA, on SC31 z/OS image

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          12:07:48
TCP Configuration Table:
DefaultRcvBufSize:  00016384  DefaultSndBufSize: 00016384
DefltMaxRcvBufSize: 00262144
MaxReTransmitTime:  120.000   MinReTransmitTime: 0.500
RoundTripGain:      0.125     VarianceGain:      0.250
VarianceMultiplier: 2.000     MaxSegLifeTime:    30.000
DefaultKeepALive:   00000120  DelayAck:          Yes
RestrictLowPort:    Yes       SendGarbage:       No
TcpTimeStamp:       Yes       FinWait2Time:      600
TTLS:               No

UDP Configuration Table:
DefaultRcvBufSize: 00065535  DefaultSndBufSize: 00065535
CheckSum:          Yes
RestrictLowPort:   Yes       UdpQueueLimit:     Yes

IP Configuration Table:
Forwarding: Yes    TimeToLive: 00064  RsmTimeOut:  00060
IpSecurity: Yes
ArpTimeout: 01200  MaxRsmSize: 65535  Format:      Long
IgRedirect: Yes    SysplxRout: Yes    DoubleNop:   No
```

```
StopClawEr: No     SourceVipa: Yes
MultiPath:  No     PathMtuDsc: No     DevRtryDur:  0000000090
DynamicXCF: Yes
  IpAddr: 10.20.10.101     SubNet: 255.255.255.0    Metric: 08
  SecClass: 255
SecClass: 255
IQDIORoute: No
TcpStackSrcVipa: No
```

Checking that Policy Agent is active with IP security policy

The first thing you need to check is that the Policy Agent has read in the current policies.

After you FTP the configuration file to the z/OS image, the following command is used to refresh the PAGENT (where pagent is the started task name for the Policy Agent):

```
MODIFY PAGENT,REFRESH
```

If no changes have been made since the last time the polices were read, then a message such as follows should be seen in response:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : NONE
```

If changes have been made, then instead, a message such as the following should be seen:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

Displaying the active IP security policy

With TRMD running, syslogd is the repository for all Policy Agent and IKE daemon messages.

```
ipsec -p TCPIPA -f display -a YO
```

The first parameter, -p, allows specific stacks policies to be queried. The second parameter indicates filters are to be displayed, and the third parameter indicates a list of all dynamic anchor filters. The display output should look something like what is shown in Example 3-20.

If the Source field shows Stack Policy, the IP security policy is installed and active.

Example 3-20 ipsec -f display for stack TCPIPA, on SC31 z/OS image

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Tue Aug  8 10:13:57 2006
Primary:  Filter          Function: Display            Format:   Detail
Source:   Stack Policy    Scope:    Current            TotAvail: 18
Logging:  On              Predecap: Off                DVIPSec:  No
NatKeepAlive:  20

FilterName:                   Omproute~1
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
```

```
Protocol:                    OSPF(89)
ICMPType:                    n/a
ICMPCode:                    n/a
OSPFType:                    All
TCPQualifier:                n/a
ProtocolGranularity:         Rule
SourceAddress:               0.0.0.0
SourceAddressPrefix:         0
SourceAddressRange:          n/a
SourceAddressGranularity:    Packet
SourcePort:                  n/a
SourcePortRange:             n/a
SourcePortGranularity:       Rule
DestAddress:                 0.0.0.0
DestAddressPrefix:           0
DestAddressRange:            n/a
DestAddressGranularity:      Packet
DestPort:                    n/a
DestPortRange:               n/a
DestPortGranularity:         Rule
OrigRmtConnPort:             n/a
RmtIDPayload:                n/a
RmtUdpEncapPort:             n/a
************************************************************************
FilterName:                  Omproute~1
FilterNameExtension:         2
GroupName:                   n/a
LocalStartActionName:        n/a
VpnActionName:               n/a
TunnelID:                    0x00
Type:                        Generic
State:                       Active
Action:                      Permit
Scope:                       Both
Direction:                   Inbound
OnDemand:                    n/a
SecurityClass:               0
Logging:                     None
Protocol:                    OSPF(89)
ICMPType:                    n/a
ICMPCode:                    n/a
OSPFType:                    All
TCPQualifier:                n/a
ProtocolGranularity:         Rule
SourceAddress:               0.0.0.0
SourceAddressPrefix:         0
SourceAddressRange:          n/a
SourceAddressGranularity:    Packet
SourcePort:                  n/a
SourcePortRange:             n/a
SourcePortGranularity:       Rule
DestAddress:                 0.0.0.0
DestAddressPrefix:           0
DestAddressRange:            n/a
DestAddressGranularity:      Packet
DestPort:                    n/a
DestPortRange:               n/a
DestPortGranularity:         Rule
OrigRmtConnPort:             n/a
RmtIDPayload:                n/a
```

```
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~2
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     IGMP(2)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~2
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     IGMP(2)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
```

```
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~3
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   520
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     1024
DestPortRange:                65535
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~3
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
```

```
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   1024
SourcePortRange:              65535
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     520
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~4
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   1024
SourcePortRange:              65535
```

```
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     520
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~4
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   520
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     1024
DestPortRange:                65535
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~5
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
```

```
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   520
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     520
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   Omproute~5
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   520
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     520
```

```
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   All_Traffic_Silver~6
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                10.10.1.241
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   500
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  10.10.1.230
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     500
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   All_Traffic_Silver~6
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      None
Protocol:                     UDP(17)
ICMPType:                     n/a
```

```
ICMPCode:                       n/a
OSPFType:                       n/a
TCPQualifier:                   n/a
ProtocolGranularity:            Rule
SourceAddress:                  10.10.1.230
SourceAddressPrefix:            n/a
SourceAddressRange:             n/a
SourceAddressGranularity:       Packet
SourcePort:                     500
SourcePortRange:                n/a
SourcePortGranularity:          Rule
DestAddress:                    10.10.1.241
DestAddressPrefix:              n/a
DestAddressRange:               n/a
DestAddressGranularity:         Packet
DestPort:                       500
DestPortRange:                  n/a
DestPortGranularity:            Rule
OrigRmtConnPort:                n/a
RmtIDPayload:                   n/a
RmtUdpEncapPort:                n/a
************************************************************************
FilterName:                     All_Traffic_Silver~7
FilterNameExtension:            1
GroupName:                      n/a
LocalStartActionName:           n/a
VpnActionName:                  IPSec__Silver
TunnelID:                       Y0
Type:                           Dynamic Anchor
State:                          Active
Action:                         Permit
Scope:                          Local
Direction:                      Outbound
OnDemand:                       Yes
SecurityClass:                  0
Logging:                        All
Protocol:                       All
ICMPType:                       n/a
ICMPCode:                       n/a
OSPFType:                       n/a
TCPQualifier:                   n/a
ProtocolGranularity:            Rule
SourceAddress:                  10.10.1.241
SourceAddressPrefix:            n/a
SourceAddressRange:             n/a
SourceAddressGranularity:       Packet
SourcePort:                     All
SourcePortRange:                n/a
SourcePortGranularity:          Rule
DestAddress:                    10.10.1.230
DestAddressPrefix:              n/a
DestAddressRange:               n/a
DestAddressGranularity:         Packet
DestPort:                       All
DestPortRange:                  n/a
DestPortGranularity:            Rule
OrigRmtConnPort:                n/a
RmtIDPayload:                   n/a
RmtUdpEncapPort:                n/a
************************************************************************
```

```
FilterName:                   All_Traffic_Silver~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                10.10.1.230
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  10.10.1.241
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   DenyAllRule_Generated___________Inbnd
FilterNameExtension:          n/a
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
```

```
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   DenyAllRule_Generated___________Outbnd
FilterNameExtension:          n/a
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Outbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   DenyAllRule_Generated__________Inbnd_v6
FilterNameExtension:          n/a
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
```

```
Type:                        Generic
State:                       Active
Action:                      Deny
Scope:                       Both
Direction:                   Inbound
OnDemand:                    n/a
SecurityClass:               0
Logging:                     All
Protocol:                    All
ICMPType:                    n/a
ICMPCode:                    n/a
OSPFType:                    n/a
TCPQualifier:                n/a
ProtocolGranularity:         Rule
SourceAddress:               ::
SourceAddressPrefix:         0
SourceAddressRange:          n/a
SourceAddressGranularity:    Packet
SourcePort:                  All
SourcePortRange:             n/a
SourcePortGranularity:       Rule
DestAddress:                 ::
DestAddressPrefix:           0
DestAddressRange:            n/a
DestAddressGranularity:      Packet
DestPort:                    All
DestPortRange:               n/a
DestPortGranularity:         Rule
OrigRmtConnPort:             n/a
RmtIDPayload:                n/a
RmtUdpEncapPort:             n/a
**********************************************************************
FilterName:                  DenyAllRule_Generated__________Outbnd_v6
FilterNameExtension:         n/a
GroupName:                   n/a
LocalStartActionName:        n/a
VpnActionName:               n/a
TunnelID:                    0x00
Type:                        Generic
State:                       Active
Action:                      Deny
Scope:                       Both
Direction:                   Outbound
OnDemand:                    n/a
SecurityClass:               0
Logging:                     All
Protocol:                    All
ICMPType:                    n/a
ICMPCode:                    n/a
OSPFType:                    n/a
TCPQualifier:                n/a
ProtocolGranularity:         Rule
SourceAddress:               ::
SourceAddressPrefix:         0
SourceAddressRange:          n/a
SourceAddressGranularity:    Packet
SourcePort:                  All
SourcePortRange:             n/a
SourcePortGranularity:       Rule
DestAddress:                 ::
```

```
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
*********************************************************************

18 entries selected
```

In the output listed in Example 3-20, filter `All_Traffic_Silver~7` is listed twice. The rule has been expanded into an inbound rule and an outbound rule. A quick glance down the list of parameters provides information about the VPN action, source, and destination IP addresses and source and destination ports.

The **ipsec** display option does not provide complete details on the VPN's policies. More complete information can be displayed using the following command:

```
pasearch -p TCPIPA -s DynamicVpn
```

Another format of the **ipsec** command can be with -t for traffic test. This command returns all the rules in the current filter table that match the given traffic type. For example:

```
ipsec -p TCPIPA -t 10.10.1.230 10.10.1.241 tcp 21 0 in 0
```

Running this command from SC31 (IP: 10.10.1.241) results in the output of all matching rules in the current filter table of SC31 that match FTP traffic, as shown in Example 3-21.

Example 3-21 The output of ipsec -t command mentioned above on SC31 z/OS image

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Tue Aug  8 10:38:57 2006
Primary:  IP Traffic Test Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    n/a               TotAvail: 2
TestData: 10.10.1.230  10.10.1.241  tcp 0 21 in 0

FilterName:                   All_Traffic_Silver~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                10.10.1.230
SourceAddressPrefix:          n/a
```

```
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  10.10.1.241
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
FilterName:                   DenyAllRule_Generated__________Inbnd
FilterNameExtension:          n/a
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************
```

In the output shown in Example 3-21, you can see that the first matching rule is a rule with action type IPSec_Silver (which means the tunnel will be secured), the Tunnel ID is Y0 (which means a dynamic tunnel will be created), and that the traffic direction is Inbound.

Displaying IKED and dynamic tunnels

In order to check that communication is working under IP security, you should try to use **ftp** or **ping** commands. If you have more than one TCP/IP stack in your z/OS image, you should run **ftp** in the following format:

```
ftp -p TCPIPA 10.10.1.230
```

In this example, we run the command from SC31, using stack TCPIPA. If the **ftp** command completed successfully, use the following command to display the IKED tunnel:

```
ipsec -p TCPIPA -k display
```

The output from this command can be seen in Example 3-22.

Example 3-22 ipsec active IKED tunnels display in SC31, after ftp to SC30

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Tue Aug  8 11:55:12 2006
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a

TunnelID:                     K1
KeyExchangeRuleName:          All_Traffic_Silver~5
KeyExchangeActionName:        All_Traffic_Silver
LocalEndPoint:                10.10.1.241
LocalIDType:                  IPV4
LocalID:                      10.10.1.241
RemoteEndPoint:               10.10.1.230
RemoteIDType:                 IPV4
RemoteID:                     10.10.1.230
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Sha
EncryptionAlgorithm:          DES
DiffieHellmanGroup            1
AuthenticationMethod:         PresharedKey
InitiatorCookie:              0X5CEB1217020C68E2
ResponderCookie:              0XF440FF3EA5136367
Lifesize:                     OK
CurrentByteCount:             312b
Lifetime:                     480m
LifetimeRefresh:              2006/08/08 18:37:04
LifetimeExpires:              2006/08/08 19:54:33
Role:                         Initiator
AssociatedDynamicTunnels:     1
NATTSupportLevel:             None
NATInFrntLclScEndPnt:         No
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     No
RmtNAPTDetected:              No
RmtUdpEncapPort:              n/a
*********************************************************************

1 entries selected
```

Note: **`ipsec`** command has three main display functions, which can be used to check the phases of IKED security association:

- To see the tunnels created in phase one of IKED security association, use **`ipsec -k display`** to see the IKED tunnels.
- To see the tunnels created in phase two of IKED security association, use **`ipsec -y display`** to see dynamic tunnels or **`ipsec -m display`** to see manual tunnels.

For more information about **`ipsec`** command, see *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781

Use the following command to display the active tunnel:

```
ipsec -p TCPIPA -y display
```

The output from this command can be seen in Example 3-23.

Example 3-23 ipsec active dynamic tunnels display in SC31, after ftp to SC30

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Tue Aug  8 11:45:49 2006
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y3
ParentIKETunnelID:            K1
VpnActionName:                IPSec__Silver
LocalDynVpnRule:              n/a
State:                        Active
HowToEncap:                   Transport
LocalEndPoint:                10.10.1.241
RemoteEndPoint:               10.10.1.230
LocalAddressBase:             10.10.1.241
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
RemoteAddressBase:            10.10.1.230
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              2206851923
 AuthOutboundSpi:             3347180942
HowToEncrypt:                 DES
 EncryptInboundSpi:           2206851923
 EncryptOutboundSpi:          3347180942
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              49
OutboundBytes:                1750
InboundPackets:               44
InboundBytes:                 8164
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2006/08/08 14:31:37
LifetimeExpires:              2006/08/08 15:09:29
CurrentTime:                  2006/08/08 11:45:49
VPNLifeExpires:               2006/08/09 11:09:29
NAT Traversal Topology:
```

```
  UdpEncapMode:                No
  LclNATDetected:              No
  RmtNATDetected:              No
  RmtNAPTDetected:             No
  RmtIsGw:                     n/a
  RmtIsZOS:                    n/a
  zOSCanInitP2SA:              n/a
  RmtUdpEncapPort:             n/a
SrcNATOARcvd:                n/a
  DstNATOARcvd:                n/a
*********************************************************************

1 entries selected
```

3.6 Implementing IPSec between z/OS and Windows

Figure 3-34 shows the setup we implemented in this scenario.

Figure 3-34 VPN between z/OS and Windows

The steps to configure a VPN between z/OS image and Windows XP are:

1. Set up the IKE daemon.
2. Set up the certificates.
3. Set up a z/OS IPSec policy, using IBM Configuration Assistant for z/OS Communication Server.
4. Set up a Windows IPSec policy, using MMC.
5. Verify that things are working.

3.6.1 Setting up the IKE daemon

In terms of procedure, user ID, and other configuration choices, the IKE daemon was set up using the same principles as outlined in 3.3.1, “Set up the Internet Key Exchange daemon” on page 78. The _CEE_ENVFILE environment variable was used to set up a STDENV DD card for controlling the environment variables. MVS data sets were used for all files. A sample of the cataloged procedure can be seen in Example 3-24.

Example 3-24 IKE daemon cataloged procedure

```
//IKED     EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD DUMMY
```

```
//STDENV   DD DSN=TCPIP.IKED.ENV(IKED),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

There is no need to specify a resolver configuration file here because, as is the case with the Policy Agent, one server should be used for all stacks within the image.

A sample config is supplied in the z/OS Communications Server installation file /usr/lpp/tcpip/samples/IBM/EZAIKCFG. The config we used is shown in Example 3-25.

Example 3-25 IKE daemon config file

```
IkeConfig
{
 IkeSyslogLevel    127
 PagentSyslogLevel 127
 Keyring           IKED/IKED_keyring
 KeyRetries        10
 KeyWait           30
 DataRetries       10
 DataWait          15
 Echo              no
 PagentWait        0
}
```

- IkeSyslogLevel and PagentSyslogLevel: These levels were set to 127 during testing to obtain helpful messages. After successful configuration, a much lower value should be used to avoid over-filling of the syslogd file.
- KeyRing: This is the RACF key ring name used to hold the certificate authority certificate required for ISAKMP/IKE authentication. This key ring was created using the user ID for the IKED started task, IKED. If the key ring had been created under any other user ID, then this user ID would need to be prepended to the key ring name using the syntax USERID/keyring. For example, in the IKE configuration file we could have specified KeyRing IKED/IKED_keyring to complete the exact same effect. Note, however, that accessing a key ring that is owned by another user ID requires UPDATE access to IRR.DIGTCERT.LISTRING.

3.6.2 Setting up the certificates

The next step is to establish a certificate environment for the key exchange. Both the z/OS host and the Windows XP host must have valid certificates configured in order for the IKE exchange to successfully establish a security association.

Because the key exchange mechanism of IKE involves a direct request of the certificate authority required, self-signed certificates were not an option. Instead, a certificate authority (CERTAUTH) certificate was created using RACF, and then this certificate was used to sign a personal certificate. Then a key ring was created and both certificates were connected. The personal certificate was connected as the default. The JCL we used is shown in Example 3-26.

Example 3-26 JCL for defining our key ring and digital certificates

```
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Create the top-level self-signed certificate for the certificate
//*   authority (CA in this case is ourselves)
//*
//SYSTSPRT DD   SYSOUT=*
```

```
//SYSTSIN  DD   *
    SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)

    RACDCERT ID(IKED) addring(IKED_keyring)

RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('SC31 RACF CA') O('IBM') -
    OU('ITSO') C('us')) WITHLABEL('SC31 RACF CA')

RACDCERT ID(IKED) GENCERT SUBJECTSDN(CN('WINXP Client') O('IBM') -
         OU('ITSO') C('us')) WITHLABEL('WINXP Client') -
         KEYUSAGE(HANDSHAKE) ALTNAME(IP(9.12.4.223))

RACDCERT ID(IKED) GENCERT SUBJECTSDN(CN('SC31 Server') O('IBM') -
         OU('ITSO') C('us')) WITHLABEL('SC31 Server') -
         SIGNWITH(CERTAUTH LABEL('SC31 RACF CA')) -
         KEYUSAGE(HANDSHAKE) ALTNAME(IP(10.10.1.241))

   SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

RACDCERT ID(IKED) CONNECT(ID(IKED) LABEL('XP Certificate') -
RING(IKED_keyring) USAGE(personal))
RACDCERT ID(IKED) CONNECT(CERTAUTH LABEL('SC31 RACF CA') -
         RING(IKED_keyring) USAGE(CERTAUTH))

RACDCERT ID(IKED) CONNECT(ID(IKED) LABEL('SC31 Server') -
   RING(IKED_keyring) USAGE(PERSONAL))
/*
```

In order to keep the configuration scenario simple, the same certificate authority and personal certificate were used on the XP workstation. To do this, the certificates have to be exported from the RACF database into MVS data sets using the commands shown in Example 3-27.

Example 3-27 Export job JCL

```
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Create the top-level self-signed certificate for the certificate
//*   authority (CA in this case is ourselves)
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
RACDCERT CERTAUTH EXPORT(LABEL('SC31 RACF CA')) -
   DSN('CS01.CERT.CACERT') -
   FORMAT(PKCS12DER) -
   PASSWORD('security')
RACDCERT ID(IKED) EXPORT(LABEL('WINXP Client')) -
   DSN('CS01.CERT.XPCERT') -
   FORMAT(PKCS12DER) -
   PASSWORD('security')
/*
```

Note that the certificate authority (signing) certificate does not require a private key. However, the personal certificate should include the private key, and hence the PKCS12DER format, along with a password, was used.

3.6.3 Setting up the z/OS IPSec policy

We used the IBM Configuration Assistant for z/OS Communication Server to create our IPSec policy file for SC31 z/OS image.

We defined two connectivity rules for the TCP/IP stack of SC31:

- Basic Services: A rule that permits Resolver, DNS, IKE, and OMPROUTE-IP_V4 traffic between all IPv4 addresses to all IPv4 addresses.
- All_Traffic_DES: A rule that creates a dynamic IPSEC tunnel for all other traffic types between the z/OS image IP address (10.10.1.241) and the Windows XP IP address (9.12.4.223).

In addition, we created a specific security level called DES that is used for All_traffic_DES connectivity rule.

The following steps are needed in order to have the same configuration we created:

1. Create a security level for DES authentication.
2. Create a requirement map objects called All_Traffic_DES and Basic_Services.
3. On the TCP/IP stack of the z/OS image, create the two connectivity rules, which are described above, Basic_Services and All_Traffic_DES.
4. Verify that the IKE daemon settings are correct.
5. Update the IPSec policy of the z/OS image.

Creating a security level for DES

To create a security level for DES:

1. Select **IPSec** → **Work with Reusable Objects** → **Security Levels** from the Configuration Assistant Navigation Tree. In the List of all defined Security Level objects panel, click **Add**.
2. In the New Security Level: Names and Type panel, enter the name of the security level and description. Select **IPSec Dynamic Tunnel** and click **Next**, as shown in Figure 3-35.

Figure 3-35 New Security Level: Names and Type panel - Adding DES

3. In New Security Level: Cipher Selections panel, select **DES** as the encryption algorithm and **HMAC SHA** as the authentication algorithm, as shown in Figure 3-36. Click **Next**.

Figure 3-36 New Security Level: Cipher Selections - Selecting DES and HMAC SHA

4. In the New Security Level: Additional Settings panel, click **Advanced Settings**.
5. In Advanced Dynamic Tunnel Settings panel, select the **Additional Settings** tab and select **Diffie-Hellman Group 1** as the Perfect Forward Secrecy (PFS) level, as shown in Figure 3-37. Click **OK** and then click **Finish**.

Figure 3-37 Advanced Dynamic Tunnel Settings panel - Selecting Group 1

The List of all defined Security Level objects panel should now include the DES object, as shown in Figure 3-38.

Figure 3-38 Security Levels panel: Including DES object

Creating requirement map objects

To create requirement map objects:

1. Select **IPSec** → **Work with Reusable Objects** → **Requirement Maps** from the Configuration Assistant Navigation Tree and click **Add**.
2. In the Requirement Map panel, enter a name and description for the object (we entered All_Traffic_DES). In addition, select **DES** to be the IPSEC - Security Level of the All_other_traffic traffic descriptor. You can do it by selecting the list box of IPSec - Security Level, as shown in Figure 3-39. Click **OK**.

Figure 3-39 Requirement Map panel: Adding All_Traffic_DES requirement map object

3. In the Requirement Map Tip panel, click **Proceed**.
4. In the Requirement Maps panel, click **Add**.

5. In the Requirement Map panel:
   a. Enter a name and description for the object.
   b. Select the **All_other_traffic** traffic descriptor and click **Remove**.
   c. In the Objects traffic descriptor list, select **DNS** and then click **Add**.

      Repeat the same procedure for the following traffic descriptor objects: OMPROUTE-IP-4 and Resolver.
   d. Change the Security Level of all the objects to **Permit** by selecting the list box of IPSec - Security Level.

      The final configuration of this panel should like the configuration shown in Figure 3-40.
   e. Click **OK**.

Figure 3-40 Requirement Map panel: Adding Basic_Services object

The Requirement Maps panel should now include the new objects, as shown in Figure 3-41.

Figure 3-41 Requirement Maps panel: Including All_Traffic_DES and Basic_Services objects

Creating the connectivity rules: Basic_Services and All_Traffic_DES

To create the connectivity rules, Basic_Services and All_Traffic_DES:

1. Select **IPSec** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name*** from the navigation tree.
2. To add the Basic_Services rule, click **Add** from the TCP/IP Stack Settings panel, and then click **Next**.
3. In the Connectivity Rule: Network Topology panel, select the **This connectivity Rule will contain only Permit and Deny Security Levels** button and click **Next**.
4. In the Connectivity Rule: Data Endpoints panel, select **All IP V4 addresses** as the source data endpoint and **All IP V4 addresses** as the destination data endpoints. In addition, enter a name to the rule. In our configuration, we entered Basic_Services, as shown in Figure 3-41. Click **Next**.
5. In Connectivity Rule: Select Requirement Map panel, select **Basic_Services** from the list and click **Next**. Then click **Finish**.
6. To add the All_Traffic_DES rule, click **Add** from the TCP/IP Stack Settings panel and click **Next**.

7. In the Connectivity Rule: Network Topology panel, select the **This connectivity Rule will contain a Security Level using IPSec tunnels** button, and then select **Host to Host** button. Click **Next**.
8. In the Connectivity Rule: Data Endpoints panel, enter the z/OS image IP address as the local data endpoint (in our configuration: 10.10.1.241) and the Windows XP IP address as the remote data endpoint (in our configuration: 9.12.4.223). In addition, enter a name to the rule. In our configuration, we entered All_Traffic_DES, as shown in Figure 3-42. Click **Next**.

Figure 3-42 Connectivity Rule: Data Endpoints panel - Defining All_Traffic_DES

9. In the Connectivity Rule: Select Requirement Map panel, select **All_Traffic_DES** from the list and click **Next**.

10. In the Connectivity Rule: Remote Security Endpoint Information panel, enter the remote IKE identity. We chose to use IP address as the remote identity type and therefore entered the Windows XP IP address. Select **Shared key** as the authentication method for remote IKE peers. Select **ASCII** and enter your shared key (in our configuration, it is abcde), as shown in Figure 3-43.

    Click **Next** and then click **Finish**.

Figure 3-43 Connectivity Rule: Remote Security Endpoint Information panel

The TCP/IP Stack Settings panel should now include the new rules, as shown in Figure 3-44.

Figure 3-44 TCP/IP Stack Settings panel: With the All_Traffic_DES and Basic_Services rules

Note: There is an importance to the order of the rules in the connectivity rules list. Verify that the rules are in the right order and use the **Move UP** button if necessary to change the order of the rules.

The resultant policy file for SC31 is displayed in Example 3-28.

Example 3-28 SC31 policy file for TCPIPA for windows traffic

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC31
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Tue Aug 15 11:51:02 EDT 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo 1
{
  IpFilterAction               Permit
  IpFilterLogging              No
}
```

```
IpGenericFilterAction          IpSec~LogYes 2
{
  IpFilterAction               IpSec
  IpFilterLogging              Yes
}

KeyExchangeOffer               KEO~9 3
{
  HowToEncrypt                 DES
  HowToAuthMsgs                SHA1
  HowToAuthPeers               PresharedKey
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    DES~R4
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    DES~N5
{
  HowToEncap                   Tunnel
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpService                      Resolver
{
  Protocol                     TCP
  SourcePortRange              1024 65535
  DestinationPortRange         53
  Direction                    BiDirectional OutboundConnect
  Routing                      Local
}

IpService                      Resolver~1
{
  Protocol                     UDP
  SourcePortRange              1024 65535
  DestinationPortRange         53
  Direction                    BiDirectional
  Routing                      Local
}

IpService                      DNS
{
```

```
  Protocol                    UDP
  SourcePortRange             53
  DestinationPortRange        1024 65535
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     DNS~2
{
  Protocol                    UDP
  SourcePortRange             53
  DestinationPortRange        53
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     DNS~3
{
  Protocol                    TCP
  SourcePortRange             53
  DestinationPortRange        1024 65535
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     DNS~4
{
  Protocol                    TCP
  SourcePortRange             53
  DestinationPortRange        53
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     IKE 6
{
  Protocol                    UDP
  SourcePortRange             500
  DestinationPortRange        500
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     OMPROUTE-IP_V4
{
  Protocol                    OSPF
  Type                        Any
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     OMPROUTE-IP_V4~5
{
  Protocol                    IGMP
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     OMPROUTE-IP_V4~6
{
```

```
  Protocol                    UDP
  SourcePortRange             520
  DestinationPortRange        1024 65535
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     OMPROUTE-IP_V4~7
{
  Protocol                    UDP
  SourcePortRange             1024 65535
  DestinationPortRange        520
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     OMPROUTE-IP_V4~8
{
  Protocol                    UDP
  SourcePortRange             520
  DestinationPortRange        520
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     All_other_traffic 7
{
  Protocol                    All
  Direction                   BiDirectional
  Routing                     Local
}

IpDynVpnAction                DES 8
{
  Initiation                  Either
  VpnLife                     1440
  Pfs                         Group1
  IpDataOfferRef              DES~R
  IpDataOfferRef              DES~N
}
##
## Connectivity Rule Basic_Services combines the following items:
##   Local data endpoint       All4
##   Remote data endpoint      All4
##   Topology                  None (Permit/Deny only)
##   Requirement Map           Basic_Services
##     Resolver                  => Permit
##     DNS                       => Permit
##     IKE                       => Permit
##     OMPROUTE-IP_V4            => Permit

IpFilterRule                  Basic_Services~1
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                Resolver
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  Basic_Services~2
```

```
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 Resolver~1
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~3
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~4
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS~2
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~5
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS~3
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~6
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 DNS~4
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~7
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 IKE
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~8
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~9
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~5
```

```
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~10
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~6
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~11
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~7
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Basic_Services~12
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~8
  IpGenericFilterActionRef     Permit~LogNo
}
##
## Connectivity Rule All_Traffic_DES combines the following items:
##   Local data endpoint       All_Traffic_DES~ADR~1
##   Remote data endpoint      All_Traffic_DES~ADR~2
##   Topology                  HH
##   Requirement Map           All_Traffic_DES
##     All_other_traffic         => DES

IpAddr                         All_Traffic_DES~ADR~1 9
{
  Addr                         10.10.1.241
}

IpAddr                         All_Traffic_DES~ADR~2
{
  Addr                         9.12.4.223
}

LocalSecurityEndpoint          All_Traffic_DES~LSE~4 10
{
  Identity                     IpAddr 10.10.1.241
  LocationRef                  All_Traffic_DES~ADR~1
}

RemoteSecurityEndpoint         All_Traffic_DES~RSE~3 11
{
  Identity                     IpAddr 9.12.4.223
  LocationRef                  All_Traffic_DES~ADR~2
}

KeyExchangeRule                All_Traffic_DES~5 12
{
  LocalSecurityEndpointRef     All_Traffic_DES~LSE~4
  RemoteSecurityEndpointRef    All_Traffic_DES~RSE~3
```

```
  KeyExchangeActionRef         All_Traffic_DES
  SharedKey                    Ascii abcde
}

KeyExchangeAction              All_Traffic_DES 13
{
  HowToInitiate                Main
  HowToRespond                 Either
  KeyExchangeOfferRef          KEO~9
  AllowNat                     No
}

IpFilterRule                   All_Traffic_DES~6 14
{
  IpSourceAddrRef              All_Traffic_DES~ADR~1
  IpDestAddrRef                All_Traffic_DES~ADR~2
  IpServiceRef                 All_other_traffic
  IpGenericFilterActionRef     IpSec~LogYes
  IpDynVpnActionRef            DES
}

KeyExchangePolicy 15
{
  AllowNat                     No
  KeyExchangeRuleRef           All_Traffic_DES~5
}

IpFilterPolicy 16
{
  PreDecap                     OFF
  FilterLogging                ON
  IpFilterLogImplicit          Yes
  AllowOnDemand                Yes
  IpFilterRuleRef              Basic_Services~1
  IpFilterRuleRef              Basic_Services~2
  IpFilterRuleRef              Basic_Services~3
  IpFilterRuleRef              Basic_Services~4
  IpFilterRuleRef              Basic_Services~5
  IpFilterRuleRef              Basic_Services~6
  IpFilterRuleRef              Basic_Services~7
  IpFilterRuleRef              Basic_Services~8
  IpFilterRuleRef              Basic_Services~9
  IpFilterRuleRef              Basic_Services~10
  IpFilterRuleRef              Basic_Services~11
  IpFilterRuleRef              Basic_Services~12
  IpFilterRuleRef              All_Traffic_DES~6
}
```

Inherent in the rules discussed below is the default rule that always exists for the Policy Agent, which is to deny everything. If no matching rule is found, the packet will be denied by this default rule.

1 This policy is used to define an action of permitting a packet and logging the event. This policy exists so that it can be later used in a rule.

2 This policy is the same as **1** above, except that it requires that the packet be protected within an IPSec VPN tunnel.

3 A VPN requires a symmetric key for encrypting data that flows along the tunnel. This policy defines the security parameters associated with the exchange of symmetric keys. DES encryption is used, though 3DES would offer more encryption security. SHA1 was used for authentication. PresharedKey is specified because this is a dynamic tunnel using a shared key to authenticate.

4 This is similar to the policy in 3 above, but this policy controls security parameters with respect to the data travelling over the VPN itself. This proposal is referred to in the Configuration Assistant as DES, which we created earlier on this section.

Note that the RefreshLifetimeAccepted has been set to between 120 and 480. The Windows XP client was found to have a default of only 900 seconds (15 minutes) for its Session Key settings. The Configuration Assistant chose a range of 120 to 480. This policy was manually edited after exporting it from the Configuration Assistant. This is the only alteration made to the originally created file.

The encapsulation mode chosen is transport mode with ESP as the desired security method.

5 This is the second choice offer (with **4** being the first choice) for security parameters for the data over the VPN.

6 Because the default rule is to deny everything, there has to be a service definition that describes the IKE traffic that is necessary for dynamic tunnel negotiation. IKE uses UDP datagrams via port 500 for negotiating tunnel characteristics.

7 This service describes the traffic that will be directed through the VPN. The Configuration Assistant contains built-in definitions of most well-known traffic types based upon well-known port usage. This service definition encompasses all IP traffic.

8 This policy defines the action to be taken for dynamic VPN setup. Either endpoint can initiate the VPN.

9 These two IpAddr statements were automatically generated as “variables” containing the z/OS host IP address (10.10.1.241) and the XP workstation address (9.12.4.223).

10 The VPN requires a method of establishing the identity of an endpoint. There are four different types available from the Configuration Assistant: IP address, Fully Qualified Domain Name, e-mail address, or X.500 distinguished name. The IP address was used to identify the endpoint.

The LocationRef points to the IpAddr defined as `All_Traffic_DES~ADR~1`, the local host’s IP address.

11 The Identity of the RemoteSecurityEndpoint is the XP workstation’s IP address.

12 This rule encompasses the two endpoints from **10** and **11** and specifies that the key exchange action identified in **13** will be used (the Configuration Assistant created this forward reference).

13 This is the action pointed to by **12** above. Specifying Main for the HowToInitiate parameter means that better (encrypted) identity exchange occurs. The z/OS VPN can respond to either aggressive or main mode exchanges.

Because no NAT devices exist between the endpoints, NAT traversal was set to `No`.

14 describes the rules that control the rule specification exchanges. Rule 0~6 describes the rule to allow the VPN traffic.

15 This is the policy that ultimately controls the behavior of the IKE daemon. It references the KeyExchangeRule from **12**.

16 This is the policy that identifies the rules to be activated. Turning off PreDecap in most cases will improve throughput by avoiding having each packet go through filtering only after decapsulation rather than both before and after. Filterlogging is set on, but implicit filter logging (that is, logging of the default deny rules) is turned off.

AllowOnDemand means that if a packet matches the rule for the VPN filter and the tunnel does not already exist, z/OS IP Security Services will attempt to establish the tunnel immediately.

Verifying that the IKE daemon settings are correct

To verify that the IKE daemon settings are correct:

1. Select **IPSec** → **Work with z/OS Images** → **Image - *image_name*** from the navigation tree.
2. In the z/OS Image Settings panel, select the **IPSec: IKE daemon Settings** tab and verify that SAF key ring database value is IKED/IKED_keyring and that the list of supported certificate authorities includes SC31 RACF CA.

Updating the IPSec policy of the z/OS image

FTP the new policy to the z/OS image, as explained in section 3.4, “Configuring quick dynamic tunnels” on page 88.

Run the MODIFY PAGENT,REFRESH console command to update the Policy Agent and IKED with the new configuration.

Checkpoint

To summarize the situation at this point in the scenario, the Policy Agent (PAGENT) and IKE daemons should both be running. The GUI has been used to configure a set of policies that will direct the z/OS host to send all traffic through a dynamic IPSec VPN. A set of certificates has been created and IKE is pointing to the key ring that contains those certificates in the RACF database. The next step is to ensure that an equivalent setup has been handled on the Windows XP workstation.

3.6.4 Setting up a Windows IPSec policy

In this section, we describe how to set up the Microsoft® Management Console (MMC). The MMC will have two snap-ins added: Certificates and IP Security Management (see Figure 3-45 on page 163). The Certificates Snap-in is used to import the certificates that are created by z/OS RACF. The IP Security Management Snap-in is used to configure the VPN connection between the Windows XP client and the z/OS server.

1. Click **Start** → **Run** from the Windows XP taskbar.
2. Enter `mmc` in the Open field. Click **OK** to start the Microsoft Management Console.
3. In the Console 1 window, click **Console** from the menu bar. From the pull-down menu, click **Add/Remove Snap-in**.
4. In the Add/Remove Snap-in window, click **Add**.
5. In the Add Standalone Snap-in window, select **Certificates** and click **Add**.
6. In the Certificates Snap-in window, select **Computer account** and click **Next**.
7. Select **Local computer** and click **Finish**.

8. In the Add Standalone Snap-in window, select **IP Security Policy Management** and click **Add**.
9. In the Select Computer window, select **Local computer** and click **Finish**.
10. In the Add Standalone Snap-in window, click **Close**.
11. In the Add/Remove Snap-in window, verify that two snap-ins have been added: Certificates (Local computer) and IP Security Policies on Local Machine. Click **OK**.

This process completes the required settings for the MMC.

Importing the z/OS certificates into Windows XP

In this section, we explain how to import two certificates that are created by z/OS RACF: The Trusted Root Certification Authority (CA) certificate and the client certificate.

Before installing the client certificate on the Windows XP client, Windows XP needs to entrust the Trusted Root CA (in this case, z/OS RACF acts as a Trusted Root CA to provide certificates to the clients). After Windows XP entrusts the CA, the client certificate can be installed on the Windows XP client. This client certificate is used for identity authentication in the IKE Phase 1 negotiation.

z/OS RACF creates an individual client certificate for each client, because the client certificate includes the client IP address information. This IP address information is used to verify the required authority on each client to connect to z/OS.

Figure 3-45 Microsoft Management Console (MMC): Certificates

FTP the certificates from z/OS image, as shown in Example 3-29.

Example 3-29 FTP the certificates to the Windows XP client

```
Connected to wtsc31.itso.ibm.com.
220-FTPMVS1 IBM FTP CS V1R8 at wtsc31.itso.ibm.com, 15:59:17 on 2006-08-16.
220 Connection will close if idle for more than 5 minutes.
User (wtsc31.itso.ibm.com:(none)): cs01
331 Send password please.
Password:
230 CS01 is logged on.  Working directory is "CS01.".
ftp> bin
200 Representation type is Image
ftp> get 'cs01.cert.xpcert' c:\xpclient.p12
200 Port request OK.
125 Sending data set CS01.CERT.XPCERT
250 Transfer completed successfully.
ftp: 1716 bytes received in 0.00Seconds 1716000.00Kbytes/sec.
ftp> get 'cs01.cert.cacert' c:\racfca.p12
200 Port request OK.
125 Sending data set CS01.CERT.CACERT
250 Transfer completed successfully.
ftp: 662 bytes received in 0.01Seconds
66.20Kbytes/sec.
```

Note: You must change the representation type of the FTP to image, before running the **`get`** commands.

To import the z/OS certificates into Windows XP, perform the following steps:

1. In the MMC window shown in Figure 3-46, click the plus sign (**+**) next to Certificates (Local Computer) to show the list of available tasks.
2. Right-click **Trusted Root Certification Authorities** and choose **All Tasks** from the pull-down menu. Choose **Import** from the next pull-down menu and click it, as shown in Figure 3-46.

Figure 3-46 Trusted Root Certification Authorities

3. In the Certificate Import Wizard window, click **Next**.

4. In the Certificate Import Wizard - File to Import window, click **Browse** and specify the Trusted Root CA file name (in this example, we choose C:\racfca.p12 for the Trusted Root CA file, as shown in Figure 3-47). Click **Next**.

Figure 3-47 Certificate Import Wizard

5. Work through the Certificate Import Wizard indicating the following:
   a. Enter the password you gave to the CA certificate. We entered security as the password.
   b. Do *not* select “Mark the private key as exportable.”, as shown in Figure 3-48.

Figure 3-48 Certificate Import Wizard: Entering the password of the certificate

c. Select **Place all certificates in the following store**.

d. Indicate that “Trusted Root Certification Authorities” is shown in the certificate store column.

e. Click **Finish** in the Completing the Certificate Import Wizard window.

You will receive a message that the import was successful.

6. In the MMC window, click the plus (**+**) sign next to Trusted Root Certification Authorities, and then click **Certificates**. Scroll down and verify that your Trusted Root CA is installed in the list. In this example, SC31 RACF CA is installed as a Trusted Root CA, as shown in Figure 3-49.

Figure 3-49 The Certificates list with the new SC31 RACF CA certificate

7. Right-click **Personal** and choose **All Tasks** from the pull-down menu. Choose **Import** from the next pull-down menu and select it, as shown in Figure 3-50.

Figure 3-50 Personal Certification Authorities

8. In the Certificate Import Wizard, click **Next**.
9. In the Certificate Import Wizard - File to Import window, click **Browse** and specify the client certificate file name (in this example, we choose C:\xpclient.p12 for the client certificate file shown in Figure 3-51). Click **Next**.

Figure 3-51 Importing the XP client certificate

10. Continue to work through the Certificate Import Wizard indicating the following:
    a. Enter the password you gave to the certificate. We entered security as the password.
    b. Do *not* select “Mark the private key as exportable.”, as shown in Figure 3-52.

Figure 3-52 Certificate Import Wizard: Entering the password of the certificate

    c. Make sure “Place all certificates in the following store” is selected and `Personal` is shown in the certificate store column.
    d. Click **Finish** in the Completing the Certificate Import Wizard window.

    You will receive a message that the import was successful.

11. In the MMC window, click the plus (**+**) sign next to Personal, and then click **Certificates**. Scroll down and verify that your client certificate is installed in the list. In this example, WinXP Client is installed as a client certificate, as shown in Figure 3-53.

Figure 3-53 Personal certificates list with WINXP Client certificate

Creating the IP security policy

In the following steps, you create the IP Security policy on your Windows XP workstation for the VPN connection between z/OS and the Windows XP client.

1. In the MMC - IP Security Policies on Local Computer window, right-click **IP Security Policies on Local Computer**. In the pull-down menu, click **Create IP Security Policy**, as shown in Figure 3-54.

Figure 3-54 MMC: IP Security Policies on Local Computer

2. In the IP Security Policy Wizard - Welcome to the IP Security Policy Wizard, click **Next**.
3. In the IP Security Policy Wizard - IP Security Policy Name window, type the name for the z/OS VPN connection. In this example, we typed zOSVPN for the VPN connection name. Type the description, if required. Click **Next**.
4. In the IP Security Policy Wizard - Requests for Secure Communication window, clear the “Activate the default response rule” check box. Click **Next**.
5. In the IP Security Policy Wizard - Completing the IP Security Policy Wizard, make sure that the **Edit properties** check box is selected. Click **Finish**.

6. In the IP Policy Properties window (in this example, the zOSVPN Properties window), select the **Use Add Wizard** check box, as shown in Figure 3-55. Click **Add**.

Figure 3-55 IP policy properties window

7. In the IP Security Wizard - Welcome to the IP Security Policy Wizard window, click **Next**.
8. In the Security Rule Wizard - Tunnel Endpoint window, select **This rule does not specify a tunnel** and click **Next**. This selection means that the z/OS image is the endpoint of the VPN tunnel with Windows XP, and the VPN tunnel is defined as transport mode.
9. In the Security Rule Wizard - Network Type window, select **All network connections**. Click **Next**. In this example, z/OS image and Windows XP are connected with the Ethernet LAN.

> **Note:** If you want to limit the remote access connection, select **Remote Access**.

10. In the IP Security Policy Wizard - Authentication Method window, select **Use this string to protect the key exchange (preshared key)**, enter your shared key, as shown in Figure 3-56. In this example, our shared key is abcde. Click **Next**.

> **Note:** We recommend that you do *not* use a shared key authentication method in a production environment. In a production environment, you should configure RSA signature as the authentication method of the IKE peers. For more information, see *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.

Figure 3-56 Authentication Method panel: Using a shared key

11. In the Security Rule Wizard - IP Filter List window, click the circle for **All IP Traffic**, as shown in Figure 3-57. Click **Edit**.

Figure 3-57 Security Rule Wizard: IP filter list

Attention: Notice that this IP filter works as a trigger event to establish the VPN tunnel. In this example, we choose All IP Traffic for the protocol and 10.10.1.241 for the Destination IP address. This means that if any IP datagram is about to issue from the Windows XP to 9.12.4.223, this IP filter detects the event and pulls a trigger to create a VPN tunnel between the Windows XP and 10.10.1.241.

12. In the IP filter List window, click **Edit**, as shown in Figure 3-58.

Figure 3-58 IP filter list

13. In the Filter Properties window (shown in Figure 3-59), select **A specific IP Address** in the Destination address column. Type the IP address of the z/OS image. (This IP address also means the VPN endpoint.) Select the **Mirrored. Also match packets with the exact opposite source and destination addresses** check box. In this example, we typed 10.10.1.241 for the Destination IP address. Click **OK**.

Figure 3-59 Filter properties

14. In the IP Filter List window, click **OK**.
15. In the Security Rule Wizard - IP Filter list window, click **Next**.
16. In the Security Rule Wizard - Filter Action window, click the circle for **Require Security**, as shown in Figure 3-60. Click **Edit**.

Figure 3-60 Security Rule Wizard: Filter Action

17. In the Require Security Properties window, choose **Negotiate security**; **Accept unsecured communication, but always respond using IPSec**; and **Session key Perfect Forward Security**.

    Use of Session key Perfect Forward Security is optional. In this example, we have to select it because the matching sample configuration in z/OS specifies to use the Session key Perfect Forward Security. Choose the topmost security method and click **Edit**, as shown in Figure 3-61.

Figure 3-61 Require Security Properties

18. In the Modify Security Method window, choose **Custom** (for expert users). Click **Settings**, as shown in Figure 3-62.

Figure 3-62 Modify security method

19. In the Custom Security Method Settings window, make sure that the “Data and address integrity without encryption (AH)” check box is not selected.

    Select **Data integrity and encryption (ESP)**, and select **SHA1** for integrity algorithm. Select **3DES** for encryption algorithm. Clear the “Generate a new key every Kbytes” check

box. Select the **Generate a new key every seconds** check box and type 7200 in the seconds column, as shown in Figure 3-63. Click **OK**.

Figure 3-63 Custom Security Method Settings

20. In the Modify Security Method window, click **OK**.
21. In the Require Security Properties window, click **OK**.
22. In the Security Rule Wizard - Filter Action window, click **Next**.
23. In the Security Rule Wizard - Completing the New Rule Wizard window, clear the “Edit properties” check box and click **Finish**.

24. In the zOSVPN Properties window, make sure that the **All IP Traffic** check box is checked, as shown in Figure 3-64. Click **OK**.

Figure 3-64 zOSVPN properties

25. Verify that the “IP Security Policies on Local Computer” now includes the zOSVPN policy.
26. Right-click the zOSVPN policy and select **Assign** from the menu.
27. Verify that the Policy Assigned status has changed to `Yes`, as shown in Figure 3-65.

Figure 3-65 The IP Security Policies on Local Computer panel: With zOSVPN policy assigned

3.6.5 Verifying that things are working

Use the following steps in order to verify that things are working:

1. Check that the Policy Agent has read the new policy.
2. Check that you succeed in running the **`ping`** command from the Windows XP to the z/OS image.

3. Verify that phase 1 of the security association has started.
4. Verify that phase 2 of the security association has started.
5. If you have problems, check the syslogd messages.

Checking that the Policy Agent has read the new policy

The first thing you need to check is that the Policy Agent has read in the current policies.

After you FTP the configuration file to the z/OS image, the following command is used to refresh the PAGENT (where pagent is the started task name for the Policy Agent):

```
MODIFY PAGENT,REFRESH
```

If no changes have been made since the last time the polices were read, then a message such as follows should be seen in response:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : NONE
```

If changes have been made, then instead, a message such as the following should be seen:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

Checking that ping command is working

To verify that IP security is working, try the following **ping** command from the Windows XP to the z/OS image.

```
ping 10.10.1.241
```

The output you get should be something like the output shown in Example 3-30.

Example 3-30 ping command

```
C:\Documents and Settings\RESIDENT>ping 10.10.1.241

Pinging 10.10.1.241 with 32 bytes of data:

Negotiating IP Security.
Reply from 10.10.1.241: bytes=32 time<1ms TTL=63
Reply from 10.10.1.241: bytes=32 time<1ms TTL=63
Reply from 10.10.1.241: bytes=32 time<1ms TTL=63

Ping statistics for 10.10.1.241:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average =
0ms
```

Verifying that security association phase 1 has started

To verify that an IKED tunnel has been created, run the **ipsec -k** command.

```
ipsec -p tcpipA -k display
```

The output you get should be something like the output shown in Example 3-31.

Example 3-31 ipsec -k command after ping command from XP to z/OS

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Tue Aug 15 10:55:53 2006
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a
```

```
TunnelID:                    K1
KeyExchangeRuleName:         0~5
KeyExchangeActionName:       0
LocalEndPoint:               10.10.1.241
LocalIDType:                 IPV4
LocalID:                     10.10.1.241
RemoteEndPoint:              9.12.4.223
RemoteIDType:                IPV4
RemoteID:                    9.12.4.223
ExchangeMode:                Main
State:                       DONE
AuthenticationAlgorithm:     Hmac_Sha
EncryptionAlgorithm:         DES
DiffieHellmanGroup           1
AuthenticationMethod:        PresharedKey
InitiatorCookie:             0X9065110C1FE7D109
ResponderCookie:             0XB2D1C745A9A588C0
Lifesize:                    OK
CurrentByteCount:            1824b
Lifetime:                    480m
LifetimeRefresh:             2006/08/15 17:02:34
LifetimeExpires:             2006/08/15 18:26:17
Role:                        Responder
AssociatedDynamicTunnels:    1
NATTSupportLevel:            None
NATInFrntLclScEndPnt:        No
NATInFrntRmtScEndPnt:        No
zOSCanInitiateP1SA:          Yes
AllowNat:                    No
RmtNAPTDetected:             No
RmtUdpEncapPort:             n/a
*********************************************************************

1 entries selected
```

Verifying that security association phase 2 has started

To verify that a dynamic tunnel has been created, run the **ipsec -y** command.

```
ipsec -p tcpipA -y display
```

The output you get should be something like the output shown in Example 3-32.

Example 3-32 ipsec -y command after ping command from XP to z/OS

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Tue Aug 15 10:57:15 2006
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                    Y4
ParentIKETunnelID:           K1
VpnActionName:               DES
LocalDynVpnRule:             n/a
State:                       Active
HowToEncap:                  Transport
LocalEndPoint:               10.10.1.241
RemoteEndPoint:              9.12.4.223
LocalAddressBase:            10.10.1.241
LocalAddressPrefix:          n/a
LocalAddressRange:           n/a
RemoteAddressBase:           9.12.4.223
```

```
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              1328063198
 AuthOutboundSpi:             3664910140
HowToEncrypt:                 DES
 EncryptInboundSpi:           1328063198
 EncryptOutboundSpi:          3664910140
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              2
OutboundBytes:                80
InboundPackets:               2
InboundBytes:                 80
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2006/08/15 12:26:42
LifetimeExpires:              2006/08/15 12:55:23
CurrentTime:                  2006/08/15 10:57:15
VPNLifeExpires:               2006/08/16 10:55:23
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
  SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
************************************************************************

1 entries selected
```

Checking the syslogd for messages

With TRMD running, syslogd is the repository for all Policy Agent and IKE daemon messages.

Enter the UNIX System Services environment, by running the command:

```
tso omvs
```

From the UNIX System Services environment, browse the log file you defined for IKED in the IKED configuration file.

```
obrowse /tmp/iked-sc31.log
```

Look for messages that can help you to solve the problem. If the log is empty, verify that TRMD is running and that syslogd is running with the right configuration file.

Note: The IKED log file has an important role in problem determination. When implementing the IP security for the first time, make sure that:

- IKED is configured to write log messages.
- TRMD is running.
- syslogd is running with the updated configuration file.

3.7 AES cryptographic support for integrated IPSec/VPN

z/OS V1R8 Communications Server introduces support for the Advanced Encryption Standard (AES) algorithm for IP security. Communications Server supports AES with a 128-bit key length. The National Institute of Standards and Technology (NIST) has named AES as the replacement for DES as the standard encryption algorithm. The IETF IPSec Working Group intends for AES to eventually be adopted as the default IPSec Encapsulating Security Payload (ESP) Cipher; therefore, AES must be included in compliant IPSec implementations.

Note: If ICSF is not currently installed on your z/OS image, follow the steps for installation and initialization in *z/OS V1R8.0 Cryptographic Services ICSF System Programmer's Guide*, SA22-7520.

Attention: AES encryption software is subject to export restrictions and might not be available in your country.

3.7.1 Coexistence requirements

The Communications Server Security Level 3 and the z/OS Security Level 3 features are required. If you want AES encryption in hardware, then ICSF FMID HCR7730 is required. This level of ICSF is available as a Web deliverable called “Cryptographic Support for z/OS V1R6/R7 and z/OS.e V1R6/R7” at:

http://www.ibm.com/servers/eserver/zseries/zos/downloads/

3.7.2 Enabling AES encryption for IP security tunnels

Perform the following steps in order to enable AES on your z/OS image. For those steps, we assume that IP Security is already configured on your z/OS system, and therefore only small changes are needed. If IPSec is still not configured on your z/OS image, you should follow the steps described in 3.3, “How IPSec is implemented” on page 78 and the steps described in 3.5, “Implementing IPSec between two z/OS systems” on page 95, and then perform the following steps:

1. Install the appropriate Security Level Feature of ICSF. This step is required.
2. If you want AES encryption to protect key exchanges between two security endpoints, follow the following steps, using the IBM Configuration Assistant for z/OS Communication Server GUI:
   a. Create a new Security Level object for AES encryption.

      From **IPSec** → **Work with Reusable Objects** → **Security Levels** panel, click **Add**.

   b. In the New Security Level: Names and Type panel, enter a name for the Security Level and verify that **IPSec Dynamic Tunnel** option is selected, as shown in Figure 3-66. Click **Next**.

Figure 3-66 Creating a new Security Level object for AES

c. In the Cipher Selections panel, select the **AES 128-bit** option, as shown in Figure 3-67. Click **Next**.

Figure 3-67 Cipher Selections panel - indicating AES as the encryption algorithm

d. In the Additional Settings panel, click **Advanced Settings**.

e. In the Advanced Dynamic Tunnel Settings panel, select the **Additional Settings** tab.

In the Additional Settings tab, select **Diffie-Hellman Group 5** or **Diffie-Hellman Group 14**, as shown in Figure 3-68. Click **OK**.

Figure 3-68 Advanced Dynamic Tunnel Settings - Selecting Diffie-Hellman Group 5

f. In the Additional Settings panel, click **Finish**.

g. From **IPSec** → **Work with Reusable Objects** → **Requirement Maps** panel, change the requirement objects which are being used for IP security to work with AES security level instead of the current configured security level.

You can do it by selecting the requirement map object from the list, clicking **Modify** and changing the IPSec - Security Level of the traffic descriptor to be **AES**.

In our test environment, we changed All_Traffic_Silver object map; see how it was created in “Step 1: Adding requirement map objects” on page 97.

We changed its name to All_Traffic_AES and changed the security level to AES, as shown in Figure 3-69. Click **OK** and then click **Proceed**.

Figure 3-69 Requirement Map panel - changing the All_Traffic_Silver to All_Traffic_AES

h. In the Question panel, click **Proceed** and then click **OK**.

i. If you look at the Stacks Connectivity Rules panel, you will see that they have been changed and are now using the changed requirement map. You can also change the name of the rule by selecting the rule and clicking **Modify Basics**. Then change the connectivity Rule Name at the bottom and click **OK**. We changed the name to All_Traffic_AES, as shown in Figure 3-70.

Figure 3-70 TCP/IP Stack Settings Connectivity Rules panel - using now All_traffic_AES requirement

Note: It is also possible to configure IP security policy to work with AES manually:

- Specify AES on the HowToEncrypt parameter on the KeyExchangeOffer statement.
- Specify either Group5 or Group14 on the DHGroup parameter on the KeyExchangeOffer statement.

3. Restart Policy Agent (PAGENT).
4. Start ICSF.

3.7.3 Verifying that AES is used in IP security

In order to determine whether AES encryption is used to protect IKE key exchange messages between two security endpoints, perform the following steps:

1. Issue the **`ipsec -k display`** command. Use the parameter -p to indicate the TCP/IP stack that you want to examine.
2. Verify that the value for the EncryptionAlgorithm parameter is AES.

The output from this command can be seen in Example 3-33.

Example 3-33 ipsec IKED tunnels with AES encryption

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Thu Aug 10 13:32:44 2006
Primary:  IKE tunnel      Function: Display           Format:   Detail
Source:   IKED            Scope:    Current           TotAvail: n/a

TunnelID:                     K1
```

```
KeyExchangeRuleName:          all_traffic_aes~5
KeyExchangeActionName:        all_traffic_aes
LocalEndPoint:                10.10.1.230
LocalIDType:                  IPV4
LocalID:                      10.10.1.230
RemoteEndPoint:               10.10.1.241
RemoteIDType:                 IPV4
RemoteID:                     10.10.1.241
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Md5
EncryptionAlgorithm:          AES
DiffieHellmanGroup            5
AuthenticationMethod:         PresharedKey
InitiatorCookie:              0XA736A74B7E650921
ResponderCookie:              0XCB4C358AB256E719
Lifesize:                     OK
CurrentByteCount:             Ob
Lifetime:                     480m
LifetimeRefresh:              2006/08/10 17:27:08
LifetimeExpires:              2006/08/10 19:40:52
Role:                         Initiator
AssociatedDynamicTunnels:     0
NATTSupportLevel:             None
NATInFrntLclScEndPnt:         No
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     No
RmtNAPTDetected:              No
RmtUdpEncapPort:              n/a
*********************************************************************

1 entries selected
```

In order to determine whether AES encryption is used to provide data confidentiality between two data endpoints, perform the following steps:

1. Issue the **ipsec -y display** command for dynamic tunnels or the **ipsec -m display** for manual tunnels. Use the -p parameter to indicate the TCP/IP stack that you want to examine.
2. Verify that the value for HowToEncrypt parameter is AES.

The output from this command can be seen in Example 3-34.

Example 3-34 ipsec dynamic tunnels with AES encryption

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Thu Aug 10 13:38:14 2006
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y3
ParentIKETunnelID:            K1
VpnActionName:                aes
LocalDynVpnRule:              n/a
State:                        Active
HowToEncap:                   Transport
LocalEndPoint:                10.10.1.230
RemoteEndPoint:               10.10.1.241
LocalAddressBase:             10.10.1.230
```

```
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
RemoteAddressBase:            10.10.1.241
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Md5
 AuthInboundSpi:              4027059663
 AuthOutboundSpi:             693787026
HowToEncrypt:                 AES
 EncryptInboundSpi:           4027059663
 EncryptOutboundSpi:          693787026
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              0
OutboundBytes:                0
InboundPackets:               0
InboundBytes:                 0
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2006/08/10 14:45:32
LifetimeExpires:              2006/08/10 15:40:47
CurrentTime:                  2006/08/10 13:38:14
VPNLifeExpires:               2006/08/10 17:24:15
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
************************************************************************

1 entries selected
```

3.8 IPv6 support for integrated IPSec/VPN

z/OS V1R8.0 Communications Server provides the necessary function to support IP filtering, IPSec, and IKE for IPv6. Specifically, the following enhancements are provided:

- The Policy Agent is enhanced to configure IP filters, manual tunnels, and dynamic tunnels for IPv6.
- The TCP/IP profile is enhanced to allow configuration of default IP filters for IPv6 when the policy-based IP filters are not active.
- The z/OS Communications Server IKE daemon is enhanced to negotiate dynamic tunnels for IPv6.
- The z/OS Network Security Configuration Assistant GUI that existed in V1R7 is replaced with an enhanced GUI called IBM Configuration Assistant for z/OS Communication Server. It is enhanced to configure IPSec for IPv6 in the Policy Agent and the IKE daemon

and to configure for IPv6 in the Policy Agent and the IKE daemon and to configure IPv6 for AT-TLS.

- The **ipsec** command is enhanced to display and modify installed IP filter information, manual tunnel information, and dynamic tunnel information for IPv6.
- The **pasearch** command is enhanced to display whether IPSec is enabled for IPv6 and to also display IPv6 policy rules and actions.
- The Traffic Regulation Management daemon (TRMD) is enhanced to support logging of IP Security events for IPv6 such as IP filter permits and denies.

3.8.1 Restrictions

z/OS Communications Server does not support IPv6 for the following functions:

- Sysplex wide security associations (SWSA)
- NAT traversal

3.8.2 Implementing IP Security over IPv6

In this scenario, we show how to set up a VPN tunnel between two z/OS systems, over IPv6.

Figure 3-71 VPN traffic between two z/OS systems over IPv6

In order to use the IPv6 support for integrated IPSec/VPN, perform the following tasks:

1. Enable integrated IP Security for IPv4 (if you have not already done so); this is a prerequisite for enabling IP Security for IPv6.
2. Define IPSec policies for IPv6 within the Policy Agent configuration files.
3. Display IPSec IPv6 policies from the Policy Agent.
4. Configure TCP/IP profile for IP Security for IPv6.
5. Display whether IP Security for IPv6 is active in the TCP/IP stack.
6. Display the security class configured for each IPv6 interface.
7. Display IPv6 filter rules.
8. Display IPv6 tunnels.
9. Display filters that apply to traffic between two IPv6 addresses.
10. Retrieve the new SNMP MIB object ibmMvsIp6IpsecEnabled, which indicates whether IP Security for IPv6 is currently active in the TCP/IP stack, and the existing SNMP MIB object ibmMvsIfSecClass, which supports the security class for each interface.

Attention: You must enable IPSec for IPv4 in order to enable IPSec for IPv6.

3.8.3 Enabling IP Security for IPv4

Follow the steps for enabling the integrated IPSec for IPv4, which are described in section 3.3, “How IPSec is implemented” on page 78.

3.8.4 Defining IPSec policies for IPv6

Specify the new IPSec policies in the configuration files identified with the CommonIpSecConfig and IpSecConfig statements. You can use IBM Configuration Assistant for z/OS Communication Server to create the IPSec Policy Agent configuration files or you can also change the configuration files manually.

In our test environment, we created a policy configuration file for IPv6 that is similar to the configuration file that we created in 3.5, “Implementing IPSec between two z/OS systems” on page 95 for IPv4. Each stack has two connectivity rules: One for permitting OmprouteIPv6 and the other for using IPSec_Silver between the SC30 and SC31 for all_other_traffic. The differences are that the IP addresses are IPv6 addresses and the requirement map objects are using traffic descriptors of IPv6.

The Connectivity Rules panel of each stack should look similar to Figure 3-72.

Figure 3-72 The Connectivity Rules panel of IPv6 IP security policy

Note that you also have to change the dynamic tunnel local identity, as shown in Figure 3-73.

Figure 3-73 IPSec: Dynamic Tunnel Local Identity panel with changes for IPv6

Our policy configuration file for IPv6 in SC30 z/OS image is shown in Example 3-35.

Example 3-35 Policy configuration file for IP security over IPv6

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC30
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Fri Aug 11 08:57:12 EDT 2006
##
## Copyright = None
##

## NOTE -- Generated IpGenericFilterAction Permit~LogYes
IpGenericFilterAction          Permit~LogYes
{
  IpFilterAction               Permit
  IpFilterLogging              Yes
}

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}
```

```
IpGenericFilterAction          IpSec~LogNo
{
  IpFilterAction               IpSec
  IpFilterLogging              No
}

KeyExchangeOffer               KEO~13
{
  HowToEncrypt                 DES
  HowToAuthMsgs                SHA1
  HowToAuthPeers               PresharedKey
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~14
{
  HowToEncap                   Transport
  HowToEncrypt                 3DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~15
{
  HowToEncap                   Transport
  HowToEncrypt                 AES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpService                      MulticastListener
{
  Protocol                     Icmpv6
  Type                         130
  Code                         0
  Direction                    BiDirectional
  Routing                      Either
}
```

```
IpService                       MulticastListener~2
{
  Protocol                      Icmpv6
  Type                          131
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       MulticastListener~3
{
  Protocol                      Icmpv6
  Type                          132
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       NeighborDiscovery
{
  Protocol                      Icmpv6
  Type                          133
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       NeighborDiscovery~4
{
  Protocol                      Icmpv6
  Type                          134
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       NeighborDiscovery~5
{
  Protocol                      Icmpv6
  Type                          135
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       NeighborDiscovery~6
{
  Protocol                      Icmpv6
  Type                          136
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V6
{
  Protocol                      OSPF
  Type                          Any
  Direction                     BiDirectional
  Routing                       Either
```

```
}

IpService                       OMPROUTE-IP_V6~7
{
  Protocol                      UDP
  SourcePortRange               521
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V6~8
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          521
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V6~9
{
  Protocol                      UDP
  SourcePortRange               521
  DestinationPortRange          521
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V6~10
{
  Protocol                      ICMPv6
  Type                          130
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V6~11
{
  Protocol                      ICMPv6
  Type                          131
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V6~12
{
  Protocol                      ICMPv6
  Type                          132
  Code                          0
  Direction                     BiDirectional
  Routing                       Either
}

## NOTE -- Generated IpService IKE~Gen
IpService                       IKE~Gen
{
  Protocol                      UDP
```

```
  SourcePortRange              500
  DestinationPortRange         500
  Direction                    BiDirectional
  Routing                      Local
}

IpService                      All_other_traffic
{
  Protocol                     All
  Direction                    BiDirectional
  Routing                      Local
}

IpDynVpnAction                 IPSec__Silver
{
  Initiation                   Either
  VpnLife                      1440
  Pfs                          None
  IpDataOfferRef               IPSec__Silver~R
  IpDataOfferRef               IPSec__Silver~R~14
  IpDataOfferRef               IPSec__Silver~R~15
}

## NOTE -- Generated IpFilterRule Pro1~1
IpFilterRule                   Pro1~1
{
  IpSourceAddr                 All6
  IpDestAddr                   All6
  IpServiceRef                 MulticastListener
  IpGenericFilterActionRef     Permit~LogYes
}

## NOTE -- Generated IpFilterRule Pro1~2
IpFilterRule                   Pro1~2
{
  IpSourceAddr                 All6
  IpDestAddr                   All6
  IpServiceRef                 MulticastListener~2
  IpGenericFilterActionRef     Permit~LogYes
}

## NOTE -- Generated IpFilterRule Pro1~3
IpFilterRule                   Pro1~3
{
  IpSourceAddr                 All6
  IpDestAddr                   All6
  IpServiceRef                 MulticastListener~3
  IpGenericFilterActionRef     Permit~LogYes
}

## NOTE -- Generated IpFilterRule Pro1~4
IpFilterRule                   Pro1~4
{
  IpSourceAddr                 All6
  IpDestAddr                   All6
  IpServiceRef                 NeighborDiscovery
  IpGenericFilterActionRef     Permit~LogYes
}

## NOTE -- Generated IpFilterRule Pro1~5
```

```
IpFilterRule                    Pro1~5
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  NeighborDiscovery~4
  IpGenericFilterActionRef      Permit~LogYes
}

## NOTE -- Generated IpFilterRule Pro1~6
IpFilterRule                    Pro1~6
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  NeighborDiscovery~5
  IpGenericFilterActionRef      Permit~LogYes
}

## NOTE -- Generated IpFilterRule Pro1~7
IpFilterRule                    Pro1~7
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  NeighborDiscovery~6
  IpGenericFilterActionRef      Permit~LogYes
}
##
## Connectivity Rule OmprouteIPv6 combines the following items:
##   Local data endpoint        All6
##   Remote data endpoint       All6
##   Topology                   None (Permit/Deny only)
##   Requirement Map            OmproutIPv6
##     OMPROUTE-IP_V6             => Permit

IpFilterRule                    OmprouteIPv6~1
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    OmprouteIPv6~2
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6~7
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    OmprouteIPv6~3
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6~8
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    OmprouteIPv6~4
{
  IpSourceAddr                  All6
```

```
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6~9
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    OmprouteIPv6~5
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6~10
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    OmprouteIPv6~6
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6~11
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    OmprouteIPv6~7
{
  IpSourceAddr                  All6
  IpDestAddr                    All6
  IpServiceRef                  OMPROUTE-IP_V6~12
  IpGenericFilterActionRef      Permit~LogNo
}
##
## Connectivity Rule TRAFFIC_IPV6 combines the following items:
##   Local data endpoint        TRAFFIC_IPV6~ADR~1
##   Remote data endpoint       TRAFFIC_IPV6~ADR~2
##   Topology                   HH
##   Requirement Map            All_Traffic_Silver
##     All_other_traffic          => IPSec__Silver

IpAddr                          TRAFFIC_IPV6~ADR~1
{
  Addr                          AOA::2EC
}

IpAddr                          TRAFFIC_IPV6~ADR~2
{
  Addr                          AOA::2F6
}

LocalSecurityEndpoint           TRAFFIC_IPV6~LSE~4
{
  Identity                      IpAddr AOA::2EC
  LocationRef                   TRAFFIC_IPV6~ADR~1
}

RemoteSecurityEndpoint          TRAFFIC_IPV6~RSE~3
{
  Identity                      IpAddr AOA::2F6
  LocationRef                   TRAFFIC_IPV6~ADR~2
}

KeyExchangeRule                 TRAFFIC_IPV6~5
{
```

```
  LocalSecurityEndpointRef     TRAFFIC_IPV6~LSE~4
  RemoteSecurityEndpointRef    TRAFFIC_IPV6~RSE~3
  KeyExchangeActionRef         TRAFFIC_IPV6
  SharedKey                    Ebcdic ABCDE
}

KeyExchangeAction              TRAFFIC_IPV6
{
  HowToInitiate                Main
  HowToRespond                 Either
  KeyExchangeOfferRef          KEO~13
  AllowNat                     No
}

## NOTE -- Generated IpFilterRule TRAFFIC_IPV6~6
IpFilterRule                   TRAFFIC_IPV6~6
{
  IpSourceAddrRef              TRAFFIC_IPV6~ADR~1
  IpDestAddrRef                TRAFFIC_IPV6~ADR~2
  IpServiceRef                 IKE~Gen
  IpGenericFilterActionRef     Permit~LogYes
}

IpFilterRule                   TRAFFIC_IPV6~7
{
  IpSourceAddrRef              TRAFFIC_IPV6~ADR~1
  IpDestAddrRef                TRAFFIC_IPV6~ADR~2
  IpServiceRef                 All_other_traffic
  IpGenericFilterActionRef     IpSec~LogNo
  IpDynVpnActionRef            IPSec__Silver
}

KeyExchangePolicy
{
  AllowNat                     No
  KeyExchangeRuleRef           TRAFFIC_IPV6~5
}

IpFilterPolicy
{
  PreDecap                     OFF
  FilterLogging                ON
  IpFilterLogImplicit          Yes
  AllowOnDemand                Yes
  IpFilterRuleRef              Pro1~1
  IpFilterRuleRef              Pro1~2
  IpFilterRuleRef              Pro1~3
  IpFilterRuleRef              Pro1~4
  IpFilterRuleRef              Pro1~5
  IpFilterRuleRef              Pro1~6
  IpFilterRuleRef              Pro1~7
  IpFilterRuleRef              OmprouteIPv6~1
  IpFilterRuleRef              OmprouteIPv6~2
  IpFilterRuleRef              OmprouteIPv6~3
  IpFilterRuleRef              OmprouteIPv6~4
```

```
  IpFilterRuleRef              OmprouteIPv6~5
  IpFilterRuleRef              OmprouteIPv6~6
  IpFilterRuleRef              OmprouteIPv6~7
  IpFilterRuleRef              TRAFFIC_IPV6~6
  IpFilterRuleRef              TRAFFIC_IPV6~7
}
```

3.8.5 Displaying IPSec IPv6 policies from Policy Agent

Issue the **pasearch -v a** command for policy rules and actions.

Issue the **pasearch -c** command for policy object.

3.8.6 Configuring TCP/IP profile for IP Security for IPv6

Perform the following steps:

1. Specify the IPSECURITY and the SOURCEVIPA parameters on the IPCONFIG6 statement in the TCP/IP profile.
2. Optionally, assign security classes to network interfaces using the SECCLASS parameter on the INTERFACE statements in the TCP/IP profile.
3. Optionally, assign security classes to dynamic XCF interfaces using the SECCLASS parameter on the IPCONFIG6 DYNAMICXCF statement.
4. Define default IPv6 filter rules using the IPSEC6RULE statement within the IPSEC block in the TCP/IP profile.
5. Restart the TCP/IP stack to enable the new configuration.

For more information, refer to the following sections in *z/OS V1R8.0 Communications Server: IP Configuration Reference*, SC31-8776: TCP/IP profile, IPCONFIG6, IPSEC, and INTERFACE.

Our TCP/IP profile for TCPIPA stack in SC30 z/OS image is shown in Example 3-36 (only the statements that are relevant for IPv6 and IP security configuration are shown here).

Example 3-36 Statements which are relevant for enabling IPv6 in TCPIPA stack PROFILE

```
; Added IPSECURITY to IPCONFIG statement -IPv4
;
IPCONFIG DATAGRAMFWD SYSPLEXROUTING IPSECURITY SOURCEVIPA
DYNAMICXCF 10.20.10.100 255.255.255.0 8
;
; Added IPCONFIG6 statement
;
IPCONFIG6 DATAGRAMFWD IPSECURITY SOURCEVIPA
;
; Added IPSEC statement
;
IPSEC  LOGENABLE
   ;;  OSPF protocol used by Omproute
   IPSECRULE *  *  NOLOG  PROTOCOL OSPF
   ;;  IGMP protocol used by Omproute
   IPSECRULE *  *  NOLOG  PROTOCOL 2
   ;;  DNS queries to UDP port 53
  IPSECRULE *  *  NOLOG  PROTOCOL UDP      SRCPORT *  DESTPORT 53  SECCL
;;  Administrative access
   IPSECRULE *  9.1.1.1  LOG  PROTOCOL *
```

```
   ;;  IPV6 OSPF protocol used by Omproute
   IPSEC6RULE *  *  NOLOG  PROTOCOL OSPF
   ;;  IPV6 IGMP protocol used by Omproute
   IPSEC6RULE *  *  NOLOG  PROTOCOL 2
   ;;  IPV6 DNS queries to UDP port 53
  IPSEC6RULE *  *  NOLOG  PROTOCOL UDP     SRCPORT *  DESTPORT 53  SECCL
   ;;  IPV6 Administrative access
   IPSEC6RULE *  901::101 LOG  PROTOCOL *
ENDIPSEC

...
...
INTERFACE LNK620E0 DEFINE IPAQENET6 PORTNAME OSA20E0
IPADDR A0A::2EC
;Osa definitions... relates to sys1.vtamlst trle definitions
;major nodes : osa2080,osa20a0,osa20c0 and osa20e0
DEVICE OSA20E0  MPCIPA
LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 11
INTERFACE LNK620E0 DEFINE IPAQENET6 PORTNAME OSA20E0
IPADDR A0A::2EC
...
...
BEGINRoutes
; Direct Routes - Routes that are directly connected to my interfaces
;     Destination     Subnet Mask   First Hop Link Name     Packet Size
 ROUTE A0A::200/10                    =        LNK620E0      MTU 1492
...
...
ENDRoutes
...
...
PORT
...
500 UDP IKED
4500 UDP IKED
..
..
START OSA20e0
START LNK620E0
```

3.8.7 Displaying whether IP Security for IPv6 is active in the TCP/IP stack

Issue the Netstat CONFIG/-f command. Make sure that IpSecurity and SourceVipa fields, under IPv6 Configuration Table, are equal to Yes.

For more information about the Netstat command, see *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781.

3.8.8 Displaying the security class configured for each IPv6 interface

Issue the Netstat DEVLINKS/-d command. Make sure that the IntfStatus field for the interface that is used for IPv6 is equal to Ready.

For more information about the Netstat command, see *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781.

3.8.9 Displaying IPv6 filter rules

Issue the **ipsec -f display** command.

For more information about the **ipsec** command, see *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781.

3.8.10 Displaying IPv6 tunnels

Issue the **ipsec -m display, ipsec -k display** and **ipsec -y display** commands.

```
ipsec -p TCPIPA -k display
```

The output for the command above, which was issued from SC31 z/OS image, is shown in Example 3-37. We issued the **ipsec -k** command after we issued the **ping** command from SC31 to SC30.

Example 3-37 Output of ipsec -p TCPIPA -k display from SC31, when using IPSec over IPv6

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Thu Aug 10 16:32:24 2006
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a

TunnelID:                     K1
KeyExchangeRuleName:          all_traffic_SilverIPv6~5
KeyExchangeActionName:        all_traffic_SilverIPv6
LocalEndPoint:                a0a::2f6
LocalIDType:                  IPV6
LocalID:                      a0a::2f6
RemoteEndPoint:               a0a::2ec
RemoteIDType:                 IPV6
RemoteID:                     a0a::2ec
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Sha
EncryptionAlgorithm:          DES
DiffieHellmanGroup            1
AuthenticationMethod:         PresharedKey
InitiatorCookie:              0XB42DDDAF464C7D90
CurrentByteCount:             312b
Lifetime:                     480m
LifetimeRefresh:              2006/08/10 22:17:08
LifetimeExpires:              2006/08/11 00:29:41
Role:                         Initiator
AssociatedDynamicTunnels:     1
NATTSupportLevel:             n/a
NATInFrntLclScEndPnt:         n/a
NATInFrntRmtScEndPnt:         n/a
zOSCanInitiateP1SA:           n/a
AllowNat:                     n/a
RmtNAPTDetected:              n/a
RmtUdpEncapPort:              n/a
************************************************************************
1 entries selected
```

```
ipsec -p TCPIPA -y display
```

The output for the command above, which was issued from SC31 z/OS image, is shown in Example 3-38. We issued the **ipsec -y** command after we issued the **ping** command from SC31 to SC30.

Example 3-38 Output of ipsec -p TCPIPA -y display from SC31, when using IPSec over IPv6

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Thu Aug 10 16:32:29 2006
Primary:  Dynamic tunnel  Function: Display            Format:   Detail
Source:   Stack           Scope:    Current            TotAvail: 1

TunnelID:                     Y2
ParentIKETunnelID:            K1
VpnActionName:                IPSec__Silver
LocalDynVpnRule:              n/a
State:                        Active
HowToEncap:                   Transport
LocalEndPoint:                a0a::2f6
RemoteEndPoint:               a0a::2ec
LocalAddressBase:             a0a::2f6
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
RemoteAddressBase:            a0a::2ec
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              3582660052
 AuthOutboundSpi:             1088473399
HowToEncrypt:                 DES
 EncryptInboundSpi:           3582660052
 EncryptOutboundSpi:          1088473399
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              2
OutboundBytes:                528
InboundPackets:               2
InboundBytes:                 528
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2006/08/10 19:30:12
LifetimeExpires:              2006/08/10 20:29:41
CurrentTime:                  2006/08/10 16:32:29
VPNLifeExpires:               2006/08/11 16:29:41
NAT Traversal Topology:
  UdpEncapMode:               n/a
  LclNATDetected:             n/a
  RmtNATDetected:             n/a
  RmtNAPTDetected:            n/a
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
*************************************************************************

1 entries selected
```

3.8.11 Displaying filters that apply to traffic between two IPv6 addresses

Issue the **ipsec -t** command.

For example, we issued the following command from SC31:

```
ipsec -p TCPIPA -t A0A::2EC A0A::2F6 tcp 21 0 in 0
```

The command checks if FTP traffic is allowed from SC30 to SC31. It can be seen in the output that FTP traffic is allowed in that direction, under secure dynamic channels, using IPSec_Silver security level. The output is shown in Example 3-39.

Example 3-39 Output of ipsec -t from SC31, when using IPSec over IPv6

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Fri Aug 11 08:52:37 2006
Primary:  IP Traffic Test Function: Display            Format:   Detail
Source:   Stack Policy    Scope:    n/a                TotAvail: 3
TestData: a0a::2ec  a0a::2f6  tcp 0 21 in 0

FilterName:                   all_traffic_SilverIPv6~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y2
Type:                         Dynamic
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      None
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                a0a::2ec
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  a0a::2f6
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
***********************************************************************
FilterName:                   all_traffic_SilverIPv6~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
```

```
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      None
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                a0a::2ec
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  a0a::2f6
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
*************************************************************************
FilterName:                   DenyAllRule_Generated__________Inbnd_v6
FilterNameExtension:          n/a
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Inbound
OnDemand:                     n/a
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                ::
SourceAddressPrefix:          0
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
```

```
SourcePortGranularity:        Rule
DestAddress:                  ::
DestAddressPrefix:            0
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
************************************************************************

3 entries selected
```

3.8.12 Retrieving the new SNMP MIB objects

The new SNMP MIB object ibmMvsIp6IpsecEnabled indicates whether IP security for IPv6 is currently active in the TCP/IP stack, and the existing SNMP MIB object ibmMvsIfsecClass supports the security class for each interface.

The action you take depends on the management application:

- For the z/OS UNIX **snmp** command, no action is required.
- For the IBM NetView® SNMP command, use the most current copy of the sample MIBDESC.DATA file, which is shipped in SEZAINST(MIBDESC).

Other management applications might require different changes.

For more information, read the description of the MIB object in the IBM MVS TCP/IP Enterprise-specific MIB module shipped file /usr/lpp/tcpip/samples/mvstcpip.mi2.

3.9 Further information

Refer to the *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775, for additional information regarding IPSec configuration.

4

Network Address Translation support

This chapter discusses the support for an IP Security (IPSec) tunnel that traverses a Network Address Port Translation (NAPT) device, which was added in z/OS V1R8. We also provide a high-level overview of Network Address Translation (NAT), Network Address Port Translation (NAPT), and the previous nonsupport for an IPSec tunnel that traversed a NAT device performing port translation.

We discuss the solution to the problem and also demonstrate the new support in our test environment and show the corresponding configuration examples used to verify this.

Note: z/OS Communications Server provides NAT traversal support for only IPv4.

This chapter discusses the topics listed in Table 4-1.

Table 4-1 Topics discussed in this chapter

Section	Topics
4.1, “Network Address Translation” on page 206	Describes the basic functionality of NAT and Network Address Port Translation (NAPT).
4.2, “IPSec and NAT incompatibles” on page 208	Provides a short description of the incompatibilities between IPSec and NATT.
4.3, “Network address port translation traversal support for integrated IPSec/VPN” on page 208	Presents the solution, testing environment, configuration examples, and problem determination suggestions.

© Copyright IBM Corp. 2007. All rights reserved.

4.1 Network Address Translation

Network Address Translation (NAT) is a protocol defined in RFC 1631. In any network today one will find addresses that have been assigned to other organizations, private addresses, and globally unique addresses.

With the increase of Internet addresses worldwide, it is becoming more difficult for enterprises to assign globally unique IP addresses to each host. There simply are not enough unique IP addresses to ensure secure and accurate data transmission from site to site or company to company. As a result, NAT devices are fast becoming a way for large, multisite enterprises to avoid IP address conflicts. A NAT device acts as a router between two networks with the ability to translate IP addresses and ports. When connecting two separate networks, there will be a designated *inside* and *outside* network.

NAT itself provides a simple IP address conversion table between a private IP address and a public IP address. After the IP address conversion is done, IP checksum and TCP checksum are recalculated and updated.

NAT encompasses the following IP mapping techniques:

- One-to-one address translation (static or dynamic NAT)
- Network Address Port Translation (NAPT)

One-to-one NAT

An internal-external IP address mapping is maintained by the NAT device. IP addresses are translated, but ports are unchanged. The mapping can be static or dynamic. For a static mapping, there is a definition in the NAT that always translates IP address *x.x.x.x* to IP address *y.y.y.y*. An outbound packet is not needed to establish the mapping. For a dynamic mapping, the NAT has a pool of IP addresses that are assigned as needed, therefore IP address *x.x.x.x* might be mapped to IP address *y.y.y.y* one time, and to IP address *z.z.z.z* at another time. The mapping is established when an outbound packet is processed.

Figure 4-1 shows static NTAT.

Figure 4-1 Static NAT

Network Address Port Translation

The NAPT technique uses multiple internal IP addresses that are translated into a single public IP address. As part of this translation process, the TCP and UDP ports in the packets are translated. NAPT is sometimes referred to as *port address translation (PAT)* or *IP masquerade*. The mapping is typically dynamic and established in the NAPT when an outbound packet is processed. For example, the NAPT might create a mapping for internal address *x.x.x.x* port *y* to external address *a.a.a.a* port *b*, and a mapping for internal address *z.z.z.z* port *v* to external address *a.a.a.a* port *c*. When only one client behind a NAPT has

negotiated a security association, it is not always possible for the remote peer to detect whether the public address is being used for one-to-one address translation or for NAPT. When multiple clients have active security associations, the remote peer can detect when port translation is being performed. The terms *in front of* and *behind* are used to convey which IP address a NAT is translating. When a NAT is said to be in front of a client, it means that the client's address will be translated by the NAT. When a server is said to be behind a NAT, it means that server's address will be translated by the NAT.

In our NAPT configuration example shown in Figure 4-2, clients communicate with a TELNET server through a NAT router. All clients are communicating with only one address called the inside global address as displayed in the router's NAT table. This registered IP address is used by users who come from the public network to reach the z/OS TELNET server in the private network.

Figure 4-2 NAPT configuration example

Example 4-1 shows the results from the corresponding z/OS "Netstat Telnet" display command for the above client connections to port 23 on the z/OS TELNET server.

Example 4-1 z/OS Telnet display

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          11:37:33
Internal Telnet Server Status:
Conn     State    BytesIn    BytesOut   ApplName LuName
----     -----    -------    --------   -------- ------
00009902 Establsh 0000000032 0000000739          SC31TS02
  Foreign socket: ::ffff:192.168.1.251..3905
00009900 Establsh 0000000032 0000000739          SC31TS01
  Foreign socket: ::ffff:192.168.1.254..3145
```

4.2 IPSec and NAT incompatibles

This section discusses the incompatibilities between IPSec and NAT functions.

Background

There are inherent incompatibilities between IPSec and NAT functions. RFC 3715, IPSec-Network Address Translation (NAT) Compatibility Requirements, provides a description of the problems that arise when IPSec is used to protect traffic that traverses a NAT. One basic problem is that when IPSec security associations traverse a NAT, the NAT is unable to update IP addresses and checksums that are part of the encapsulated data (encrypted, authenticated, or both). RFCs 3947 and 3948 define mechanisms that enable specific uses of IPSec to traverse one or more NAT devices.

- RFC 3947, Negotiation of NAT-Traversal in the IKE, allows an Internet Key Exchange (IKE) daemon to detect when one or more NATs are being traversed.
- RFC 3948, UDP Encapsulation of IPSec ESP Packets, defines two IPSec encapsulation modes, UDP-Encapsulated-Tunnel mode and UDP-Encapsulated-Transport mode. These modes facilitate the traversal of IPSec traffic through a NAT by encapsulating ESP packets within a UDP packet.

UDP-Encapsulated-Tunnel mode and UDP-Encapsulated-Transport mode security associations are negotiated by IKE when NAT traversal is enabled and a NAT is detected. Manually configured security associations do not support UDP-Encapsulated-Tunnel mode or UDP-Encapsulated-Transport mode.

There are several reasons why Network Address Translation might be used. One reason is to economize on the use of public addresses within the internal network, using a public address only when data must be globally routed. A second reason is to hide internal IP addresses from network segments outside the internal IP address domain.

It should be noted that the NAT traversal support, as defined by the Internet Engineering Task Force (IETF), transmits internal IP addresses to its IPSec peer. These internal addresses are not exposed on the external network. However, the internal addresses are available for display at the remote security endpoint. If you are considering using this support, evaluate whether transmitting internal IP addresses to an IPSec peer is acceptable from a security policy perspective.

Attention: UDP encapsulation is *not* encapsulating a UDP packet. UDP encapsulation is inserting a UDP header between the IP header and the ESP header. The payload data can have a TCP, UDP, or other transport header.

4.3 Network address port translation traversal support for integrated IPSec/VPN

z/OS V1R8 Communications Server extends IP security support to protect traffic that traverses a NAPT device. A NAPT device translates multiple internal IP addresses to a single public address and translates the TCP or UDP port to make the connection unique. There are inherent incompatibilities between IPSec and NAT functions. Problems occur when IPSec is used to protect traffic that traverses a NAT or NAPT device. NAT traversal support as defined by the IETF in RFC 3947 (Negotiation of NAT-traversal in the IKE) and RFC 3948 (UDP encapsulation of IPSec ESP packets) defines mechanisms that allow specific uses of IPSec to traverse one or more NAT or NAPT devices. IP security provides NAT traversal support for

a defined group of configurations where an IPSec security association (SA) traverses a NAPT device.

Restrictions

In configurations where the remote security endpoint is behind a NAPT device, the following SA restrictions exist:

- The remote security endpoint must initiate the SA.
- The remote data endpoint must initiate the data.
- Only TCP, UDP, and Internet Control Message Protocol (ICMP) traffic is supported. ICMP traffic has limited support.
- In a sysplex-wide security association (SWSA) environment, the SA can be distributed only to targets at release level V1R8 or later that have IP security configured.
- In an SWSA environment, SWSA takeover functions are not available for the SA.

4.3.1 Enabling NAPT traversal support for IPSec

The objective was to successfully establish an IPSec dynamic tunnel from a Windows client, via a Cisco NAPT device to a TELNET server running on the z/OS mainframe. In order to test and verify the new support, we made the following configuration changes:

- IBM Configuration Assistant: Enabled NAT traversal support for TCPIPA
- Cisco router: Configured NAT and Port Address Translation (PAT) for port 23
- Windows XP: Installed and configured IPSec

Figure 4-3 represent our test environment.

Figure 4-3 NAPT traversal diagram

We now go through the configuration changes we made to our test environment to successfully establish the IPSec dynamic tunnel via the NAPT traversal device.

IBM Configuration Assistant changes

In this section, we walk you through the additional changes required on the Configuration Assistant graphical user interface (GUI) in order to enable the NAPT support.

Important: Before making these changes, refer to and complete the IPSec for Windows XP implementation in Chapter 3, “IP Security” on page 75.

1. From the main Configuration Assistant panel, we clicked **TCPIPA** → **IPSec: Stack Level Settings**, as shown in Figure 4-4.

Figure 4-4 TCP/IP Stack Settings

2. From the IPSec: Stack Level Settings panel, we selected **NAT Defaults** → **Allow** → **Apply**, as shown in Figure 4-5.

Figure 4-5 IPSec: Stack Level Settings

3. From the TCP/IP Stack Settings panel, we clicked rule → **10.10.1.24 to 192.168.1.253** → **Modify Basics**, as shown in Figure 4-6.

Figure 4-6 TCPIP Stack Settings

4. From the Connectivity Rule panel (Figure 4-7), we clicked **Additional Settings**.

Figure 4-7 Connectivity Rule

5. From the Additional Settings panel, we clicked **Advanced**, as shown in Figure 4-8.

Figure 4-8 Connectivity Rules: Additional Settings

6. From the Advanced Connectivity Rule Settings panel (Figure 4-9), we clicked **IPSec: Dynamic Tunnels: Key Exchange Settings**.

Figure 4-9 Advanced Connectivity Rule Settings

7. From the IPSec: Dynamic Tunnels: Key Exchange Settings for the Network Address Translation NAT Traversal field, we selected the **Allow** button, as shown in Figure 4-10.

Figure 4-10 IPSec: Dynamic Tunnels: Key Exchange Settings

8. We transferred the new IPSec rules to the PAGENT server on z/OS, as shown in Figure 4-11.

Figure 4-11 Configuration Files Installation

Cisco Router

For our NAPT device, we used a Cisco 2600 series router on which we configured a Static Port Address Translation (PAT) feature for ports 23, 500, and 4500. Apart from the normal Telnet port: 23, we also had to configure ports 500 and 4500 for the IKE and IPSec Dynamic tunnel. Our configuration can be seen in Example 4-2.

Example 4-2 Cisco NAPT configuration

```
provrtr01#sh run
Building configuration...

Current configuration : 1409 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname provrtr01
!
boot system flash c2600-ik9s-mz.122-15.t17.bin
boot system flash c2600-is-mz.122-11.t3.bin
logging queue-limit 100
enable secret 5 $1$1WC1$Vhn3KRUVOba5gobLe2Jbb.
enable password itsc
!
ip subnet-zero
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 ip nat outside
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
!
interface FastEthernet0/1
 ip address 10.10.2.200 255.255.255.0
 ip nat inside
 duplex auto
 speed auto
!
ip nat inside source static tcp 10.10.1.241 23 192.168.1.1 23 extendable
ip nat inside source static udp 10.10.1.241 500 192.168.1.1 500 extendable
ip nat inside source static udp 10.10.1.241 4500 192.168.1.1 4500 extendable
ip nat outside source list 1 interface FastEthernet0/1
ip http server
no ip http secure-server
ip classless
ip route 10.10.0.0 255.255.0.0 10.10.2.1
!
access-list 1 permit 192.168.1.0 0.0.0.255
!
snmp-server community public RO
snmp-server enable traps tty
call rsvp-sync
!
```

```
line con 0
line aux 0
line vty 0 4
 password swj43r
 login
!
!
end
```

Windows XP

We performed the steps in this section after completing the initial Windows XP IPSec configuration described in Chapter 3, “IP Security” on page 75. However, because our IPSec tunnel traversed a NAT device, we also had to enable NAT traversal support for Windows and therefore we discuss it here.

Attention: If these actions are not executed, the IPSec NAT traversal tunnel will not establish successfully.

1. We had Windows XP Service Pack 2 installed and due to a change in the default behavior for Windows IPSec NAT traversal, we had to make a modification to the Windows Registry. We recommend that you do a search for “NAT Traversal” on the “support.microsoft.com” Web site to determine whether your system requires similar changes. The details and description for our change can be found at the following link:

   http://support.microsoft.com/kb/885407/en-us

2. We also changed the Destination address in Figure 3-59 on page 172 under 3.6.4, “Setting up a Windows IPSec policy” on page 162, to reflect our Inside global IP address **198.168.1.1**. This address corresponds to the IP address configured on Interface FastEthernet0/0 of the Cisco device shown in Example 4-3 on page 219.

4.3.2 Testing and verification

To verify our configuration and functionality of the IPSec NAT traversal support for NAPT, we established a Telnet connection from our Windows client **192.168.1.253** to the outside global address 1**92.168.1.1** port **23**. The outside global address in return gets translated by the PAT on the Cisco router to **10.10.1.241** port **23**, which represent the z/OS server. This action created a dynamic IKE IPSec tunnel between the Windows client and the z/OS server, which confirmed the functionality of the z/OS V1R8 IPSec NAPT traversal support.

In the following steps, we provide relevant examples and descriptions of the snapshots that were taken during this verification stage after we successfully established the Telnet connection from the Windows client to the z/OS server:

1. The MSG10 screen on the client (Figure 4-12) was the first indication that we managed to establish the IPSec tunnel.

```
MSG10
        - International Technical Support Organization - ITSO




         Enter:
         NVAS      - NetView Access Services
         SCxxTS    - TSO on SCxx (fill in the "xx")




Your IP Address:     192.168.1.253                Your Telnet Port:  04147
-----------------------------------------------------Last Command:
LU: SC31TS04         Sense Code:                  Date: 08/16/06 Time: 08:52:28
```

Figure 4-12 MSG10 Screen on the Windows client

2. We logged on to the Cisco router and issued the **`show ip nat translation`** command. The output in Example 4-3 confirmed two very important points:
   - Our NAPT configuration was done correctly and performed as intended.
   - Further confirmation that the IPSec tunnel was successfully established over the NAPT device. Port 4500 represents the IPSec dynamic tunnel.

Example 4-3 Cisco NAT display

```
provrtr01#sh ip nat translation
Pro Inside global      Inside local       Outside local      Outside global
tcp 192.168.1.1:23     10.10.1.241:23     ---                ---
udp 192.168.1.1:4500   10.10.1.241:4500   ---                ---
udp 192.168.1.1:500    10.10.1.241:500    ---                ---
udp 192.168.1.1:4500   10.10.1.241:4500   192.168.1.253:4500 192.168.1.253:4500
provrtr01#
```

3. On the z/OS server, we issued a Netstat Telnet command, which confirmed that we had a connection established on port (04147) with the Windows client. The port number corresponds with the port number displayed on the MSG10 screen. See Example 4-4.

Example 4-4 Netstat Telnet display on z/OS

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          08:55:23
 Internal Telnet Server Status:
 Conn     State    BytesIn    BytesOut   ApplName LuName
 ----     -----    -------    --------   -------- ------
 00000130 Establsh 0000000032 0000000739          SC31TS01
   Foreign socket: ::ffff:192.168.1.253..04147
 ***
```

4. From the z/OS OMVS shell, we issued the **ipsec** command to verify both the IKE and Dynamic tunnel. First, we issued the **ipsec -p tcpipa -k** and secondly the **ipsec -p tcpipa -y** commands. The results from these commands are shown in Example 4-5 and Example 4-6 respectively.

Example 4-5 Output from the ipsec -p tcpipa -k display

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Wed Aug 16 08:53:33 2006
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a

TunnelID:                     K1
KeyExchangeRuleName:          All_Traffic_DES~5
KeyExchangeActionName:        All_Traffic_DES
LocalEndPoint:                10.10.1.241
LocalIDType:                  IPV4
LocalID:                      10.10.1.241
RemoteEndPoint:               192.168.1.253
RemoteIDType:                 IPV4
RemoteID:                     192.168.1.253
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Sha
EncryptionAlgorithm:          DES
DiffieHellmanGroup            1
AuthenticationMethod:         PresharedKey
InitiatorCookie:              0XEBFAC6490364F62A
ResponderCookie:              0XDA471394239D2C08
Lifesize:                     0K
CurrentByteCount:             544b
Lifetime:                     480m
LifetimeRefresh:              2006/08/16 15:05:08
LifetimeExpires:              2006/08/16 16:48:49
Role:                         Initiator
AssociatedDynamicTunnels:     1
NATTSupportLevel:             D2RFC
NATInFrntLclScEndPnt:         Yes
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     Yes
RmtNAPTDetected:              No
RmtUdpEncapPort:              4500
```

Example 4-6 Output from the ipsec -p tcpipa -y display

```
CS V1R8 ipsec  TCPIP Name: TCPIPA  Wed Aug 16 08:53:26 2006
Primary:  Dynamic tunnel  Function: Display            Format:   Detail
Source:   Stack           Scope:    Current            TotAvail: 1

TunnelID:                     Y2
ParentIKETunnelID:            K1
VpnActionName:                DES
LocalDynVpnRule:              n/a
State:                        Active
HowToEncap:                   Transport
LocalEndPoint:                10.10.1.241
RemoteEndPoint:               192.168.1.253
LocalAddressBase:             10.10.1.241
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
```

```
RemoteAddressBase:            192.168.1.253
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              708956256
 AuthOutboundSpi:             2506164033
HowToEncrypt:                 DES
 EncryptInboundSpi:           708956256
 EncryptOutboundSpi:          2506164033
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              27
OutboundBytes:                3127
InboundPackets:               37
InboundBytes:                 986
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2006/08/16 11:54:38
LifetimeExpires:              2006/08/16 12:48:49
CurrentTime:                  2006/08/16 08:53:27
VPNLifeExpires:               2006/08/17 08:48:22
NAT Traversal Topology:
  UdpEncapMode:               Yes
  LclNATDetected:             Yes
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    No
  RmtIsZOS:                   No
  zOSCanInitP2SA:             Yes
  RmtUdpEncapPort:            4500
  SrcNATOARcvd:               0.0.0.0
  DstNATOARcvd:               0.0.0.0
```

Debugging tools

We used the following commands and tools to gather debugging information depending on what type of problem we worked on.

- TCP/IP Netstat, Ping, Tracerte and displays
- Omproute (OSPF) d tcpip,tcpipa,omp,ospf,xxxxx displays
- Log files, MVS console log, Syslogd
- 2600 Router, show ip nat translations, debug ip nat detailed

5

Application Transparent Transport Layer Security

Transport Layer Security, or TLS, is the latest evolution of Secure Sockets Layer (SSL) technology. With it, you can encrypt and protect your most important e-commerce transactions and other data as it crosses the network.

Implementing and taking advantage of this highly secure approach used to require extensive programming changes to applications within the mainframe environment. With the availability of Application Transparent Transport Layer Security (AT-TLS), you can now deploy TLS encryption without the time and expense of re-coding your applications.

AT-TLS support is policy driven and is managed by a Policy Agent or PAGENT (discussed in Chapter 1, “Policy Agent” on page 3). Socket applications continue to send and receive clear text over the socket, but data sent over the network is protected by system SSL. Support is provided for applications that require awareness of AT-TLS for status or to control the negotiation of security.

This chapter discusses the topics listed in Table 5-1.

Table 5-1 Topics discussed in this chapter

Section	Topic
5.1, “AT-TLS definition” on page 224	We discuss the role AT-TLS plays in securing socket-based applications, basic concepts, and the different implementations that can be exploited.
5.2, “Why AT-TLS is important” on page 226	In this section, we discuss different uses for AT-TLS.
5.3, “Recommendations” on page 226	AT-TLS is not suitable for all socket applications. In this section, we make recommendations on applications such as those that already incorporate TLS security.
5.4, “Restrictions” on page 227	This section discusses AT-TLS application restrictions.
5.5, “How AT-TLS is implemented” on page 227	We discuss our experiences when we secured our Rexx IP socket application using AT-TLS.

© Copyright IBM Corp. 2007. All rights reserved.

5.1 AT-TLS definition

The Transport Layer Security (TLS) protocol provides transport layer security (authenticity, integrity, and confidentiality) for a secure connection between two applications. The TLS protocol begins with a handshake, in which the two applications agree on a cipher suite, a group of cryptographic algorithms they will use for authentication, and session encryption. Once the client and server applications have negotiated a cipher suite, they authenticate each other and generate a session key. The session key is used to encrypt and decrypt all data traffic sent between the client and the server.

Implementing TLS protocols directly into applications (without using AT-TLS) requires modification to incorporate a TLS toolkit. These toolkits are available for limited programming environments only, and do not incorporate zSeries capabilities such as RACF key rings and digital certificates. AT-TLS supports existing socket applications without any change, providing TLS support on behalf of these applications.

Note: AT-TLS only supports TCP-based applications; it cannot be used to provide security for UDP-based applications. To provide security for UDP-based applications, consider taking advantage of IP Security (IPSec) support (discussed in Chapter 3, “IP Security” on page 75).

Although AT-TLS can be used to provide security for the majority of applications transparently, some applications need to control the security functions being performed by TCP/IP. This communication between the application and AT-TLS is done through the transparent TLS API using the SIOCTTLSCTL Input/Output Control (IOCTL) macros.

5.1.1 Basic concepts

AT-TLS provides application-to-application security using policies. The policies are defined and loaded into the stack by Policy Agent. When AT-TLS is enabled and a newly established connection is first used, the TCP layer of the stack searches for a matching AT-TLS policy. If no policy is found, the connection is made without AT-TLS involvement. If a policy is found, a sequence like the one illustrated in Figure 5-1 is followed.

Figure 5-1 AT-TLS basic flow

The flow is:

1. The client connection to the server gets established.
2. The server sends data in the clear and the TCP layer queues it.
3. The TCP layer invokes System SSL to perform an SSL handshake under the identity of the server.
4. The TCP layer invokes System SSL to encrypt the queued data and sends it to the client.
5. The client then sends encrypted data and the TCP layer invokes System SSL to decrypt the data.
6. The server receives data in the clear.

When AT-TLS is enabled, statements in the Policy Agent define the security attributes for connections that match AT-TLS rules. This policy-driven support can be deployed transparently underneath many existing sockets, leaving the application unaware of the encryption and decryption being done on its behalf. Support is also provided for applications that need to negotiate TLS or need to participate in client authentication; however, these applications must be aware of AT-TLS and use IOCTL support (see “Controlling applications” in 5.1.2, “AT-TLS application types” on page 225). AT-TLS supports the TLS, SSLv3, and SSLv2 protocols.

IOCTL support is provided for applications that need to be aware of AT-TLS for status or to control the negotiation of security. TLS can be requested by applications where the application issues AT-TLS API calls to indicate that a connection should start or stop using TLS. Client identification services are also available for applications where TLS API calls are used to receive user identity information based on X.509 client certificates. SIOCTTLSCTL is an IOCTL specifically available with AT-TLS for applications to control AT-TLS for a connection. Applications can do things such as initializing a connection (TTLS_INIT_CONNECTION), resetting a connection (TTLS_RESET_CONNECTION), and resetting ciphers, using the SIOCTTLSCTL IOCTL macros.

Note: Be careful to coordinate your use of AT-TLS with other application-specific encryption implementations; otherwise, you could end up encrypting the same data twice: First by the application and then by AT-TLS. If possible, use AT-TLS as a consistent security solution for all of your TCP-based applications. Considerations for the general use of AT-TLS for application security are discussed in 5.1.2, “AT-TLS application types” on page 225, and 5.3, “Recommendations” on page 226.

5.1.2 AT-TLS application types

Applications have different requirements concerning security. Some applications need to be aware of when a secure connection is being used. Others may need to assume control if and when a TLS handshake occurs. For this reason, there are different application types supported by AT-TLS. These application types include:

- Not enabled applications
  - Pascal API and Web Fast Response Cache Accelerator (FRCA) applications are not supported by AT-TLS.
  - When there are no AT-TLS policies in place (including applications that start during the InitStack window) or if the policy explicitly says `enabled off`.
  - Applications such as FTP and Telnet, which may use either AT-TLS or the SSL/TLS toolkit directly.

Note: Applications such as FTP and Telnet have already been programmed to use the SSL/TLS toolkit directly and can provide additional functions (such as application-negotiated encryption and certificate-based user ID mapping) that cannot be used with AT-TLS without application changes to utilize the SIOCTTLSCTL IOCTL. If, however, you do not need those additional functions, you will be better off leveraging AT-TLS as a consistent solution for *all* of your TCP applications.

- Basic applications
  - The AT-TLS policy says `enabled on`.
  - The application is unchanged and unaware of AT-TLS.
- Aware applications
  - The AT-TLS policy says `enabled on`.
  - The application is changed to use the SIOCTTLSCTL IOCTL to extract AT-TLS information.
- Controlling applications
  - The application protocol may negotiate the use of TLS in cleartext prior to starting a secure session.
  - Where the policy says `enabled on` and `ApplicationControlled on`.
  - The application is changed to use SIOCTTLSCTL IOCTL to extract and control AT-TLS.

5.1.3 For additional information

For additional information regarding AT-TLS, refer to the *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.

5.2 Why AT-TLS is important

AT-TLS provides security for your TCP-based applications without the development costs of implementing security directly into your applications. Because AT-TLS can be used as a consistent solution across *all* of your TCP-based applications, systems administrators reap the benefits of improved productivity and infrastructure simplification with streamlined management.

5.3 Recommendations

If possible, use AT-TLS as a consistent security solution for all of your TCP-based applications. Certain applications (such as FTP and Telnet), however, have already been programmed to use the SSL/TLS toolkit directly and provide additional security functions (such as application-negotiated SSL/TLS and certificate-based user ID mapping) that cannot be used with AT-TLS without application changes to utilize the SIOCTTLSCTL IOCTL. If, however, you do not need those additional functions, you will be better off leveraging AT-TLS as a consistent solution for most, if not *all*, of your TCP applications.

Note: With the current *native* SSL/TLS support in FTP, an application can negotiate the use of SSL/TLS using an FTP protocol exchange known as the AUTH command. Because FTP is not yet enabled to be an AT-TLS controlling application, in order to use AT-TLS to secure FTP file transfers (rather than just using the current native SSL/TLS support), you would need to use *implicit* SSL/TLS. With implicit SSL/TLS, the fact that SSL/TLS is used is hidden from the FTP application and a specific port (TCP port 990) must be used. The use of TCP port 990 *implicitly* requires the use of SSL/TLS encryption.

Use of application-negotiated SSL/TLS is recommended by the IETF over the use of implicit SSL/TLS; however, implicit SSL/TLS might provide an acceptable *tactical* solution in your environment, allowing you to try to standardize on a single consistent encryption solution.

5.4 Restrictions

The following applications will not map to AT-TLS policies and are not supported by AT-TLS:

- Applications using the Pascal API to access TCP/IP:
  - Line Print daemon and commands LPD, LPQ, LPRM
  - Simple Mail Transfer Protocol (JES Spool Server)
  - TSO Telnet client
- Web servers using Fast Response Cache Accelerator
- Network administration applications permitted to the EZB.INITSTACK RACF profile
  - Connections established and mapped prior to the installation of the AT-TLS policy will proceed in cleartext.
  - Connections established and mapped after installation of the AT-TLS policy are subject to the installed policy.

Those applications that are not supported by AT-TLS will be permitted to proceed in cleartext.

5.5 How AT-TLS is implemented

Based upon our recommendations from 5.3, “Recommendations” on page 226, we show implementation details for using AT-TLS with a Rexx socket server application running on SC30/A23 using stack TCPIPA connecting to a Rexx socket client application running on SC31/A24 using stack TCPIPA.

The AT-TLS policies are provided to the stack by the Policy Agent. The Policy Agent thus has to be set up prior to activating AT-TLS.

We set up our Policy Agent configurations files, which pertain to AT-TLS as follows. We coded a CommonTTLSConfig statement naming file /etc/sc30.pagent_CommonTTLS_conf containing AT-TLS objects shared across our TCP/IP stacks. This file contains the following policy statements:

- TTLSRule statements
- TTLSGroupAction statements
- TTLSEnvironmentAction statements
- TTLSConnectionAction statements

We also coded TcpImage statements naming a file /etc/sc30.tcpip#_image.conf for each TCP/IP stack. Each one of these stack TTLSConfig files contains a TTLSConfig statement

that points to TCPIP.SC30.POLICIES(TTLS#). This file contains the following policy statements:

- TTLSRule statements
- TTLSGroupAction statements
- TTLSEnvironmentAction statements
- TTLSConnectionAction statements

Figure 5-2 gives a diagrammatic representation of how these config files are interlinked.

Figure 5-2 AT-TLS PAGENT config file relationship

Setting up Policy Agent and its associated security aspects is discussed in detail in Chapter 1, “Policy Agent” on page 3.

The following RACF aspects are important for the successful implementation of AT-TLS:

- Setting up TTLS Stack Initialization access control
- Enabling CSFSERV resources
- Creating digital certificates and key rings

5.5.1 Rexx socket application scenario

We set up a Rexx socket client and server application on our two z/OS machines, SC30 and SC31, as shown in Figure 5-3 on page 229, in order to demonstrate how this application can make use of the TLS protocol without requiring changes. The server application runs under the job name of APISERV, and repeatedly accepts connections, writes out socket end-point information, and returns this data to the client application.The server binds to port 7000. The

client application, APICLN, runs as a batch job on SC31, sends data to the server on SC30, and receives returned data.

Figure 5-3 AT-TLS Rexx socket APPLs

We performed the following steps to set up AT-TLS for our Rexx client/server socket applications:

1. Configuring the server policies
2. Configuring the client policies
3. Defining the digital certificates and key rings
4. Enabling CSFSERV resources
5. Controlling access during the window period
6. Enabling AT-TLS in the TCP/IP profile

Configuring the server policies

We used the IBM Configuration Assistant for z/OS Communication Server to create our AT-TLS policy file for our server on SC30.

We performed the following steps:

1. Adding a traffic descriptor object for REXX server
2. Adding a requirement map object for the REXX server
3. Adding a z/OS image for the REXX server (in our case: SC30)
4. Adding a TCP/IP stack to the z/OS image (in our case: TCPIPA)
5. Adding a connectivity rule

Perform the following steps:

1. Open the IBM Configuration Assistant for z/OS Communication Server. Select **File** → **Open** → **Create a New Configuration**.
2. In the Information panel, click **OK**.

3. In the Indicate a Directory and File Name for the New Configuration panel, enter a file name for the configuration file and click **Save**.
4. In the Welcome panel, select **AT-TLS only**, as shown in Figure 5-4. Click **OK**.

Figure 5-4 Welcome panel - selecting AT-TLS only policy

Before defining the z/OS images, TCP/IP stacks and the connectivity rules, you should define the traffic descriptors and the requirement maps that you are going to use for the connectivity rules.

Adding a traffic descriptor object for REXX server

To add a traffic descriptor object for REXX server:

1. Select **AT-TLS** → **Work with Reusable Objects** → **Traffic Descriptors** from the Configuration Assistant Navigation Tree. Click **Add**.
2. In the Traffic Descriptor Object panel, enter a name and description for the new traffic descriptor and click **Add**.

3. In the Traffic Type Details panel, enter the details for the traffic descriptor. The server traffic descriptor screen shown in Figure 5-5 contains the AT-TLS handshake role button, which we set to Server. We are also using the key ring as defined on our z/OS image. We define our connection direction as incoming from all ephemeral ports to local port 7000.

   Click **OK**.

Figure 5-5 Server traffic descriptor

4. The Traffic Descriptor Object panel should now look like the panel shown in Figure 5-6. Click **OK**.

Figure 5-6 Traffic Descriptor Object panel: Adding a traffic descriptor for REXX server

Adding a requirement map object for REXX server

To add a requirement map object for REXX server:

1. Select **AT-TLS** → **Work with Reusable Objects** → **Requirement Maps** from the Configuration Assistant Navigation Tree.
2. In the Configuration Assistant - Requirement Maps panel, click **Add**.
3. In the Requirement Map panel, enter a name and a description for the requirement map. In addition, select the traffic descriptor you want to configure in this requirement map object and click **Add**. Select the right AT-TLS - Security Level for the traffic descriptor that you added by clicking the drop-down list. For example, we defined our server traffic descriptor with the supplied AT-TLS GOLD security level, which uses the following ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F and TLS_RSA_WITH_AES_128_CBC_SHA, as shown in Figure 5-7. Click **OK**.

Figure 5-7 Server requirement map: Adding a requirement map for REXX server

Adding a z/OS image for the REXX server

To add a z/OS image for the REXX server:

1. In the Configuration Assistant - Current Configuration panel, click **Add a New z/OS Image**.
2. In New z/OS Image: Welcome panel, click **Next**.

3. In New z/OS Image: Information panel, enter a name for z/OS image and description and click **Next**, as shown in Figure 5-8.

Figure 5-8 New z/OS Image: Information panel - Adding SC30

4. In New z/OS Image: AT-TLS Image Level Settings panel, enter the settings for the key ring database. We selected **Key ring is in SAF product** and entered the name of the key ring that we created in RACF, ATTLS_keyring. We selected all the trace levels available: **Level 1 - Errors (to TCP/IP Joblog)**, **Level 2 - Errors (to Syslog)**, and **Level 4 - Information (to Syslog)**, as shown in Figure 5-9. Click **Next**.

Figure 5-9 New z/OS Image: AT-TLS Image Level Settings panel

5. In the New z/OS Image: Finish panel, click **Finish**.
6. In the Proceed to the next step? panel, click **Yes**.

Adding a TCP/IP stack to the z/OS image

To add a TCP/IP stack to the z/OS image:

1. In the New TCP/IP Stack Wizard: Welcome panel, click **Next**.
2. In the New TCP/IP Stack: Name panel, enter the TCP/IP stack name and description (in our case: TCPIPA), as shown in Figure 5-10. Click **Next**.

Figure 5-10 New TCP/IP Stack: Name panel - Adding TCPIPA

3. In the New TCP/IP Stack: Finish panel, click **Finish**.
4. In the Proceed to the next step? panel, click **Yes**.

Adding a connectivity rule

To add a connectivity rule:

1. In the Connectivity Rule: Welcome panel, click **Next**.
2. In the Connectivity Rule: Data Endpoints panel, enter the endpoints data and a rule name, as shown in Figure 5-11. Click **Next**.

Figure 5-11 Connectivity Rule: Data Endpoints panel - Creating a rule between SC30 and SC31

3. In Connectivity Rule: Select Requirement Map panel, select the requirement map that you created, as shown in Figure 5-12. Click **Next**.

Figure 5-12 Connectivity Rule: Select Requirement Map panel

4. In the Connectivity Rule: Additional Settings, click **Finish**.

The resultant policy file for SC30 is displayed in Example 5-1. An explanation of these policies, along with the changes, follows.

Example 5-1 Server AT-TLS policy for TCPIPA on SC30

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC30
##    Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created = Wed Aug 09 13:04:34 EDT 2006
##
## Copyright =  None
##
TTLSRule                          ATTLS_SC30_SC31~1 1
{
  LocalAddrRef                    addr1
  RemoteAddrRef                   addr2
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Direction                       Both
```

```
  Priority                        255
  TTLSGroupActionRef              gAct1~REXXServer
  TTLSEnvironmentActionRef        eAct1~REXXServer
  TTLSConnectionActionRef         cAct1~REXXServer
}
TTLSGroupAction                   gAct1~REXXServer
{
  TTLSEnabled                     On 2
  Trace                           7
}
TTLSEnvironmentAction             eAct1~REXXServer
{
  HandshakeRole                   Server 3
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~A23
}
TTLSConnectionAction              cAct1~REXXServer
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~AT-TLS__Gold
  Trace                           7
}
TTLSKeyringParms                  keyR~A23
{
  Keyring                         ATTLS_keyring 4
}
TTLSCipherParms                   cipher1~AT-TLS__Gold 5
{
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddr                            addr1
{
  Addr                            10.10.1.230 6
}
IpAddr                            addr2
{
  Addr                            10.20.10.101 7
}
PortRange                         portR1
{
  Port                            7000 8
}
PortRange                         portR2
{
  Port                            1024-65535 9
}
```

1 The TTLSRule statement is used to define an AT-TLS rule. This policy is defined for inbound connections only and has been given the highest possible priority of 255 for the duration of our testing.

2 TTLSEnabled is the statement that turns on the AT-TLS function.

3 This statement controls who initiates the handshake. In the server role, AT-TLS will wait for an inbound hello from the client SSL handshake, which is performed like a server's handshake.

4 The key ring used was permitted to the user ID under which the Rexx server started task was running. Even though the TCP/IP stack itself does the SSL calls, the security environment under which the calls execute is that of the application.

5 The AT-TLS__Gold cipher was selected, including the cipher specifications for this AT-TLS session.

6 This is the SC30 VIPA address.

7 This is the client source IP address; in our case, it is SC31 DYNAMICXCF address.

8 The destination port used for testing was 7000.

9 The source port used for testing was all ephemeral ports.

Configuring the client policies

We used the IBM Configuration Assistant for z/OS Communication Server to create our AT-TLS policy file for our client on SC31.

We performed the following steps:

1. Adding a traffic descriptor object for REXX client
2. Adding a requirement map object for the REXX client
3. Adding a z/OS image for the REXX client (in our case: SC31)
4. Adding a TCP/IP stack to the z/OS image (in our case: TCPIPA)
5. Adding a connectivity rule

As you can see, the steps are very similar to the steps for configuring the server policies. The resultant policy file for SC31 is displayed in Example 5-2. The outbound connection direction for this client is reflected in this policy.

Example 5-2 Client AT-TLS policy for TCPIPA on SC31

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC31
##    Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created = Wed Aug 09 13:07:32 EDT 2006
##
## Copyright =  None
##
TTLSRule                          ATTLS_SC31_SC30~1
{
  LocalAddrRef                    addr1
  RemoteAddrRef                   addr2
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Direction                       Outbound
  Priority                        255
  TTLSGroupActionRef              gAct1~REXXClient
  TTLSEnvironmentActionRef        eAct1~REXXClient
  TTLSConnectionActionRef         cAct1~REXXClient
}
TTLSGroupAction                   gAct1~REXXClient
{
```

```
  TTLSEnabled                     On
  Trace                           7
}
TTLSEnvironmentAction             eAct1~REXXClient
{
  HandshakeRole                   Client
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~A24
}
TTLSConnectionAction              cAct1~REXXClient
{
  HandshakeRole                   Client
  TTLSCipherParmsRef              cipher1~AT-TLS__Gold
  Trace                           7
}
TTLSKeyringParms                  keyR~A24
{
  Keyring                         tlsKeyring
}
TTLSCipherParms                   cipher1~AT-TLS__Gold
{
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddr                            addr1
{
  Addr                            10.20.10.101
}
IpAddr                            addr2
{
  Addr                            10.10.1.230
}
PortRange                         portR1
{
  Port                            1024-65535
}
PortRange                         portR2
{
  Port                            7000
}
```

The above policy resulted from the use of the IBM Configuration Assistant for z/OS Communication Server. The AT-TLS key ring was defined in the AT-TLS Image Level Settings screen on our SC31 z/OS Image exactly as defined for SC30 in Figure 5-9 on page 235. We used a shared RACF database, which results in the ATTLS_keyring also being shared between the two systems.

The client traffic descriptor screen shown in Figure 5-13 contains the AT-TLS handshake role button, which we set to Client. We also used the key ring as defined on our z/OS image. We define our connection direction as outbound from all ephemeral ports to remote port 7000.

Figure 5-13 Client traffic descriptor

We defined our Client traffic descriptor with the supplied AT-TLS GOLD security level, which uses the following ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F and TLS_RSA_WITH_AES_128_CBC_SHA, as shown in Figure 5-14.

Figure 5-14 Client requirement map

It is important to note that AT-TLS functions at a different level from IP filtering and VPN policies. There is thus no need to integrate the AT-TLS policies into the rules used for VPN and filtering.

Defining the digital certificates and key rings

We created a key ring called ATTLS_keyring and connected a certificate authority certificate and server certificate to it, as shown in Example 5-3. We used a shared RACF database and therefore a shared key ring, so we were not required to export our CA certificate to a client key ring. The CA certificate was defined as TRUSTED, which made our server certificate TRUSTED as well.

Example 5-3 Defining our digital certificates and key ring

```
RACDCERT ID(CS09) addring(ATTLS_keyring)

SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)

RACDCERT ID(cs09) CERTAUTH GENCERT                    -
SUBJECTSDN( O('I.B.M Corporation')                    -
            CN('itso.ibm.com')                        -
            C('US')) TRUST                            -
            WITHLABEL('LOCALCA')                      -
```

```
            KEYUSAGE(certsign)

SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)

RACDCERT ID(CS09) GENCERT                              -
         SUBJECTSDN  (CN('SC30ServerCert')             -
                      OU('ITSO')                       -
                      C('US'))                         -
RACDCERT ID(CS09) GENCERT                              -
         SUBJECTSDN  (CN('SC30ServerCert')             -
                      OU('ITSO')                       -
                      C('US'))                         -
                      WITHLABEL('SC30ServerCert')      -
                      SIGNWITH(CERTAUTH                -
                      label('LOCALCA'))

SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

RACDCERT ID(CS09) CONNECT(ID(CS09)                     -
                     LABEL('SC30 Server Certificate')  -
                     RING(ATTLS_keyring)               -
                     USAGE(personal))
   RACDCERT ID(CS09) CONNECT(ID(CS09) CERTAUTH         -
                     LABEL('LOCALCA')                  -
                     RING(ATTLS_keyring)               -
                     USAGE(certauth))
```

Enabling CSFSERV resources

If you are using cryptographic hardware in conjunction with TLS security, and you have defined resources in the CSFSERV classes to protect cryptographic services, you should permit the user ID associated with the server to these resources.

With AT-TLS, the system SSL verifies that the user ID associated with the server is permitted to use CSFSERV resources. We defined the CSFDSV and CSFPKE services and permitted the RACF user ID CS09 to use the CSFSERV resource class, as shown in Example 5-4.

Example 5-4 Enabling CSFSERV resources

```
//RACFDEF EXEC PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
RDEFINE CSFSERV CSFDSV UACC(NONE)
RDEFINE CSFSERV CSFPKE UACC(NONE)
SETROPTS RACLIST(CSFSERV) REFRESH
PERMIT CSFDSV CLASS(CSFSERV) ID(CS09) ACCESS(READ)
PERMIT CSFPKE CLASS(CSFSERV) ID(CS09) ACCESS(READ)
SETROPTS RACLIST(CSFSERV) REFRESH
/*
```

Controlling access during the window period

When AT-TLS is enabled, the INITSTACK profile must be defined. The Policy Agent and any socket-based programs it requires must be permitted to this resource. Other programs or users that do not need to wait for the TTLS policy to be installed in the stack may be permitted to this resource. Users who are not permitted to this resource will not be able to open sockets

on this stack until the TTLS policy is installed. When the resource is not defined, no stack access is permitted. We defined this profile for SC30 and SC31, as shown in Example 5-5.

Example 5-5 Setup TTLS stack initialization access control for SC30 and SC31

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE  SERVAUTH EZB.INITSTACK.SC30.TCPIPA UACC(NONE)
PERMIT   EZB.INITSTACK.SC30.TCPIPA CLASS(SERVAUTH) ID(*) ACCESS(READ) -
          WHEN(PROGRAM(PAGENT,EZAPAGEN))
RDEFINE  SERVAUTH EZB.INITSTACK.SC31.TCPIPA UACC(NONE)
PERMIT   EZB.INITSTACK.SC31.TCPIPA CLASS(SERVAUTH) ID(*) ACCESS(READ) -
          WHEN(PROGRAM(PAGENT,EZAPAGEN))
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

Setting up the profile

To activate AT-TLS, the TTLS parameter has to be added to the TCPCONFIG profile config statement, as shown in Example 5-6.

Example 5-6 Profile statement to enable AT-TLS

```
TCPCONFIG TTLS
```

AT-TLS operability verification

We installed our policies on the client and server side, started PAGENT as shown in Example 5-7, and started our APISERV application on SC30, and our APICLN application on SC31.

Example 5-7 PAGENT startup on SC30

```
000090  $HASP373 PAGENT   STARTED
000090  EZZ8431I PAGENT STARTING
000090  EZZ8432I PAGENT INITIALIZATION COMPLETE
000090  EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : TTLS
```

We did a NETSTAT TTLS display on the client side, as shown in Example 5-8, to determine whether the stack mapped a connection to our client AT-TLS policy and, if so, to which policy it was mapped.

Example 5-8 Display result of NETSTAT TTLS on client

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          07:41:06
 TTLSGrpAction                              Group ID           Conns
 ----------------------------------------  -----------------  -----
 gAct1~REXXClient                           00000002               0
```

We did a NETSTAT TTLS display on the server side, as shown in Example 5-9, to determine whether the stack mapped a connection to our server AT-TLS policy and, if so, to which policy it was mapped.

Example 5-9 Display result of NETSTAT TTLS on server

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          07:42:30
TTLSGrpAction                             Group ID           Conns
---------------------------------------  -----------------  -----
gAct1~REXXServer                          00000002               0
```

Our NETSTAT ALLCONN command on SC30 showed that the APISERV application was listening on port 7000 (see Example 5-10).

Example 5-10 NETSTAT ALLCONN on server

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          07:48:47
 User Id  Conn     State
 -------  ----     -----
 APISERV  00004765 Listen
   Local Socket:   0.0.0.0..7000
   Foreign Socket: 0.0.0.0..0
:
:
```

Our NETSTAT ALL command in Example 5-11 on SC31 showed that our client application APICLN connected to IP address 10.10.1.230 and port 7000 from the client IP address 10.20.10.101 ephemeral port 1041.

Example 5-11 Display results of NETSTAT ALL on client

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          07:38:19
Client Name: APICLN                   Client Id: 00004C0B
   Local Socket: 10.20.10.101..1041
   Foreign Socket: 10.10.1.230..7000
 BytesIn:            00000000000000000031
 BytesOut:           00000000000000000031
 SegmentsIn:         00000000000000000011
 SegmentsOut:        00000000000000000013
 Last Touched:       07:38:13          State:              TimeWait
 RcvNxt:             3475413650        SndNxt:             3489619133
 ClientRcvNxt:       3475412176        ClientSndNxt:       3489618831
 InitRcvSeqNum:      3475412144        InitSndSeqNum:      3489618799
 CongestionWindow:   0000065120        SlowStartThreshold: 0000065535
 IncomingWindowNum:  3475446389        OutgoingWindowNum:  3489651872
 SndWl1:             3475413650        SndWl2:             3489619133
 SndWnd:             0000032739        MaxSndWnd:          0000032768
 SndUna:             3489619133        rtt_seq:            3489619103
 MaximumSegmentSize: 0000008140        DSField:            00
 Round-trip information:
   Smooth trip time: 0.000             SmoothTripVariance: 201.000
 ReXmt:              0000000000        ReXmtCount:         0000000000
 DupACKs:            0000000000
 SockOpt:            8000              TcpTimer:           0C
 TcpSig:             00                TcpSel:             C0
 TcpDet:             E0                TcpPol:             02
 QOSPolicyRuleName:
 TTLSPolicy:         Yes
   TTLSRule:         ATTLS_SC31_to_SC30~1
   TTLSGrpAction:    gAct1~REXXClient
   TTLSEnvAction:    eAct1~REXXClient
   TTLSConnAction:   cAct1~REXXClient
 ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
```

Our NETSTAT ALL command on SC30 showed that our server application APISERV was listening on port 7000 (see Example 5-12).

Example 5-12 Display results of NETSTAT ALL

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          07:40:53
 Client Name: APISERV                 Client Id: 00000037
   Local Socket: 0.0.0.0..7000
Foreign Socket: 0.0.0.0..0
  BytesIn:            00000000000000000000
  BytesOut:           00000000000000000000
  SegmentsIn:         00000000000000000000
  SegmentsOut:        00000000000000000000
  Last Touched:       07:40:43          State:              Listen
  RcvNxt:             0000000000        SndNxt:             0000000000
  ClientRcvNxt:       0000000000        ClientSndNxt:       0000000000
  InitRcvSeqNum:      0000000000        InitSndSeqNum:      0000000000
  CongestionWindow:   0000000000        SlowStartThreshold: 0000000000
  IncomingWindowNum:  0000000000        OutgoingWindowNum:  0000000000
  SndWl1:             0000000000        SndWl2:             0000000000
  SndWnd:             0000000000        MaxSndWnd:          0000000000
  SndUna:             0000000000        rtt_seq:            0000000000
  MaximumSegmentSize: 0000000536        DSField:            00
  Round-trip information:
    Smooth trip time: 0.000             SmoothTripVariance: 1500.000
  ReXmt:              0000000000        ReXmtCount:         0000000000
  DupACKs:            0000000000
  SockOpt:            8000              TcpTimer:           00
  TcpSig:             00                TcpSel:             00
  TcpDet:             C0                TcpPol:             00
  QOSPolicyRuleName:
ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
    ConnectionsIn:      0000000001        ConnectionsDropped: 0000000000
    CurrentBacklog:     0000000000        MaximumBacklog:     0000000010
    CurrentConnections: 0000000000        SEF:                100
    Quiesced: No
```

Problem determination

The NETSTAT command can aid in problem determination and assist in checking the status of your connections. The following functions that pertain to AT-TLS are available:

- NETSTAT ALL
- NETSTAT ALLCONN
- NETSTAT TTLS
- pasearch -t

Other useful problem determination aids are:

- Reviewing SYSLOGD
- Running a CTRACE with option TCP or a packet trace
- Setting debug traces using the TTLSConnectionAction statement

The trace value is interpreted by AT-TLS as a bit map. Each of the options is assigned a value that is a power of 2, as shown in Table 5-2. You should add together the values of each option that you want to activate.

The default trace value is 2, which provides error messages to syslogd. When you are deploying a new policy, you might find it beneficial to specify a trace value of 6 or 7. This provides connection information messages, in addition to error messages in syslogd. The

information messages provide positive feedback that connections are mapping to the intended policy. Trace options event (8), flow (16), and data (32) are intended primarily for diagnosing problems. Trace values larger than 7 can cause a large number of trace records to be dropped instead of being sent to syslogd.

Table 5-2 Trace values and descriptions

Trace value	Description
0	No tracing is enabled.
1	Errors are traced to the TCP/IP joblog.
2	Errors are traced to syslogd.
4	Tracing of when a connection is mapped to an AT-TLS rule and when a secure connection is successfully initiated is enabled.
8	(Event) Tracing of major events is enabled.
16	(Flow) Tracing of system SSL calls is enabled.
32	(Data) Tracing of encrypted negotiation and headers is enabled.
64	Reserved.
128	Reserved.
255	All Tracing is enabled.

For AT-TLS codes that are above 5000, refer to the *z/OS V1R8.0 Communications Server: IP Diagnosis Guide*, GC31-8782, or to the appropriate IP messages manual.

For codes below 5000, refer to the manual *z/OS V1R8.0 Cryptographic Services System SSL Programming*, SC24-5901.

6

Intrusion Detection Services

Intrusion is a term describing undesirable activities. The objective of an intrusion may be to acquire information that a person is not authorized to have. It may be to gain unauthorized use of a system as a stepping stone for further intrusions elsewhere. It may also be to cause business harm by rendering a network, system, or application unusable. Most intrusions follow a pattern of information gathering, attempted access, and then destructive attacks. Intrusion Detection Services (IDS) thus guards against these intrusions, thereby providing protection against potential hackers.

This chapter discusses the topics discussed in Table 6-1.

Table 6-1 Topics discussed in this chapter

Section	Topic
6.1, “What IDS is” on page 250	This section covers the different types of intrusions, and how policies are used to fend them off.
6.2, “Basic concepts” on page 251	This section details are given about the scan detection, attack detection, and traffic regulation.
6.3, “How IDS is implemented” on page 259	This section covers the use of eServer IDS Configuration Manager to create IDS policies, which are loaded into Lightweight Directory Access Protocol (LDAP).

© Copyright IBM Corp. 2007. All rights reserved.

6.1 What IDS is

IDS is a z/OS Communications Server security protection mechanism that inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or to compromise a system. IDS can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. It can also provide a reactive system whereby IDS responds to the suspicious activity by taking policy actions.

As shown in Figure 6-1, in V1R8, a flat file for IDS policies has been implemented that will allow all policies to be stored in flat files, removing any requirements for an LDAP server environment. Policies are now stored on an LDAP server and are downloaded to the Policy Agent (PAGENT) or they can be stored directly in the Policy Agent. The Policy Agent in turn installs the policies in the stack. When an attack is identified, any of the following resultant policy actions can be taken:

- Event logging
- Statistics gathering
- Packet tracing
- Discarding of the attack packets

Some IDS policies log events and statistics in syslogd and the system console via Traffic Regulation Monitoring daemon (TRMD).

Figure 6-1 IDS architecture

6.2 Basic concepts

IDS functions can be subdivided into three areas:

- Scan detection
- Attack detection
- Traffic regulation

IDS is managed through policies. The policy is designed by the network administrator and based on preconceived events. The policy must include factors such as who, what, where, when, and how:

- *Who* is allowed to connect to the host?
- *What* applications/ports are clients allowed to use?
- *Where* is the attack/intruder/traffic emanating from?
- *When* should I consider something to be an attack or scan?
- *How* is my system affected by the attack, scan, or traffic?

In z/OS V1R8.0 Communications Server, IDS policies are supported in a Policy Agent configuration file as well as an LDAP server. This solution provides an IDS policy solution that is consistent with other policy types for those installations that do not have an LDAP infrastructure in place or that favor using configuration files instead of an LDAP configuration.

Note: The IBM Configuration Assistant for z/OS Communication Server is intended to replace the zIDS Manager for configuring IDS.

In this chapter, we cover only the Policy Agent configuration file that is created with the IBM Configuration Assistant for z/OS Communication Server.

The policy information is loaded into the Policy Agent application during PAGENT startup. All IDS policies allow the logging events to a specified message level in syslogd or the system console. Most IDS policies support discarding packets when a specified limit is reached. Most IDS policies support writing statistics records to the INFO message level of syslogd on a specified time interval or if exception events have occurred. All IDS policies support tracing all or part of the triggering packet to an IDS-specific CTRACE facility, SYSTCPIS. IDS assigns a correlator value to each event. Messages written to the system console and syslogd and records written to the IDS ctrace facility all use this correlator. A single detected event may involve multiple packets. The correlator value helps to identify which message and packets are related to each other.

The type of policy written can be a *scan policy*, *attack policy*, or *traffic regulation policy*. The following sections give detailed descriptions of each of these policies.

6.2.1 Scan policies

Scans are detected because of multiple information gathering events from a single source IP within a defined time frame. Scanning is not harmful and may be part of normal operation, but many serious attacks, especially access violation attacks, are preceded by information gathering scans. Due to the fact that scans use consistent source IP addresses, they can be monitored and the data processed to help prevent an attack or determine the origins of a previous attack.

The scanner is defined as a source host that accesses multiple unique resources (ports or interfaces) over a specified period of time. The number of unique resources (threshold) and the time period (interval) can be specified via policy. Two categories of scans are supported: Fast scans and slow scans.

Fast scan

During a fast scan many resources are rapidly accessed in a short time period (usually program driven and takes less than five minutes), as shown in Figure 6-2.

Figure 6-2 Fast scan

Slow scan

During a slow scan different resources are intermittently accessed over a longer period of time (many hours). This could be a scanner trying to avoid detection, as shown in Figure 6-3.

Figure 6-3 Slow scan

A fast scan scenario may be one in which an attack is based on the information provided through a program that loops through ports 1 - 1025 (normally the ports used by the server for listening ports), determining which ports have active listeners. This information may be the basis for a future attack. A slow attack is more deliberate; occasional packets may be sent out to different ports over a long period of time with the same fundamental purpose, obtaining host information.

The same port being accessed will not generate multiple event records, for example, if a client from the same source IP address generates 20 connections to port 23 (TN3270 server). This is not considered a scan because only one unique resource has been accessed.

Note: Scan policies do not provide the ability to reject a connection. The actual rejecting of the connection based on the source IP address must be configured in the Traffic Regulation policy or firewall.

Scan policy parameters

A scan policy provides the ability to control the following parameters that define a scan:

- Fast scan time interval
- Slow scan time interval
- Fast scan threshold
- Slow scan threshold
- Exclude well-known legitimate scanners via an exclusion list
- Specify a sensitivity level by port or port range (to reduce performance impacts)
- Notify the installation of a detected scan via console message or syslogd message
- Trace potential scan packets

The policy allows the user to set a sensitivity level. This is known as policy-specified sensitivity. This is used in parallel with the categorization of the individual packets to determine if a packet should be counted as a scan event. The event classification is a normal, possibly suspicious, or very suspicious event. This logic is used to control the performance impact and analysis load of scan monitoring by only counting those individual packets where the chart indicates a count value. This value is then added with the current count total of scan events and compared with the threshold value to determine if we have met or exceeded the threshold in a specified time interval.

Scan events

Scan events are classified into Internet Control Message Protocol (ICMP), UDP port, and TCP port scans categories. The scan categories are described here:

- ICMP scan

  ICMP requests (echo, information, time stamp, and subnet mask) are used to obtain or map network information. The type of ICMP request determines the event classification.

- TCP port scans

  TCP is a stateful protocol. There are many different events that may be classified as normal, possibly suspicious, or highly suspicious.

- UDP port scans

  UDP is stateless. The stack is unable to differentiate between a client port and a server port. A scanner sending messages to many ephemeral ports looks very similar to a DNS server sending replies to many clients on ephemeral ports. TCP/IP configuration allows UDP ports to be RESERVED, therefore restricting a port so that it cannot be used.

Any countable scan event will count against an origin source IP address. The total number of countable events from all categories is compared to the policy thresholds. When an origin source IP address has exceeded the policy-defined fast or slow threshold, an event may be sent to the TRMD for logging to syslogd, a console message may be issued, and optionally a packet trace record issued. This is all dependent upon the notification actions set in the action of the policy. Once a scan event is logged for a particular source IP address, no further scan events will be reportable within the specified fast interval. The intervals and thresholds for fast and slow scan are global. Only one definition of them is allowed across all event categories at a given point in time.

False positive scans

IDS attempts to reduce the recording of false scan events. This can be manually coded in the policy by excluding a source IP address, port, or subnet. This is useful if you have a particular client that probes the TCP/IP stack for general statistical information. Also, only unique events from a source IP address are counted as a scan event. An event is considered unique if the four-tuple, client IP address, client port, server IP address, and server port are unique, as well as the IP protocol for this scan interval. In the case of ICMP, a packet is unique if the type has not been seen before within this scan interval.

6.2.2 Attack policies

An *attack* is defined as an assault on system security that derives from an intelligent threat. It is an intelligent act that is a deliberate attempt to evade security services and violate the security policy of a system. An attack may in the form of a single packet or multiple packets. There are two types of attacks, active and passive:

- An *active attack* is designed to alter system resources or affect their operation.
- A *passive attack* is designed to learn or make use of system information but not affect system performance.

For more information about passive and active attacks, refer to RFC 2828.

The attack policies designed for IDS are based on active attacks. One may consider scanning to be more of a passive attack.

IDS attack policy allows the network administrator to provide network detection for one or more categories of attacks independently of each other. In general, the types of actions that can be specified for an attack policy are notifications (that is, event logging, statistics gathering, packet tracing) and discarding the attack packets.

IDS attack categories

The IDS categories of attacks are described in Table 6-2.

Table 6-2 Attack categories

Category	Attack description	Actions
Malformed packets	There are numerous attacks designed to crash a system's protocol stack by providing incorrect partial header information. The source IP address is rarely reliable for this type of attack.	► TCP/IP stack: Always discards malformed packets. ► IDS policy: May provide notification.
Inbound fragment restrictions	Many attacks are the result of fragment overlays in the IP or transport header. This support allows you to protect your system against future attacks by detecting fragmentation in the first 256 bytes of a packet.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.
IP protocol restrictions	There are 256 valid IP protocols. Only a few are in common usage today. This support allows you to protect your system against future attacks by prohibiting those protocols that you are not actively supporting.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.
IP option restriction	There are 256 valid IP options, with only a small number currently in use. This support allows you to prevent misuse of options that you are not intentionally using. Checking for restricted IP options is performed on all inbound packets, even those forwarded to another system.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.

Category	Attack description	Actions
UDP perpetual echo	Some UDP applications unconditionally respond to every datagram received. In some cases, such as Echo, CharGen, or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases, it may be polite application behavior to send error messages in response to incorrectly formed requests. If a datagram is inserted into the network with one of these applications as the destination and another of these applications spoofed as the source, the two applications will respond to each other continually. Each inserted datagram will result in another perpetual echo conversation between them. This support allows you to identify the application ports that exhibit this behavior.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause packet to be discarded.
ICMP redirect restrictions	ICMP redirect packets can be used to modify your routing tables.	► TCP/IP stack: Will discard ICMP redirects if IGNOREREDIRECT is coded in the tcpip.profile. ► IDS policy: May provide notification and disable redirects (this can optionally be coded as a parameter in the tcpip.profile).
Outbound raw restrictions	Most network attacks require the ability to craft packets that would not normally be built by a proper protocol stack implementation. This support allows you to detect and prevent many of these crafting attempts so that your system is not used as the source of attacks. As part of this checking, you can restrict the IP protocols allowed in an outbound RAW packet. It is recommended that you restrict the TCP protocol on the outbound raw rule.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.
TCP SYNflood	One common denial of service attack is to flood a server with connection requests from invalid or nonexistent source IP addresses. The intent is to use up the available slots for connection requests and thereby deny legitimate access from completing.	► TCP/IP stack: Provides internal protection against SYN attack. ► IDS policy: May provide notification.

Attack policy notification

The IDS attack policy (object class name ibm-idsNotification) notification allows attack events to be logged to syslogd and the system console. For all attack categories except flood, a single packet triggers an event. To prevent message flooding to the system console, you can specify the maximum number of console messages to be logged per attack category within a five-minute interval (ibm-idsMaxEventMessage). There is no default, therefore it is recommended that you code a maximum number of event messages that are to be written to the console. To prevent message flooding to syslogd, a maximum of 100 event messages per attack category will be logged to syslogd within a five-minute interval.

Note: The console messages provide a subset of the information provided in the syslogd messages.

Attack policy statistics

The IDS attack policy statistics action provides a count of the number of attack events detected during the statistics interval. The count of attacks is kept separately for each category of attack (for example, malformed), and a separate statistics record is generated for each. If you want to turn on statistics for attacks, it is recommended that you specify exception statistics (ibm-idsTypeActions:EXCEPTSTATS). With exception statistics, a statistics record will only be generated for the category of attack if the count of attacks is non-zero. If statistics are requested (ibm-idsTypeActions:STATISTICS), a record will be generated every statistics interval regardless of whether an attack has been detected during that interval.

6.2.3 Attack policy tracing

The IDS attack policy tracing uses the component trace facility SYSTCPIS. The attack policy tracing attributes are ibm-idsTraceData and ibm-idsTraceRecordSize, which indicate whether packets associated with the attack events are to be traced. For all attack categories except flood, a single packet triggers an event and the packet is traced. In the case of a flood, a maximum of 100 attack packets per attack category will be traced during a five-minute interval.

Note: In order to use the attack policy tracing via the ctrace component SYSTCPIS, the component must be started. See *z/OS V1R8.0 Communications Server: IP Diagnosis Guide*, GC31-8782, for more information.

6.2.4 Traffic Regulation policies

The IDS Traffic Regulation (TR) policies are used to limit:

- Memory usage
- Queue delay time

There are two types of TR policies, namely TCP and UDP Traffic Regulation policies.

TR TCP policy information

The IDS TR policies for TCP ports limit the total number of connections an application has active at one time. This can be used to limit the number of address spaces created by forking applications such as otelnetd. The TR TCP terminology is very important when coding the policy to ensure the desired goal is achieved. The following section describes TR TCP terminology.

Connections

Connections can be separated into two groups: Total connections and number of available connections.

Total connections

This is the total number of connections that are coded in your policy. This number can never be exceeded for a particular port.

Number of available connections

This is the total number of available connections, which is equal to the connections in use subtracted from the total connections. This value is used in the fair share algorithm.

Fair share algorithm

The fair share algorithm is designed to limit the number of connections available to any source IP address. The algorithm is based on the percentage of the available remaining connections for a particular port compared with the total connections already held by the source IP address for that port. The fair share equation and logic statements are shown in Example 6-1.

Example 6-1 Fair share logic

```
Equation Statement :
% SourceIPAddr = Num. of Conn. held by SourceIPAddr / Currently Available Sessions x 100

Logic Statement:
If %SourceIPAddr < Policy Percentage Then Allow the Session
Else Reject the
Session
```

Note: If a host does not currently have any connections open on the port and connections are available, a host will always be allowed at least one connection.

Quality of Service policy

Multi-user source IP addresses may be allowed a larger number of connections by specifying a Quality of Service (QoS) policy with a higher number of connections (MaxConnections) than allowed by the TR policy. TR will honor the QoS Differentiated Services Policy if the port is *not* in a *constrained state*. A QoS exception is made only when QoS Differentiated Services Policy is applied for the specific source server port and specific outbound client destination IP address.

Constrained state

TR TCP generates a constrained event when a port reaches approximately 90% of its connection limit (total connections). An unconstrained event is generated when the port falls below approximately 88% of its limit. This 2% deviance is designed to avoid message flooding.

TR UDP policy information

Traffic Regulation for UDP connections can be done in two ways: Through the UDPQUEUELIMIT parameter in the TCPIP.PROFILE or by coding a TR UDP policy. If both are in effect, the TR UDP policy takes priority.

UDPQUEUELIMIT

Traffic Regulation for UDP-based applications can be provided through the TCPIP.PROFILE statement of UDPQUELIMIT. This statement relates to inbound packets for bound UDP ports. Packets are queued until the queue limit is reached or buffer memory is exhausted. If NOUDPQUEUELIMIT is coded, any single bound port under a flood attack or with a stalled application could consume all available buffer storage. It is recommended that UDPQUEUELIMIT always be set to active. This limits the amount of storage that can be consumed by inbound datagrams for any single bound port. Sockets that use the Pascal API have a limit of 160 KB in any number of datagrams. Sockets that use other APIs have a limit of 2000 datagrams or 2880 KB.

Note: If a policy is in effect for a UDP port, the queue limit size is controlled by the policy for that port.

TR UDP policies

IDS TR policies for UDP ports specify one of four abstract queue sizes for specified bound IP addresses and ports. The four abstract sizes are VERY_SHORT, SHORT, LONG, and VERY_LONG. The abstract size comprises two values, the number of packets and the total number of bytes on the queue. If either one of these values is exceeded, inbound data is discarded. See Table 6-3 for the internal values.

Table 6-3 TR UDP abstract queue information

Abstract size	Number of packets	Queue limit
VERY_SHORT	16	32 KB
SHORT	256	512 KB
LONG	2048	4 MB
VERY_LONG	8192	16 MB

Most UDP applications have time-out values based on human perceptions of responsiveness. These values tend to stay constant while system processing speeds and network delivery speeds continue to advance rapidly. This may require the physical sizes of these queues to change over time. For performance reasons, sockets that use the Pascal API will only enforce the byte limit. Sockets that use other APIs will enforce both limits. Sockets without a policy specified for their port can use the existing UDPQUEUELIMIT mechanism.

For applications that can process datagrams at a rate faster than the average arrival rate, the queue acts as a speed matching buffer that shifts temporary peak workloads into following valleys. The more the application processing rate exceeds the average arrival rate and the larger the queue, the greater the variation in arrival rates that can be absorbed without losing work. Very fast applications with very bursty traffic patterns may benefit from LONG or VERY_LONG queue sizes.

For applications that consistently receive datagrams at a higher rate than they are able to process them, the queue acts to limit the effective arrival rate to the processing rate by discarding excess datagrams. In this case, the queue size only influences the average wait time of datagrams in the queue and not the percentage of work lost. In fact, if the wait time gets too large, the peer application may have given up or retransmitted the datagram before it is processed. Slow applications with consistently high traffic rates may benefit from SHORT queue sizes. In general, client-side applications will tend to have lower system priority, giving them lower datagram processing rates. They also tend to have much lower datagram arrival rates. Giving them SHORT or VERY_SHORT queue sizes may reduce the risk to system buffer storage under random port flood attacks with little impact on percentage of datagrams lost.

6.3 How IDS is implemented

IDS is implemented through policies. The Policy Agent is an integral part of setting up the environment for IDS to execute these policies. See Chapter 1, “Policy Agent” on page 3, for discussion and implementation examples for PAGENT.

The IBM Configuration Assistant for z/OS Communication Server graphical user interface (GUI) tool is the new flat file alternative for storing IDS policies. You may also change the IDS policy configuration file manually.

z/OS IDS comes with a set of beginner and advanced definition examples. These sample definitions are called:

- /usr/lpp/tcpip/samples/pagent_starter_IDS.ldif
- /usr/lpp/tcpip/samples/pagent_advanced_IDS.ldif

Using these samples as a base is a good way to start your IDS implementation. To learn more about modifying these definitions to create customized policies, consult the *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.

6.3.1 Installing the Policy Agent

PAGENT reads the configuration files that contain the IDS policy configuration statements, checks them for errors, and installs them into the TCP/IP stack. Setting up the PAGENT is described in Chapter 1, “Policy Agent” on page 3.

Note: You need superuser authority to start PAGENT, and the PAGENT executable modules must be in an APF-authorized library.

After setting it up, you need to define the IDSConfig statement to specify the path of the policy file that contains stack-specific IDS policy statements to PAGENT. Example 6-2 shows the IDS statements in the TCP/IP stack configuration file used by Policy Agent.

Example 6-2 The /etc/sc30.tcpipa_image.conf file with IDS configured

```
# This is a file that contains statements for TCP/IP stack TCPIPA in z/OS image SC30
# This file is used by Policy Agent.
#
# IDSConfig Statements
IDSConfig //'TCPIP.SC30.POLICIES(IDSA)'
```

6.3.2 The IBM Configuration Assistant for z/OS Communication Server

The IBM Configuration Assistant for z/OS Communication Server enables centralized configuration of intrusion detection policies for z/OS V1R8. This solution provides IDS policy solution that is consistent with other policy types for those installations that do not have an LDAP infrastructure in place or that favor using configuration files instead of an LDAP configuration.

The IBM Configuration Assistant for z/OS Communication Server is a tool designed to allow a network administrator to produce a configuration file for the Policy Agent (PAGENT).

IBM Configuration Assistant for z/OS Communication Server’s GUI provides a user-friendly front end for the entry of policy information. It also produces a configuration file with the information required by PAGENT. This file can be sent via FTP or moved to and placed in the PAGENT configuration file manually.

The IBM Configuration Assistant for z/OS Communication Server is a tool for network administrators. Therefore, before you begin you should:

- Read the chapter on policy-based networking in *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.
- Be familiar with your particular environment so that you can make decisions on what events are to be detected under what circumstances and what action to take.

6.3.3 Requirements and download instructions

This section outlines the requirements and support of the IBM Configuration Assistant for z/OS Communication Server.

Download and installation

The download and installation instructions are written for Windows. The following information and executables are located at:

http://www.ibm.com/software/network/commserver/zos/support/

6.3.4 Configuring IDS policy using the GUI

This section is intended to help the network administrator manage and understand the GUI provided. Each first-level directory will be discussed and screen captures provided to assist in the education. The sections are:

- Creating a new IDS policy
- Adding a z/OS image to the policy
- Adding a TCP/IP stack under the z/OS image
- The default IDS requirement map settings
- Manually creating IDS objects and rules
- Creating reusable objects

Upon completion of this chapter, you will have created:

- Reusable objects
- A scan global policy
- An attack, scan event, and TR TCP (condition, action, and policy)

Creating a new IDS policy

To create a new IDS policy:

1. From IBM Configuration Assistant for z/OS Communication Server, click **File** → **Open** → **Create a New Configuration**.
2. In the information panel, click **OK**.
3. Enter a file name for the new configuration and click **Save**.
4. In the Welcome panel, select **Intrusion Detection Services (IDS)**, as shown in Figure 6-4. Click **OK**.

> **Note:** IBM Configuration Assistant for z/OS Communication Server help is available via the Help menu option. If detailed information is needed for a particular field, click the **?** button and then click the specific panel or specific field in the panel in which you want to get help.

Figure 6-4 Welcome panel: Choosing to create an IDS configuration file

This brings you to the basic panel for configuring IDS policy, which is shown in Figure 6-5.

Figure 6-5 IBM Configuration Assistant for z/OS Communication Server first panel when choosing IDS

Adding a z/OS image to the policy

To add a z/OS image to the policy:

1. Click **IDS** → **Work with z/OS Images** from the Configuration Assistant Navigation Tree. Click **Add a New z/OS Image**.
2. In the New z/OS Image: Information panel, enter the name of the z/OS image and a description, as shown in Figure 6-6. Click **OK**. Then click **Yes** to proceed to the next step.

Figure 6-6 New z/OS Image: Information panel: Adding SC30 z/OS image

Adding a TCP/IP stack under the z/OS image

In the New TCP/IP Stack: Name panel, enter the TCP/IP stack name and definition, as shown in Figure 6-7. Click **OK**.

Figure 6-7 Adding a TCP/IP stack to the SC30 z/OS image

The default IDS requirement map settings

In the Proceed to the next step? panel, you have three options:

- Accept the default requirement map settings
- Get a tutorial about requirement maps
- Create a new requirement map with a wizard

We chose to use the default requirement map settings as a start. To see the default settings, click **View Details**.

Select **Accept the default Requirement Map settings**, as shown in Figure 6-8. Click **OK**.

Figure 6-8 Proceed to the next step? panel: Choosing the default settings

This should bring you to the panel shown in Figure 6-9.

Figure 6-9 z/OS Image Settings panel: After defining the default settings for IDS policy

Manually creating IDS objects and rules

We have now arrived at the most critical task: Defining the IDS policy rules. This task is typically done iteratively until the final policy rules are defined.

- You are required to establish one condition set and one action set in at least one policy rule.
- You may optionally specify that those rules apply only during validity periods.

Use this section to specify IDS policy rules, which can include condition sets, actions, policy keyword sets, or validity periods. Only one policy rule and associated actions can be applied to a particular packet.

Use the following steps to create an IDS policy rule:

1. Create reusable objects that will be used in your action and condition sets.
2. Create the actions and conditions based on those reuseable objects.
3. Build your policy rule from the available condition and action sets.

The following sections walk you through this process.

When you finish specifying IDS policy rules, click **IDS** → **Work with z/OS images** → **Image - *image_name*** → **Stack - *stack_name*** from the Configuration Assistant Navigation Tree to store the policy information into z/OS image.

Creating reusable objects

The reuseable objects are designed to be incorporated into multiple policies, conditions, or actions, depending on the type of object.

There are two types of reusable objects:

Traffic Descriptors Reusable objects which describe properties of network traffic such as protocols and ports.

Requirement Maps Reusable objects which are entire sets of intrusion detection requirements. Contains by default only the default requirement map settings object.

Figure 6-10 illustrates some of the traffic descriptors that are available in the Work with Reusable Objects → Traffic Descriptors panel.

Figure 6-10 Traffic Descriptors for IDS policy

Creating a new IDS policy

The next step is to configure an attack protection.

1. Select **IDS** → **Work with Reusable Objects** → **Requirement Maps** from the Configuration Assistant Navigation Tree.
2. In the Configuration Assistant - Requirement Maps panel, click **Add**.
3. In the Add Requirement Map: Name panel, enter a name and description, as shown in Figure 6-11. Click **Next**.

Figure 6-11 Add Requirement Map: Name - Adding an attack protection rule

Attacks

An attack can be a single packet designed to crash or hang a system, or multiple packets designed to consume a limited resource causing a network, system, or application to be unavailable to its intended users (a denial of service). The IDS attack policy lets you turn on attack detection for one or more categories of attacks independently of each other. In general, the types of actions that you can specify for an attack policy are event logging, statistics gathering, packet tracing, and discarding of attack packets.

In the Add Requirement Map: Attacks panel, verify that **Enable attack protection** check box is selected.

There are eight attack types that you can enable a protection from them: Flood, perpetual echo, unwanted IP protocols, unwanted IP options, ICMP redirect, malformed packet, outbound raw, and IP fragment. For only five of them (flood, perpetual echo, unwanted IP protocols, unwanted IP options, and outbound raw) you may modify and change their specific values.

We chose to leave the default configuration that you get with the IBM Configuration Assistant for z/OS Communication Server, as shown in Figure 6-12. Click **Next**.

Figure 6-12 Add Requirement Map: Attacks - Configuring attack protection

For information about the attack types, click the **?** button and then place the cursor in this field and click it.

Scans

You can specify sets of global scan detection parameters (threshold and interval for fast and slow scans). These attributes apply to all scan events. If you configure a certain category of scan events, the action will be triggered if the number of those events received from one IP address exceeds the slow scan threshold during the slow scan interval. Similarly, if you configure a certain category of scan event, the action will be triggered if the number of those events received from one IP address exceeds the fast scan threshold during the fast scan interval. The slow scan threshold must be greater than the fast scan threshold. The slow scan interval must be greater than the fast scan interval.

1. In the Add Requirement Map: Scans panel, select **Enable scan**.

   Notice that there are default values for the Fast Scan Interval, Fast Scan Threshold, Slow Scan Interval, and Slow Scan Threshold fields. You can see the default values by clicking the **Modify Fast and Slow Scan Settings** button and then clicking **OK**. We accept the default values.

Scan events come from the following categories:

- ICMP scans: ICMP requests (echo, information, time stamp, and subnet mask) are used to map network topology. Any request sent to a subnet base or broadcast address will be treated as very suspicious. Echo requests (PING) and time stamp requests are normal, unless they include the Record Route or Record Timestamp option, in which case they are possibly suspicious.
- TCP port scans: Because TCP is a stateful protocol, many different events may be classified as normal, suspicious, or highly suspicious. For more details, see the section "Scan policies" of *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.
- UDP port scans: A datagram received for a restricted port is very suspicious; one received for an unreserved but unbound port is possibly suspicious; and one received for a bound port is normal.

The individual packets used in a scan can be categorized as normal, possibly suspicious, or very suspicious. To control the performance impact and analysis load of scan monitoring, you can adjust your interest level in potential scan events. If you set the sensitivity level to:

- *High*: Normal, possibly suspicious, and very suspicious events will be counted.
- *Medium*: Possibly suspicious and very suspicious events will be counted.
- *Low*: Only very suspicious events will be counted.
- *None*: No events will be counted.

For more information about the sensitivity, click the **?** button and then click the **Sensitivity** field.

We accept the default configuration of enabled scans, as shown in Figure 6-13.

Note: There is an importance to the order of the enabled scans list. Packets are checked against the rules in the order they appear in the table.

Click **Default Report Settings for Scans**.

Figure 6-13 Add Requirement Map: Scans panel - Choosing the default enabled scans

2. In Report Types panel, indicate where to report IDS events. We selected **System console** and **SYSLOGD**, as shown in Figure 6-14. Click **Modify Details**.

Figure 6-14 Report Types panel - Indicating where to report IDS scan events

3. When you are configuring your IDS policy, you should select a high log level. We selected **6 - info**, as shown in Figure 6-15.

Figure 6-15 Report Events Details - SYSLOGD panel - Selecting a log level

In Report Events Details panel, click **OK**.

4. In Report Types panel, click **OK**.
5. In Add Requirement Map: Scans panel, click **Next**.

Traffic Regulation

Traffic Regulation (TR) policies are used to limit memory resource consumption and queue delay during peak loads. TR policies for TCP ports can limit the total number of connections an application has active at one time. This can be used to limit the number of address spaces created by forking applications such as FTPD and otelnetd. A fair share algorithm is also provided based on the percentage of remaining available connections held by a source IP address. IDS policies for UDP ports specify a queue length. Longer queues let applications with higher processing rate capacity absorb higher bursts of traffic. Shorter queues let applications with lower processing rate capacity reduce the queue delay time of packets that they accept.

1. In the Add Requirement Maps: Traffic Regulation, select **Enable traffic regulation** check box.

   From the Traffic Descriptors, select **All_Well-Known_TCP** and click **Enable** as shown in Figure 6-16.

Figure 6-16 Add Requirement Maps: Traffic Regulation panel - Adding All_Well-Known_TCP

2. In the Traffic Regulation Details panel, we accept the default values, as shown in Figure 6-17. However, you should consider the following in your environment:
   - For TCP, cap the number of connections, or the number of connections any one user can have with an application, for example:
     - Limit by total connections: Limit the size of the total connection pool for IDS TCP traffic regulation functions.
     - Limit by percentage: Limit the percentage of the total connections that can be used by a single host.
     - Limit by socket or by all sockets: Determine whether the limits should be applied to each socket, or to the aggregate of all sockets for each port. Note that you cannot specify local host addresses for the traffic regulation rule if you choose to limit connections by the aggregate of all sockets for the port.
   - For UDP, cap the number of queued packets, for example:
     - Select one of a number of abstract queue sizes that map to internally defined limits. These sizes are generally defined for each application as a function of response time and are subject to change over time.

   Click **OK** and then click **Finish**.

Figure 6-17 Traffic Regulation Details panel

Scan remote exclusions and local addresses

Setting scan remote exclusions and local addresses is optional. A scan exclusion consists of one or more scan exclusion range attributes that specify known legitimate scanners. To reduce false positives (that is, undesirable reports of scans by legitimate scanners), you can specify source IP addresses, a subnet mask length, and source port numbers of sources that you trust to be excluded from scan detection.

To do this, perform the following steps:

1. Select **IDS** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name*** from the Configuration Assistant Navigation Tree.
2. From the Configuration Assistant - Requirement Maps panel, select **IDS_Policy** from the requirement map list, as shown in Figure 6-18. Click **Set Addresses**.

Figure 6-18 Configuration Assistant - Requirement Maps for stack TCPIPA

3. In the Advanced Stack Settings panel, select the traffic descriptor for which you want to set remote exclusions and local addresses and click **Set Remote Exclusions and Local Addresses**.
4. In the Stack Level Scan Settings panel, select **Exclude the following remote addresses and ports** and click **Add**.
5. In the Scan Exclusion Remote Details panel, enter the remote exclusion address and port. Click **OK**.

   We did not add any exclusion, therefore we clicked **Cancel**.
6. If you want to set local addresses for which the rule will apply, select the **Local Addresses** tab from Stack Level Scan Settings panel, and then select **Rule applies to only the following specified local addresses**. Click **Add** and enter the local IP address. Click **OK** to save the changes. We clicked **Cancel**, because we do not want to set local addresses.
7. In Stack Level Scan Settings panel, click **OK**.
8. In Advanced Stack Settings panel, click **OK**.

6.3.5 Installing the IDS policy

After you finish configuring the IDS policy, perform these steps:

1. Right-click **IDS** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name*** from the Configuration Assistant Navigation Tree.

   From the drop-down menu, select **Install Configuration FIles**, as shown in Figure 6-19.

Note: You should click the **Health Check** button from the Connectivity Rules tab before you install the configuration file to the z/OS image. The health check reports on errors in your policy and we found it helpful.

Figure 6-19 Configuration Assistant - Requirement Maps panel - Installing configuration files

2. In the Installation - Stack = “stack_name” panel, select the ***stack_name* - IDS: Policy Agent Stack Configuration** from the list, as shown in Figure 6-20. Click **FTP**.

Figure 6-20 Installation - Stack panel - Installing the IDS configuration file

3. In the FTP Configuration File panel, enter the host name, the FTP port (usually 21), a user ID and its password. In addition, enter the file name and location of the IDS policy. This file name and location should be the same file name and location that are mentioned in the

TCP/IP stack configuration file, which is used by Policy Agent, as shown in Figure 6-21. Click **Send**.

Figure 6-21 FTP Configuration File - Sending the IDS policy to SC30 z/OS image

4. After you get the message "`The FTP transfer was successful`", refresh the Policy Agent by running the console command:

   ```
   MODIFY PAGENT,REFRESH
   ```

   You should get the following log messages:

   ```
   EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
   EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : IDS
   ```

6.3.6 Checking that things are working

In order to check that your policy is working, use tools which create the specific attacks that the policy is supposed to provide protection against.

For example, in order to check that the scan policy works, you can download a tool from the Web, which scans ports and then follow these steps:

1. Run a ports scan on the z/OS image IP address.
2. Run NETSTAT IDS command to get an IDS summary of scan, attack, and traffic regulation detected.

Note: The user who invokes the NETSTAT IDS commands needs to be permitted for READ access to the resource name: EZB.NETSTAT.mvsname.tcpprocname.IDS. For more information about the NETSTAT command, see *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781.

In our test environment, we used a free utility that scans ports on a specific address. The console log messages shown in Example 6-3 appeared on the console.

Example 6-3 Console messages after running a fast ports scan from client on z/OS IP address

```
EZZ8762I EVENT TYPE: FAST SCAN DETECTED
EZZ8763I CORRELATOR 9 - PROBEID 0300FFF1
EZZ8764I SOURCE IP ADDRESS 192.168.1.254 - PORT 0
EZZ8766I IDS RULE ScanGlobal
EZZ8767I IDS ACTION ScanGlobalAction
```

We then issued the NETSTAT IDS command and the output can be seen in Example 6-4. We copied only the part of the output that relates to scan detection.

Example 6-4 The output of NETSTAT IDS command after running the ports scan

```
MVS TCP/IP NETSTAT CS V1R8       TCPIP Name: TCPIPA          19:51:22
Intrusion Detection Services Summary:
Scan Detection:
  GlobRuleName: ScanGlobal
  IcmpRuleName: ICMP~1
  TotDetected:  1           DetCurrPlc: 1
  DetCurrInt:   0           Interval:   30
  SrcIPsTrkd:   1           StrgLev:    00000M
```

6.3.7 Additional information

In this section, we provide additional information, including a summary of common mistakes and logging.

NetView and z/OS IDS

NetView z/OS V5R1, PTF UA11043, provides management support for z/OS Communications Server IDS. It provides the ability to:

- Trap IDS messages from the system console or syslogd and take predefined actions based on IDS event type.
- Route IDS messages to designated NetView consoles.
- Provide e-mail notifications to security administrators (including running trmdstat and attaching the output to the e-mail).
- Issue predefined commands.

IDSAUTO is a set of NetView REXX clists and automation table entries that automates IDS messsages and performs notifications and reporting via e-mails. These clists can be downloaded from the following Web site:

http://www.ibm.com/support/docview.wss?uid=swg24001743

Tivoli Risk Manager and z/OS IDS

You are able to send TEC events to IBM Tivoli® Risk Manager (V4R1 or later) for enterprise-wide correlation and analysis of intrusion events.

The format file provided by z/OS Communications Server to convert syslog messages to TEC events is available at:

http://www-1.ibm.com/support/docview.wss?uid=swg24006973

7

Quality of Service

Quality of Service (QoS) refers to a set of networking technologies intended to ensure satisfactory end-to-end application performance in the presence of network congestion, essentially by giving time-sensitive applications (such as interactive transaction processing) priority in the network over less time-sensitive applications (such as print or file transfer).

This QoS discussion could have been placed in the IBM Redbook *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341, as performance issues are frequently perceived as availability issues. However, we chose to instead include the QoS discussion in this book (*Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-based Network Security*, SG24-7342), because QoS is a key part of policy-based networking and is implemented in z/OS through the Policy Agent (PAGENT), discussed in Chapter 1, “Policy Agent” on page 3.

This chapter discusses the topics listed in Table 7-1.

Table 7-1 Topics discussed in this chapter

Section	Topic
7.1, “QoS definition” on page 280	Discusses the basic concepts of QoS.
7.2, “Why QoS is important” on page 286	Discusses key characteristics of QoS and why it may be important in your environment.
7.4, “Using the IBM Configuration Assistant for z/OS Communication Server” on page 286	Presents configuration examples and problem determination suggestions.

© Copyright IBM Corp. 2007. All rights reserved.

7.1 QoS definition

The terminology used in the networking industry can be confusing, partly because we have historically shown a tendency to reuse old terms in new and different ways. Quality of Service (QoS) is such a term.

The term QoS originally came out of the work on Asynchronous Transfer Mode (ATM) technology—specifically, a parameter on the ATM User-to-Network Interface (UNI). If an ATM network user signals across the UNI for a specific QoS, the network is supposed to determine if it can support the requested QoS and grant or reject the connection accordingly (similar to connecting a voice telephone call). In theory, if the network grants the connection, it must *guarantee* the QoS for the connection.

ATM never caught on as an end-to-end networking technology (partially due to its complexity) and, in the case of Ethernet and TCP/IP, there is no UNI; so what does QoS really mean today?

The ATM QoS concept of a *contract* between a network user and the network, guaranteeing certain network throughput (and delay and delay variability), was implemented in TCP/IP as *Integrated Services*. Integrated Services is supported by the Resource Reservation Protocol (RSVP), which is used to allow an application to request (or *signal*) the network to reserve a certain amount of bandwidth with particular QoS criteria. RSVP is defined in Internet Engineering Task Force (IETF) Internet standard RFC 2205 and can be used to provide something similar to a dedicated circuit over an IP network. Integrated Services and RSVP can provide an essential capability to support certain network applications such as high-quality, interactive, voice, or video; however, due to its complexity and the fact that adequate performance can be achieved for most applications more simply by just using prioritization, RSVP has not been widely implemented.

Short of Integrated Services, QoS may be thought of as “network prioritization done right.” Historically, organizations individually configured each router in the network to inspect and prioritize each message. As the complexity of networks and applications have increased, however, it has become increasingly difficult to individually configure each network component yet still ensure that the overall network implementation matches the desired business policies. Consequently, organizations are now developing *enterprise-wide* QoS policies (encompassing the network and advanced servers such as zSeries mainframes). The *Differentiated Services* form of QoS involves associating individual packets or flows with a particular *class of service* (not to be confused with SNA class of service) and having each node along the network path handle packets in a cooperative manner, according to a common set of rules, resulting in end-to-end service classes. Additionally, policy-based networking has emerged as a standards-based approach for defining QoS policies in one place and applying them uniformly across the entire IT environment.

7.1.1 Differentiated Services

Differentiated Services (DiffServ) was developed to allow a network to support multiple service classes without the need to maintain the state of each traffic flow along the path or to perform signaling between nodes (illustrated in Figure 7-1 on page 281). It can, therefore, scale to support the traffic seen in today’s global networks. The network domain manager or administrator defines aggregate traffic service classes (for example, premium, gold, silver, and bronze). DiffServ is, therefore, less complex than Integrated Services. It is less network intensive and is appropriate for networks of networks even where portions of the network are outside the control of the network domain manager.

Figure 7-1 DiffServ end-to-end architecture

DiffServ is described in IETF RFC 2474, RFC 2475, RFC 2597, and RFC 2598. DiffServ is meant to handle traffic aggregates. This means that traffic is classified according to the application requirements relative to other application traffic. Each node then handles the traffic using internal mechanisms to control bandwidth, delay, jitter, and packet loss. Through the use of standard per-hop-behaviors (PHBs), packets receive the proper handling and the result is end-to-end QoS.

For true end-to-end QoS, each administrative domain must implement cooperative policies and PHBs. Packets entering a DiffServ domain can be metered, marked, shaped, or policed to implement traffic policies as defined by the administrative authority. This is handled by the DiffServ traffic conditioner block (TCB) function. DiffServ boundary nodes typically perform traffic conditioning. A traffic conditioner typically classifies the incoming packets into predefined aggregate classes, meters them to determine compliance to traffic parameters, marks them appropriately by writing or re-writing the DSCP, and finally shapes the traffic as it leaves the node.

The DS field

To distinguish the data packets from different applications in DS-capable network devices, the IP packets are modified in a specific field. A small bit pattern, called the *DS field*, in each IP packet is used to mark the packets that receive a particular forwarding treatment at each network node. The DS field uses the space of the former TOS octet in the IPv4 IP header and the traffic class octet in the IPv6 header. All network traffic inside of a domain receives a service that depends on the traffic class that is specified in the DS field.

The DS field uses six bits to determine the Differentiated Services Code Point (DSCP) as defined in RFC 2474 and RFC 2475. This code point will be used by each node in the net to select the PHB. A two-bit currently unused (CU) field is reserved. The values of the CU bits are ignored by DS-compliant nodes when PHB is used for received packets. Figure 7-2 shows the structure of the defined DS field.

Figure 7-2 DS field

In the event that some nodes in a network recognize only the IP precedence bits, standard DSCP PHBs are constructed in such a way that they remain compatible with IP precedence. For example, the DSCP values can be used such that the values for IP precedence relate to the classes, as shown in Table 7-2.

Table 7-2 Relationship between IP precedence and DSCP

RFC 791 precedence		RFC 2474, RFC 2475 DiffServ	
Network Control	111 (7)	Preserved	111000
Internetwork Control	110 (6)	Preserved	110000
CRITIC/ECP	101 (5)	Express Forwarding	101xxx
Flash Override	100 (4)	Class 4	100xxx
Flash	011 (3)	Class 3	011xxx
Immediate	010 (2)	Class 2	010xxx
Priority	001 (1)	Class 1	001xxx
Routine	000 (0)	Best Effort	000000

7.1.2 QoS with z/OS Communications Server

In the z/OS Communications Server environment, support for Integrated Services is provided by the RSVP Agent. The RSVP Agent queries the Policy Agent for relevant information and communicates with the network to request the desired QoS on behalf of the application.

Differentiated Services is supported by the PAGENT. PAGENT gets policy definitions from a local configuration file or a Lightweight Directory Access Protocol (LDAP) server. PAGENT then installs the policies in the z/OS Communications Server stacks as desired.

Figure 7-3 shows the relationship between the various z/OS QoS components. Tasks or daemons such as PAGENT and RSVPD work together and with the TCP/IP protocol stack to classify and mark packets for QoS. Data collection points are also available for performance management.

Figure 7-3 z/OS CS QoS components

Note: Without a cooperative framework of host-based components and QoS mechanisms within the network (*and the necessary interorganzational coordination*), it is impossible to establish and implement end-to-end service levels. It could well happen that you set up the policies and go through the effort to set the DS field in messages only to have the network overwrite your settings with their own.

7.1.3 PAGENT QoS policies

We suggest that when you first implement QoS policies you start with a small number of critical applications or traffic types. Then, as you develop more knowledge of the traffic patterns and interactions, continue to apply a set of service classes to applications or traffic streams as needed.

The PAGENT supports the following QoS policies:

- Differentiated Services (DS) policies
- Integrated Services (RSVP) policies
- Sysplex Distributor (SD) policies

Note: Sysplex Distributor policies are discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341.

Policy conditions consist of a variety of selection criteria that act as traffic filters. Traffic can be filtered based on source/destination IP addresses, source/destination ports, protocol, inbound/outbound interfaces, application name, application-specific data, or application priority. Only packets that match the filter criteria are selected to receive the accompanying action. Policy rules can refer to several policy actions, but only one policy action is executed per policy scope. A given policy action may be referred to by several policy rules.

Differentiated Services policies

Policies to be implemented can be configured via the Policy Agent configuration file, in an LDAP server, or both. Once read, the policies are combined into a single list. Policy rules and actions map subsets of outbound traffic to various QoS classes and can be used to create end-to-end Differentiated Services.

Setting DSCP using the Policy Agent

PAGENT policies are defined by rules and actions. The rules consist of a variety of selection criteria to provide a *match condition*. Matching the *rule* then forces the *action*. One particularly important action is the setting of the DS field. Outbound traffic can be marked with the desired Differentiated Services Control Point (DSCP) value. This marking will then be interrogated by the network and the appropriate per-hop behavior (PHB) applied as the packet traverses the network.

Integrated Services (RSVP) policies

Given the narrow applicability of Integrated Services and RSVP, they are not covered in this book.

7.1.4 Configuring QoS in the z/OS Communications Server

The two components responsible for QoS within the z/OS Communications Server are the Policy Agent and the RSVP Agent. In this section, we provide an overview of the configuration steps necessary to use the z/OS CS Policy Agent for QoS. PAGENT runs in the z/OS environment and reads policy definitions from a local configuration file or a central repository that uses the LDAP.

There are two graphical user interfaces (GUI) for configuring the QoS policies:

- zQoS Manager for configuring QoS GUI: This is the GUI which was used in older versions of Communications Server and is using LDAP server.
- IBM Configuration Assistant for z/OS Communication Server GUI: This is the new GUI which generates a flat configuration file and does not need LDAP server.

In this chapter, we show how to use the IBM Configuration Assistant for z/OS Communication Server.

After the policies are updated, PAGENT installs policies in one or more z/OS CS stacks, replacing existing policies or updating them as necessary.

Note: The IBM Configuration Assistant for z/OS Communication Server GUI is intended to replace the zQoS Manager for configuring QoS.

Policies

Policies consist of several related objects. The main object is the *policy rule.* A policy rule object refers to one or more *policy condition*, *policy action*, or *policy time period condition objects*, and also contains information about how these objects are to be used. Policy time period objects are used to determine when a given policy rule is active. Active policy objects are related in a way that is analogous to an IF statement in a program. For example:

```
IF condition THEN action
```

In other words, when the set of conditions referred to by a policy rule are TRUE, then the policy actions associated with the policy rule are executed.

Differentiated Services rule

The most common QoS deployment uses rules to map outbound traffic from particular applications into subclasses. Example 7-1 illustrates this type of policy. The goal of this priority control policy is to map a subset of the traffic outbound from an FTP server.

Example 7-1 Sample Priority_Control rule for FTP

```
policyRule Priority_Control~9
{
   PolicyRulePriority        64920
   DestinationAddressRange   192.168.1.254 1
   SourcePortRange           21 1
   DestinationPortRange      1024-65535
   ProtocolNumberRange       6
   PolicyActionReference     action~3
}

policyRule Priority_Control~10
{
   PolicyRulePriority        64910
   DestinationAddressRange   192.168.1.254 1
   SourcePortRange           20 1
   DestinationPortRange      1024-65535
   ProtocolNumberRange       6
   PolicyActionReference     action~3
}
PolicyAction action~3
{
   PolicyScope               DataTraffic
   OutgoingTOS               01000000 2
}
```

This policy is identified as a Differentiated Services policy by the PolicyScope DataTraffic attribute on the PolicyAction statement, as well as the use of several DS-only attributes.

The following statements apply to Example 7-1:

- 1 The policy rules selects traffic originated by port 21 for TCP (FTP outbound data connection uses port 20) from the source address 192.168.1.254.
- 2 The policy action specifies that the TOS byte be set to '10000000' for traffic that conforms to this policy.

7.1.5 For additional information

For additional information, refer to:

- *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS V1R8.0 Communications Server: IP Configuration Reference*, SC31-8776

7.2 Why QoS is important

Over the past decade, the amount of available network bandwidth has increased almost exponentially while bandwidth costs have declined almost as dramatically. Yet, still, bandwidth is not free (nor is it equally available in all locations) and, consequently, organizations must strive to provide required application performance in the face of constrained network capacity. The best way to do so is to understand the service levels required for each type of traffic in the network and prioritize that traffic accordingly. QoS, along with policy-based networking, provides the facilities to do that prioritization consistently, and end-to-end, across the entire IT environment.

7.3 Install the Policy Agent

The Policy Agent reads the configuration files that contain the QoS policy configuration statements, checks them for errors, and installs them into the TCP/IP stack. Setting up the PAGENT is described in Chapter 1, “Policy Agent” on page 3.

Note: You need superuser authority to start PAGENT, and the PAGENT executable modules must be in an APF-authorized library.

After setting it up, you need to define the QoSConfig statement to specify the path of the policy file that contains stack-specific QoS policy statements to PAGENT. Example 7-2 shows the QoS statements in the TCP/IP stack configuration file used by Policy Agent.

Example 7-2 The pagent /etc/sc30.tcpipa_image.conf file with QoS configured

```
# This is a file that contains statements for TCP/IP stack TCPIPA in z/OS image SC30
# This file is used by Policy Agent.
#
# QoSConfig Statements
QoSConfig //'TCPIP.SC30.POLICIES(QOSA)'
```

7.4 Using the IBM Configuration Assistant for z/OS Communication Server

The IBM Configuration Assistant for z/OS Communication Server is discussed in the following sections:

- IBM Configuration Assistant for z/OS Communication Server
- Download and installation
- Using the GUI

IBM Configuration Assistant for z/OS Communication Server

The IBM Configuration Assistant for z/OS Communication Server enables centralized configuration of Quality of Service policies for z/OS. The IBM Configuration Assistant for z/OS Communication Server helps a network administrator produce a flat QoS policy file.

The IBM Configuration Assistant for z/OS Communication Server is a tool for network administrators. Therefore, before you begin you should read the chapter on policy-based networking in *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.

Be familiar with your particular environment so that you can make decisions about what events are to be detected under what circumstances and the appropriate actions to take.

Download and installation

The download and installation instructions are written for Windows. The information and executable in the following sections are also located at:

http://www.ibm.com/software/network/commserver/zos/support/

Using the GUI

This section is intended to help the network administrator manage and understand the GUI provided. The first-level directories are:

- z/OS host information
- QoS policy rules

Open the IBM Configuration Assistant for z/OS Communication Server.

The first window displayed when starting the IBM Configuration Assistant for z/OS Communication Server is shown in Figure 7-4.

Note: IBM Configuration Assistant for z/OS Communication Server Help is available via the Help button. If detailed information is needed for a particular field, click the **?** button and then click the desired field.

Figure 7-4 IBM Configuration Assistant for z/OS Communication Server Manager

In the Welcome panel, select **Quality of Service (QoS)**, as shown in Figure 7-5. Click **OK**.

Figure 7-5 Welcome panel: Selecting Quality of Service policy

7.4.1 z/OS host information

This section provides information that will be included in PAGENT's configuration file on the z/OS host. The information that you provide will depend upon your policies. The factors to consider are:

- Should policies be applied to Sysplex Distributor?
- Are you using OSA Express cards in QDIO mode?
- Are you running with multiple TCP/IP stacks on this LPAR? And if so, to which instances of TCP/IP should your policies be applied?

Adding z/OS image to the policy

To add z/OS image to the policy:

1. From the Configuration Assistant Navigation Tree, right-click **QoS** → **Work with z/OS Images** and click **Add new z/OS Image** from the menu, as shown in Figure 7-6.

Figure 7-6 Work with z/OS Images panel: Adding a new z/OS image to the QoS policy

2. In the New z/OS Image: Information panel, enter the z/OS image name and description. We added SC30, as shown in Figure 7-7. Click **OK**.

Figure 7-7 New z/OS Image: Information panel: Defining SC30 z/OS image

3. In the Proceed to the next step? panel, click **Yes**. See Figure 7-8.

Figure 7-8 Proceed to the next step? panel

Adding TCP/IP stack to the policy

To add TCP/IP stack to the policy:

1. In the New TCP/IP Stack Wizard: Welcome panel click **Next**.
2. In the New TCP/IP Stack: Name panel, enter the name of the TCP/IP stack and a description, as shown in Figure 7-9. Click **Next**.

Figure 7-9 New TCP/IP Stack: Name - Adding TCPIPA stack

3. In the New TCP/IP Stack Wizard: Sysplex Distributor panel, indicate if the stack will use QoS for Sysplex Distributor. In our configuration, we chose not to use QoS for Sysplex Distributor. Select **This Stack will NOT use QoS for Sysplex Distributor**, as shown in Figure 7-10. Click **Next** and then click **Finish**.

Figure 7-10 New TCP/IP Stack Wizard: Sysplex Distributor panel

4. In the Proceed to the next step? panel, click **No**.

QoS performance logging

Performance logging is used to collect QoS performance monitoring data. The performance data can be collected on a policy rule or action or on both rules and actions. Policy Rules map to the GUI's connectivity rules and traffic descriptors. Policy actions map to the GUI's priority levels and traffic shaping levels. The collected data can also be logged to a specified performance log file for offline collection and monitoring by a user application, or can be accessed in near real time using the Policy API (PAPI).

From the Configuration Assistant Navigation Tree, select **QoS** → **Work with z/OS Images** → **Image - *image_name*** → **Stack - *stack_name***.

In the Configuration Assistant - TCP/IP Stack Settings panel, select the **QoS Performance Logging** tab and select **Enable QoS Performance Logging** check box. In addition, enter a file name for the performance log file, as shown in Figure 7-11. Click **Apply Changes**.

Figure 7-11 QoS Performance Logging tab: Enabling performance logging

VLAN/QDIO priority tagging

This panel is used to set specific parameters for Virtual LAN (VLAN) and device priority tagging.

- **VLAN priority tagging**: This function allows you to correlate the TOS byte in outgoing IP packets to a priority in LAN frames according to IEEE 802.1Q specifications. Specifically, it sets the packet priorities for switches.
- **Device priority tagging**: This function allows you to correlate the TOS byte in outgoing IP packets to a priority for interfaces that use OSA-Express configured in QDIO mode.

1. From the Configuration Assistant Navigation Tree, select **QoS** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name***.
2. In the Configuration Assistant - TCP/IP Stack Settings panel, select the **VLAN/QDIO Priority Tagging** tab and select **Enable priority tagging** check box.

   There are two options for the priority tagging:

   – Enabling the priority logging for all interfaces
   – Enabling the priority logging for specific interfaces

   We chose the first option. Verify that **Enable for all interfaces** is selected. If you want to specify specific interfaces, select **Enable for only the following interfaces** and click **Add**.
3. In VLAN Tagging Settings for Single Interface panel enter the local interface and then click **OK**. This can be an IPv4 address or an interface name. If an interface name is specified, it must match a name specified on one of the following statements in the TCP/IP profile:

   – LINK statement for an IPv4 interface
   – INTERFACE statement for an IPv6 interfaces

4. In the Configuration Assistant - TCP/IP Stack Settings panel, click **OK**, as shown in Figure 7-12. Click **Apply Changes**.

Figure 7-12 TCP/IP Stack Settings panel: After enabling priority tagging for all interfaces

7.4.2 QoS policy rules

Specifying the QoS policy rules is the most critical task and typically will be done iteratively until the final policy rules are accepted:

- You are required to define the data endpoints and to use one requirement map in at least one policy rule.
- You may optionally specify that rules apply only during validity periods.

Use this section to specify QoS policy rules. When you have finished specifying QoS policy rules, right-click **QoS** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name*** and select **Install Configuration Files** from the menu.

Creating QoS policy rules

The Connectivity Rules tab under Configuration Assistant - TCP/IP Stack Settings panel (shown in Figure 7-13) enables you to create a policy rule by linking together policy requirement map object with data endpoints and validity periods (time periods when the policy condition will be active). You also have the option of identifying whether this particular rule will be used by the Sysplex Distributor for load distribution. However, before you get to this stage, you must create the QoS requirement maps and define their validity ranges.

Figure 7-13 Connectivity rules panel

QoS requirement maps

Like the Connectivity Rules panel, the Configuration Assistant - Requirement Maps panel (shown in Figure 7-14) links together sets of information. In this case, these are all actual conditions that you will want PAGENT to check for.

Figure 7-14 Configuration Assistant - Requirement Maps panel

Creating a requirement map object

When adding a new requirement map, each row of the Requirement Map table is a mapping between a traffic descriptor and an action. For QoS, the actions are a priority level and a traffic shaping level. Each mapping characterizes a level of service (QoS) for a type of traffic. Here we define what should happen to a packet if it matches a particular traffic descriptor.

- **Priority levels**: Objects which characterize different ways to prioritize network traffic. Network traffic is prioritized by specifying a specific value for the IPv4 Type-of-Service (TOS), the IPv6 Traffic Class value, or Differentiated Services Code Point (DSCP) field to be set in the outgoing IP packets.
- **Traffic shaping level**: Objects which characterize different ways to control the levels of traffic. Traffic shaping levels can control TCP connections, TCP throughput, and Token Bucket traffic policing.

To add a requirement map object:

1. From the Configuration Assistant Navigation Tree, select **QoS** → **Work with Reusable Objects** → **Requirements Maps**.
2. In the Configuration Assistant - Requirement Maps panel, click **Add**.
3. In the Requirement Map panel, enter a name and description. Select a traffic descriptor from the Objects list and click **Add**. For example, we chose **FTP-Client** and **FTP-Server**, as shown in Figure 7-15. We left the default priority level and traffic shaping level.

Figure 7-15 Requirement Map panel: Adding a requirement map for FTP

4. In the Requirement Map panel, you can click **Set Effective Times** to set the periods of time on which the requirement map should be active. After completing setting the effective times, in the Effective Times panel, click **OK**.
5. In the Requirement Map panel, click **OK**.

> **Note:** Be aware that these requirement map objects are reusable and as such may be included in multiple QoS connectivity rules.

QoS connectivity rules

Now that you have created your requirement map objects, the next step is to add connectivity rules. This is done through the Connectivity Rules panel. Here we define a rule for data endpoints and a specific requirement map object. We chose to use one of the built-in objects.

To add a connectivity rule:

1. From the Configuration Assistant Navigation Tree, select **QoS** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name***.
2. From the Configuration Assistant - TCP/IP Stack Settings panel, verify that you are working on **Connectivity Rules** tab and click **Add**.
3. In the Connectivity Rules: Data Endpoints panel, identify the data endpoints for the rule. For example, we defined the rule from **All IPv4 addresses** to 192.168.1.254, as shown in Figure 7-16. Click **Next**.

Figure 7-16 Connectivity Rule: Data Endpoints panel - Identifying the data endpoints

4. In the Requirement Map: Select Requirement Map panel, select the requirement map that you want to use in this rule. We chose one of the built-in requirement map objects, **Priority_Control**, as shown in Figure 7-17.

Figure 7-17 Connectivity Rule: Select Requirement Map - When the stack does not participate in sysplex

5. If the TCP/IP stack that you are configuring now is not participating in a sysplex, then click **Finish**, as shown in Figure 7-17. If the stack is configured to participate in a sysplex, click **Next**, as shown in Figure 7-18.

Figure 7-18 Connectivity Rule: Select Requirement Map - When the stack participates in a sysplex

6. This step is relevant only if the stack is participating in a sysplex.
   a. In the Connectivity Rule: QoS Sysplex Target Role panel, indicate if the rule is for a QoS sysplex target by selecting the **This rule is for a QoS sysplex target** check box. In addition, indicate if this rule should be a primary or secondary target of the Sysplex Distributor. If you select **This Stack will be a primary target for applications specified in this Connectivity Rule:**, click the **Add** button to add each local interface that should be a primary target. Click **Next**.
   b. In the Connectivity Rule: Additional Settings panel, you may click **Advanced** to modify the QoS sysplex distributor settings. After you finish modifying, click **OK**.
   c. In the Connectivity Rule: Additional Settings panel, click **Finish**.

Note: The actual setup of Sysplex Distributor for high availability and workload balancing is discussed in detail in Chapter 6, “Internal application workload balancing,” of *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341. In that book, however, a simple configuration flat file is used to define Sysplex Distributor policies to PAGENT. If you need to create more complex or dynamic Sysplex Distributor policies, you should use the IBM Configuration Assistant for z/OS Communication Server. Some of the following panels illustrate the use of IBM Configuration Assistant for z/OS Communication Server Manager for Sysplex Distributor policies.

Connectivity rules priorities

Policies consist of several related objects. The main object is the connectivity rule. A connectivity rule object refers to one requirement object, which refers to one or more traffic descriptors, priority levels, and traffic shaping levels. Validity periods determine when each requirement map object is active. Active requirement map objects are analogous to an IF statement in a program. For example:

```
IF condition THEN action
```

In other words, when the set of conditions referred to by a requirement map on a connectivity rule are TRUE, then the traffic shaping levels associated with the requirement map object are executed. Only one connectivity rule and associated actions can be applied to a particular packet. The prioritization of the policy can be seen when you add a connectivity rule and use the **Move Up** and **Move Down** buttons, as shown in Figure 7-19.

Figure 7-19 Configuration Assistant - TCP/IP Stack Settings panel: Move up button

The first connectivity with a true condition will be executed. Thus, the prioritization of connectivity rules must be evaluated prior to implementation. One can easily prioritize a connectivity rule by clicking a connectivity rule in the right pane and clicking the **Move Up** or **Move Down** buttons.

FTP the QoS policy to the z/OS image

When you finish defining the QoS policy, FTP it to the z/OS image.

1. From the Configuration Assistant Navigation Tree, right-click **QoS** → **Work with z/OS Images** → **Image - *image_name*** → **Stack - *stack_name*** and select **Install Configuration Files**, as shown in Figure 7-20 on page 300.

> **Note:** You should click the **Health Check** button from the Connectivity Rules tab before you install the configuration file to the z/OS image. The health check reports on errors in your policy and we found it helpful.

Figure 7-20 Configuration Assistant - TCP/IP Stack Settings panel: Installing the configuration files

2. If you are asked to apply changes in the Do you want to Apply Changes? panel, then click **Apply Changes**.
3. In the Installation - Stack = “stack_name” panel, select the configuration file from the list, as shown in Figure 7-21. Click **FTP**.

Figure 7-21 Installation - Stack - “TCPIPA” panel: Installing the QoS configuration files

4. In the FTP Configuration File panel, enter the host name, user ID, and its password. In addition, verify that the port number is 21 and enter the file name and location for the QoS configuration file, as shown in Figure 7-22. The name and location should be equal to the name and location that were defined in the QoSConfig statement at the TCP/IP configuration file for Policy Agent. Click **Send**.

Figure 7-22 FTP Configuration File panel: Installing the QoS configuration file

5. From the console, refresh the Policy Agent policies, by issuing the F PAGENT,REFRESH command. You should see the following messages on the log:

```
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : QOS
```

7.4.3 Problem determination

You can see the effect of defined QoS connectivity rules in the following ways:

- Check the log file of the Policy Agent to see if you got any error messages. In our configuration, the log file is: /tmp/pagent.sc30.log.
- Use the Network SLAPM2 Subagent to display service policy and mapped application information, as well as to manage and display Network SLAPM2 performance monitoring.
- Use the SLA Subagent to display service policy and mapped application information, as well as to manage and display SLA performance monitoring.
- Use the z/OS UNIX **pasearch**, z/OS UNIX **netstat**, and TSO NETSTAT commands as follows:
  - The NETSTAT SLAP (**netstat -j**) command shows performance metrics for active QoS policy rules.
  - The NETSTAT ALL or **netstat -A** command has additional information for each active connection that shows the QoS policy rule name if the connection maps to a QoS policy.
  - Issue **pasearch -q** to see all QoS policies that are active in Policy Agent.

Available management tools

There are also tools available that can help you manage your network QoS configuration and parameters including the z/OS Communications Server SNMP SLA Subagent and network device MIB variables and tools.

z/OS Communications Server SNMP SLA Subagent

The z/OS CS SLA Subagent allows network administrators to retrieve data and determine if the current set of SLA policy definitions is performing as needed or if adjustments need to be made. The SLA Subagent supports the Service Level Agreement Performance Monitor (SLAPM) MIB. Refer to RFC 2758 for more information about the SLAPM MIB.

Network MIB variables and tools

If you are using Cisco networking gear, the following MIBs may be helpful.

Cisco QoS MIB

The Cisco Class-Based QoS MIB provides you with the same statistics that the **`show policy interface`** command provides.

To measure packet loss through the network with regard to different classes of service, use Class-Based Queuing over Security Management Information Base (CBQoSMIB (available in 12.1(5) T), SAA/IPM, or QPM.

See also:

- CISCO-CLASS-BASED-QOS-MIB.my
- CISCO-CLASS-BASED-QOS-MIB-CAPABILITY.my

Cisco QoS Device Manager

Cisco QoS Device Manager (QDM) is a Web-based network management application that provides an easy-to-use graphical user interface for configuring and monitoring advanced IP-based QoS functionality in Cisco Systems routers.

QDM is intended for users who are configuring QoS functionality in their network for the first time. It is an easy-to-use management application to help you configure and monitor QoS features in the most critical router devices in your network. Using QDM, you can quickly and easily configure QoS functionality and immediately observe the effect that this QoS configuration has on the pattern of network traffic through the network.

Cisco QoS Policy Manager

QoS Policy Manager (QPM) is a QoS policy system that makes it easy to define traffic policies and automate multiple service levels across any network topology. The product enables network-wide, content-based Differentiated Services; centralized policy control for voice/video/data networks; automated QoS configuration and deployment; and campus-to-WAN policy control. By automating the process of translating application performance requirements into QoS policy, QPM helps ensure reliable performance for Internet business applications and voice traffic that contends with noncritical traffic. Using QPM, a network administrator can quickly construct rules-based QoS policies that identify and partition application traffic into multiple levels of service.

Part 2

SAF-based security

In this part, we explain how you can use the z/OS Security Access Facility (SAF) to protect your network and communications. SAF is the high-level infrastructure that allows you to plug in any commercially available security product. This book specifically focuses on the IBM Resource Access Control Facility (RACF) element of the z/OS Security Server.

Attention: Many of the tasks, examples, and references in this section assume that you are using the z/OS Security Server (RACF). References to RACF apply to any other SAF-compliant security products that contain the required support. If you are using another security product, read the documentation for that product for instructions about task performance.

© Copyright IBM Corp. 2007. All rights reserved.

8

RACF demystified

In this chapter, we explain how you can use the IBM Resource Access Control Facility (RACF), a component of z/OS System Authorization Facility (SAF), to protect your network and communications. SAF is the high-level infrastructure that allows you to plug in to any commercially available security product.

Note: The tasks, examples, and references in this chapter are based upon the z/OS Security Server (RACF). The basic concepts are similar to other commercially available security products.

Table 8-1 Topics discussed in this chapter

Section	Topic
8.1, “Basic concepts” on page 306	We try to demystify RACF for you by explaining the basic concepts in simple terms.
8.2, “How to protect your network resources” on page 308	We show you how the TCP/IP resources such as the stack, the ports, the commands, and so on are protected by RACF.
8.3, “How to protect your programs” on page 309	This section explains concepts of program protection.
8.4, “How to associate a user ID with a started task” on page 310	We explain how RACF correlates user IDs to STCs.
8.6, “RACF multilevel security for network resources” on page 311	We explain the basic concepts of MLS.
8.7, “Digital certificates in RACF” on page 312	This section explains RACF support for keys and certificate management.

© Copyright IBM Corp. 2007. All rights reserved.

8.1 Basic concepts

RACF has evolved over more than 30 years to provide protection for a variety of resources, features, facilities, programs, and commands on the z/OS platform. Because of its vast array of commands and numerous methods of protection, you could quickly become confused by RACF. In this chapter, we try to demystify RACF for you by explaining the basic concepts.

The RACF concept is very simple: It keeps a record of all the resources that it protects in the RACF database. It can, for example, set permissions for file patterns even for files that do not yet exist. Those permissions are then used should the file (or other object) be created at a later time. In other words, RACF establishes security policies rather than just permission records.

RACF initially identifies and authenticates users via user ID and password when they log on to the system. When a user tries to access a resource, RACF checks its database and, based on the information that it finds in the database, it either allows or denies the access request. It displays an ICH408I message if the access is denied.

To understand the basic concepts, let us look closely at one of these ICH408I access denial messages (shown in Example 8-1) to see what RACF is telling us.

Example 8-1 ICH408I message

```
ICH408I USER(UTSM) GROUP(MTSM) NAME(TSOMON STC-USERID)
 EZB.PORTACCESS.SX00.TCP2.SAPSYS CL (SERVAUTH)
 INSUFFICIENT ACCESS AUTHORITY FROM EZB.PORTACCESS.*.*.SAPSYS (G)
```

This message means that the user is not authorized to access the TCP/IP port.

- The user is UTSM.
- The user belongs to RACF group MTSM. RACF keeps users with similar security access requirements in groups so that any access changes can be done just to the group profile (record) rather than to each individual user's profile.
- The name recorded in the RACF database for the user is TSOMON STC-USERID.
- The TCP/IP port that failed access has a name SAPSYS and belongs to the TCP/IP stack named TCP2 on z/OS system SX00.
- The TCP/IP port belongs to the resource class SERVAUTH and the resource name that we use to query RACF is EZB.PORTACCESS.SX00.TCP2.SAPSYS.

When the user UTSM tried to open the port named SAPSYS on the system SX00 and on TCP/IP stack TCP2, RACF checked its database for a discrete profile specific to EZB.PORTACCESS.SX00.TCP2.SAPSYS. It could not find it, but instead found a generic profile EZB.PORTACCESS.*.*.SAPSYS, which covered the resource. (A generic profile protects multiple resources having similar characteristics.) The user UTSM was not in the access list and RACF failed the request.

Now let us see what you should have done to protect the TCP/IP port and to give proper access to the legitimate user. See Example 8-2.

Example 8-2 RACF commands to protect a TCP/IP port

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST(SERVAUTH)
RDEFINE EZB.PORTACCESS.*.*.SAPSYS UACC(NONE)
PERMIT EZB.PORTACCESS.*.*.SAPSYS CLASS(SERVAUTH) ID(UTSM) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Let us go through each of those TSO commands and explain why it is needed and what it does. You need to have RACF authority to issue these commands.

SETROPTS CLASSACT(SERVAUTH) activates the SERVAUTH class of profiles that protect resources managed by the z/OS TCP/IP stack. When you activate a resource class, you are basically telling RACF to do authorization checking whenever someone tries to access any resource protected under that class. RACF keeps resources with similar characteristics in one class. TCP/IP resources such as the stack, network, and port belong to the SERVAUTH class.

Another example of a resource class is OPERCMDS that protects the use of sensitive operator commands such as VARY TCPIP.

Note: In most installations the SERVAUTH class will be active. In that case, you can skip this step. You can issue a RACF command SETROPTS LIST in TSO to check if it is active. Look in the section starting with ACTIVE CLASSES =.

SETROPTS RACLIST(SERVAUTH) tells RACF to read the profiles for the SERVAUTH class from the RACF database into the RACF data space and to activate the sharing of these in-storage profiles. With these profiles in storage, RACF does not have to do an input/output (I/O) to read the RACF database when making an access decision, and this improves performance.

RDEFINE EZB.PORTACCESS.*.*.SAPSYS UACC(NONE) defines a generic profile to cover all TCP/IP ports that have the name SAPSYS. The profile that you have to define to protect the TCP/IP ports is of the format EZB.PORTACCESS.systemname.stackname.portname. The first two qualifiers of the profile have to be EZB.PORTACCESS. This tells the system that this profile is protecting TCP/IP ports. The third qualifier specifies the z/OS system name, the fourth one specifies the name of the TCP/IP stack, and the last one the name of the port. We have the wildcard character “*” for systemname and stackname. This will cover TCP/IP ports with name SAPSYS on all TCP/IP stacks and on all z/OS systems. Note that we have set UACC(NONE) to restrict its access.

PERMIT EZB.PORTACCESS.*.*.SAPSYS CLASS(SERVAUTH) ID(UTSM) ACCESS(READ) gives READ access for the user ID UTSM to access the TCP/IP port.

SETROPTS RACLIST(SERVAUTH) REFRESH updates the in-storage SERVAUTH class profiles in the RACF data space.

Let us look at another example. The socket option IPV6_NEXTHOP is sensitive and you want to restrict its usage to authorized persons (see Example 8-3).

Example 8-3 RACF commands to restrict the use of socket option IPV6_NEXTHOP

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST(SERVAUTH)
RDEFINE SERVAUTH EZB.SOCKOPT.*.*.IPV6_NEXTHOP UACC(NONE)
PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(UTSM) ACCESS(READ)
PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(*) WHEN(PROGRAM(TSOMON)) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Example 8-3 shows the RACF commands you will need.

RDEFINE SERVAUTH EZB.SOCKOPT.*.*.IPV6_NEXTHOP UACC(NONE) defines the RACF profile to restrict the use of the IPV6_NEXTHOP socket option.

PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(UTSM) ACCESS(READ) gives access for the user UTSM to use the IPV6_NEXTHOP socket option in the user's programs.

You can also protect the use of the socket option in another way. You can say that any user can use the socket option, provided the user is doing it via a specific program. That way you are giving authority to a program rather than to a user to access the resource (socket option).

PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(*) WHEN(PROGRAM(TSOMON)) ACCESS(READ) allows anyone to access the socket option, provided the user is using program TSOMON.

In the following section, we show the various network resources protected by RACF.

8.2 How to protect your network resources

You have seen how you can define resource profiles to protect a TCP/IP port and also to protect the use of an IPv6 socket option. All network resources are protected by RACF in the same way. Most TCP/IP resources are protected by profiles defined in the SERVAUTH resource class.

> **Tip:** All z/OS Communications Server profiles in the SERVAUTH class have EZA, EZB, or IST as the High Level Qualifier (HLQ).

To protect a resource, all you need to do is:

- If the SERVAUTH class is not active, activate it with the SETROPTS command. You need to do this only once in your system and in most cases this will already be active on your system.
- Identify the profile that protects the resource from Chapter 9, "Protecting network resources" on page 313. Define the profile with the RDEFINE command.
- Allow access to authorized users to this profile using the PERMIT command.
- Refresh the RACLIST in-storage profiles of the SERVAUTH class in the RACF data space using the SETROPTS command.

Refer to Chapter 9, "Protecting network resources" on page 313, for more detailed information about how RACF protects the various TCP/IP resources using the above method.

8.3 How to protect your programs

One of the main strengths of the z/OS platform is the foolproof protection of its programs from unauthorized alteration. This is one of RACF's most powerful features and makes the z/OS platform immune to computer viruses, making it stand out from most other platforms. Very strict controls and protection mechanisms in RACF make it is impossible for any unauthorized person to modify programs on the z/OS platform.

z/OS security has evolved and matured over a period of more than quarter of a century. Many other operating systems' platforms cannot match the inherent security of the z/OS platform, because they were originally designed either with a single user in mind or for academic collaboration, where security is a hindrance.

RACF uses the following mechanisms to secure programs from unauthorized access:

- Authorized Program Facility (APF)
- Program Protection by RACF resource class PROGRAM
- Program Access to Data Sets (PADS)
- Controlling Program Access by SYSID
- The sticky bit in the UNIX environment

Let us examine each one of them.

Authorized Program Facility

z/OS protects the use of sensitive system functions and supervisor calls (SVC) using the APF facility. Programs have to be APF authorized to use these system functions. To get APF authorization, the program should meet two conditions:

- It must reside in a library that is in the APF list or in the Link Pack Area (LPA).
- The program must be link-edited with authorization code AC=1.

In addition, the program libraries are protected by RACF. These protections make virus attacks impossible on z/OS.

Program protection by RACF resource class PROGRAM

RACF treats program load modules as protected resources. PROGRAM is the RACF resource class that protects programs. Example 8-4 shows the RACF commands to protect a program. You use the ADDMEM parameter in RDEFINE to specify the library where the program resides.

Example 8-4 RACF command to protect a program

```
RDEFINE PROGRAM MYPROGRAM ADDMEM('SYS1.LINKLIB') UACC(NONE)
PERMIT MYPROGRAM CLASS(PROGRAM) ID(SOMEUSER) ACCESS(READ)
```

Program Access Control

You can use the Program Access Control facility to specify that access to a resource is allowed only if you are accessing it using a specific program. The program itself has to be in a controlled library and restricted to only authorized users.

In Example 8-5, we show how to restrict the use of advanced IPV6 socket options by program access control.

Example 8-5 PADS to protect use of sockect option

```
PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(*) WHEN(PROGRAM(TSOMON)) ACCESS(READ)
```

Controlling program access by SYSID

Access to programs (load modules) can be controlled based on the SMF system ID of the z/OS system, as shown in Example 8-6.

Example 8-6 Controlling program access by system ID

```
PERMIT MYPROGRAM CLASS(PROGRAM) ID(SOMEUSER) WHEN(SYSID(PROD_SYSTEM))
```

The sticky bit in the z/OS UNIX environment

Because z/OS UNIX files are not as secure as MVS data sets, sensitive programs running under z/OS UNIX do not load from the z/OS UNIX file system. z/OS will instead turn to the standard MVS search order to look for a copy of the executable file in an MVS load library. z/OS UNIX System Services uses the sticky bit on the program library to bypass loading of a program from the UNIX Systems Services file system. Often the program needs to reside in APF authorized libraries protected by program control.

Sticky bit is one of the bits in the Access Control List (ACL) of the z/OS UNIX file. To see if the sticky bit is set on a file (program), you can issue the UNIX command **ls -l**, as shown in Example 8-7. The “T” as the last character in the access list for the file IMWCGIBN indicates that its sticky bit is on. This means the system will not look for the program IMWCGIBN in the UNIX files; instead it will search for it in more secure authorized z/OS libraries.

Example 8-7 UNIX command to show the sticky bit

```
/usr/lpp/internet: >ls -l
total 40
-rw-r--r-- 2 WEBADM IMWEB envvars
-rw-r--r-- 2 WEBADM IMWEB httpd_msg.cat
drwxr-xr-x 2 WEBADM IMWEB IBM
-rwxr--r-T 2 WEBADM IMWEB IMWCGIBN
Drwxr-xr-x 2 WEBADM IMWEB logs
Drwxr-xr-x 3 WEBADM IMWEB Samples
Drwxr-xr-x 10 WEBADM IMWEB ServerRoot
/usr/lpp/internet: >
```

8.4 How to associate a user ID with a started task

RACF makes sure that everyone who accesses the system resources is accountable. This applies to the system tasks as well. For this, RACF associates every started task (STC) with a specific user ID. RACF keeps this information in a resource class called STARTED. Example 8-8 shows you how to define this to RACF.

Example 8-8 RACF commands to associate a user ID with a started task

```
SETROPTS GENERIC(STARTED)
SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
RDEFINE STARTED TCPIP.* STDATA(USER(tcpip_user) PRIVILEGED(NO) TRUSTED(NO) TRACE(NO))
RDEFINE STARTED FTPD.* STDATA(USER(tcpip_user) PRIVILEGED(NO) TRUSTED(NO TRACE(NO))
SETROPTS  RACLIST(STARTED)  REFRESH
```

Before you can start an STC in the system, you have to tell RACF to give the STC user ID access to all the resources used by the STC using the PERMIT commands.

8.5 How to set up security for daemons in z/OS UNIX

TCPIP and other related daemons work in the z/OS UNIX environment and use many of its services. Therefore, it is important to understand how to set up security for daemons working in the z/OS UNIX security environment.

To set up a daemon under z/OS UNIX, the following steps are necessary:

1. Define a user ID for the daemon.
2. Define an OMVS segment for the user ID.
3. Give superuser authority for the user ID.
4. Give user ID access to various RACF profiles protecting the resources for which the daemon will need access.
5. Associate the user ID with the daemon.
6. For some daemons, you have to turn the sticky bit on to indicate that the program module resides in a protected z/OS library rather than the z/OS UNIX file pointed to by the module.

For more details, refer to *z/OS V1R8.0 UNIX System Services User's Guide*, SA22-7801.

8.6 RACF multilevel security for network resources

Multilevel security (MLS) addresses government requirements for highly secure data. This supports sharing of classified information among multiple agencies on demand. As security controls become more critical in the emerging on demand virtual environments, this new technology has applications in the general business sectors as well. This secondary layer is on the top of existing RACF resource protection.

8.6.1 Basic concepts of MLS

In MLS, the resources are divided into a number of categories based on where they belong. For example, you can classify the resources of your organization based on departments such as PAYROLL, PERSONNEL, RESEARCH, MARKETING, SALES, PRODUCTION, and so on. Resources in each category are further classified based on their importance and sensitivity. For example, you can classify them into GENERAL, CONFIDENTIAL, SENSITIVE, and TOP-SECRET in the ascending order of their importance and sensitivity. This classification is hierarchical, which means GENERAL would be the lowest that everyone can access. The level goes up with CONFIDENTIAL, then SENSITIVE, and the highest level is TOP-SECRET.

After this classification is done, assign a similar category and security level for each user by default. After you switch on MLS, when a user tries to access a resource, RACF will check if the user's security level is equal to or above that of the resource and also that the user and the resource belong to the same category. Thus, a user in the PERSONNEL department will be able to access a resource only in the PERSONNEL department. Also, the user should have the right security level. For example, a user from PERSONNEL with a security level of GENERAL will not be able to access a PERSONNEL resource with a security level of CONFIDENTIAL. A user from MARKETING will not be able to access a resource from RESEARCH, though the user may have TOP-SECRET security level in the MARKETING department.

RACF uses security labels (SECLABELs) to enforce multilevel security.

SECLABELS

A *SECLABEL*, or security label, consists of two entities:

- A security *category* such as PAYROLL, PERSONNEL, or RESEARCH
- A security *level* such as CONFIDENTIAL, SENSITIVE, or TOP-SECRET

The security administrator sets security labels for each user and each resource. When a user tries to access a resource, RACF allows access only if the security *level* in the user's SECLABEL is higher or equal to the security *level* specified in the resource's SECLABEL for the security *category* being accessed.

A user might be permitted to access several security labels, but can only be logged onto one of them at a time.

You can provide additional layers of protection for your network resources by implementing MLS. For more details, refer to section 4.1 of *z/OS 1.6 Security Services Update*, SG24-6448.

8.7 Digital certificates in RACF

RACF allows you to create and maintain security keys, key rings, and digital certificates in the RACF database. In a client/server environment, RACF has the ability to map a client's digital certificate to a RACF user ID by either storing the digital certificate in the RACF database or mapping by using a certificate name filter rule. A digital certificate or digital ID, issued by a Certificate Authority, contains information that uniquely identifies the client.

See Chapter 3, "IP Security" on page 75, for information about how to set up digital certificate keys and key rings.

8.8 Further information

You can find samples of jobs with the RACF commands required for z/OS Communications Server and applications in your installation library TCPIP.SEZAINST(EZARACF). The high-level qualifier of this library could be different in your installation.

9

Protecting network resources

This chapter discusses the RACF security profiles that can be used to protect access to various network resources.

Table 9-1 Topics discussed in this chapter

Section	Topic
9.1, “The SERVAUTH resource class” on page 314	We explain the basic setup using the RACF SERVAUTH class to protect your resources.
9.2, “Protecting your TCP/IP stack” on page 314	We show you how the TCP/IP resource is protected by RACF.
9.3, “Protecting your network access” on page 315	This section explains how to protect your network.
9.4, “Protecting your network ports” on page 318	We explain how RACF protects your ports.
9.5, “Protecting the use of socket options” on page 320	We explain how to restrict the use of sensitive socket options.
9.6, “Protecting sensitive network commands” on page 321	We show how to set up security for your sensitive network commands to prevent unauthorized use.
9.7, “Protecting FTP-related resources” on page 328	We show the setup to protect FTP resources.
9.8, “Protecting network management resources” on page 329	We show how to use RACF to protect network management resources (such as data collection agents).
9.9, “Protecting miscellaneous resources” on page 329	This section explains RACF support to protect miscellaneous network resources.

© Copyright IBM Corp. 2007. All rights reserved.

9.1 The SERVAUTH resource class

Most network resources are protected by the SERVAUTH resource class profiles. To protect a resource:

- If the SERVAUTH class is not active, you need to activate it with the SETROPTS command. You need to do this only once in your system and in most cases this would already be active on your system.
- Identify the profile that protects the resource from the list that follows in this chapter. Define the profile with the RACDEF command.
- Allow access to authorized users to this profile with the PERMIT command.
- Refresh the RACLIST in-storage profiles of the SERVAUTH class in the RACF data space with the SETROPTS command.

In the following sections, we describe how to set up the RACF profiles to protect various network resources.

9.2 Protecting your TCP/IP stack

This section discusses the Security Access Facility (SAF) security profiles that can be used to protect access to a TCP/IP stack's resources.

9.2.1 Stack access overview

Stack access control provides a way to permit or deny users or groups of users access to a TCP/IP stack. The function controls the ability of a user to open an IP socket with the socket() API function. Stack access control is implemented by defining a SERVAUTH class RACF profile. The profile name is in the format:

`EZB.STACKACCESS.`*`sysname`*`.`*`tcpipname`*

Where:

- EZB.STACKACCESS is constant.
- *sysname* is the name of the z/OS image (&SYSNAME symbol).
- *tcpipname* is the job name of the TCP/IP stack.

9.2.2 Example setup

This example shows how to control access to the TCP/IP stack running with a job name of TCPIPD on the z/OS system SC30. The first thing to do is to add the SERVAUTH STACKACCESS profile to RACF. As mentioned previously, the format of the profile name is EZB.STACKACCESS.*sysname*.*tcpname*, therefore in our example, the profile name will be EZB.STACKACCESS.SC30.TCPIPD, as shown in Example 9-1.

Example 9-1 RACF SERVAUTH profile to protect TCP/IP stack access

```
RDEFINE EZB.STACKACCESS.SC30.TCPIPD UACC(NONE)
PERMIT EZB.STACKACCESS.SC30.TCPIPD CLASS(SERVAUTH) ID(UTSM) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

We have specified that Universal Access (UACC) is none, meaning that the default for any user is to be denied access. Then we have given access to user UTSM using the RACF

PERMIT command. When the profile has been added, we refresh the in-storage RACF profiles with a **`setropts raclist(servauth) refresh`** command.

We have just covered enabling a user to access the TCP/IP stack. This concept of a user applies equally to the owner of any server running on the stack. For example, the FTP started task user ID has to be given access to the stack's RACF profile.

9.3 Protecting your network access

Network access control enables system administrators to represent access to an IP network, subnetwork, or host as a RACF resource. The ability to send IP packets to those networks, subnetworks, or hosts can then be permitted or denied at a RACF user or group level. This feature provides an additional layer of security to any authentication or authorization that is used at the target system. It might be used, for example, to prevent access to the Internet by anyone except the SMTP server, or it could be used to stop general users attempting to Telnet to a server that contained payroll information.

9.3.1 Network access control overview

In the TCP/IP profile, there is a parameter block, NETACCESS/ENDNETACCESS. This is where you specify the mapping of an IP network, subnetwork, or host to a SAF profile. Example 9-2 shows a sample NETACCESS block.

Example 9-2 Sample NETACCESS block

```
NETACCESS
  10.40.2.0/24       MYSUBNET    ;my workstation subnet
  10.40.2.119/32     MYPC        ;my workstation
  DEFAULT 0          WORLD       ;everything else
ENDNETACCESS
```

In this example, access to hosts on subnet 10.40.2 is mapped to RACF profile MYSUBNET, access to host 10.40.2.119 is mapped to SAF profile MYPC, and access to any other host is mapped to SAF profile WORLD. These RACF profiles are defined to RACF in the SERVAUTH class. The profile name to be defined is in the following format:

`EZB.NETACCESS.`*`sysname`*`.`*`tcpipname`*

Where:

- EZB.NETACCESS is constant.
- *sysname* is the name of the z/OS image (&SYSNAME symbol).
- *tcpipname* is the job name of the TCP/IP stack.
- *resourcename* is the name specified in the NETACCESS block.

Tip: On the NETACCESS statement, there are two ways to specify the subnet mask. One way is the traditional decimal notation, such as 255.255.255.0. The second way is to use a number, up to 32, that specifies the number of bits, left to right, that should be used as a subnet mask if the mask is expressed in binary. For example, the subnet mask 255.255.255.0 expressed in binary is 11111111.11111111.11111111.00000000. Note that there are 24 bits set on. To specify this particular mask on a NETACCESS statement, you could use either of the following two ways, using the IP address 192.168.100.0 as an example:

```
NETACCESS 192.168.100.0     255.255.255.0
```

Or:

```
NETACCESS 192.168.100.0/24
```

The system that we used in Example 9-2 on page 315 was on a z/OS image named SC30 with a TCP/IP stack name of TCPIPD. Therefore, the three profiles that need to be defined are:

- EZB.NETACCESS.SC30.TCPIPD.MYSUBNET
- EZB.NETACCESS.SC30.TCPIPD.MYPC
- EZB.NETACCESS.SC30.TCPIPD.WORLD

If you define these profiles to RACF with UACC(NONE), then users must be specifically permitted access to these profiles in order to send IP packets to the addresses represented by the profiles.

If a user is attempting to send an IP packet to a host that is not covered by any network/mask entry in the NETACCESS block, access is automatically allowed. However, if a DEFAULT statement is present, then access is granted or denied based on the user's access to the SAF profile mapped by the DEFAULT statement.

9.3.2 Server considerations

End users are not the only users of TCP/IP to be affected by NETACCESS control. Any IP applications (servers) that run under their own user IDs would need access to the RACF profiles of the desired networks, if these networks are protected by a NETACCESS statement. For example, a server such as the FTP daemon would need to be permitted access to any hosts or subnets that an FTP transfer will be performed with.

There is another subtlety to be considered. If you have a user out in the network who wants to FTP to/from your FTP server, then the user ID that this user logs on with must have been permitted to NETACCESS if the user's own IP address or network is protected.

9.3.3 Using NETSTAT for network access control

The console command D TCPIP,stackname,Netstat,ACCess,NETWork shows how IP addresses/masks are mapped to SAF profiles. See Figure 9-1 for the NETSTAT console command to display the network access control for the test TCPIPD system.

```
D TCPIP,TCPIPD,NETSTAT,ACCESS,NETWORK
EZD0101I NETSTAT CS V1R8 TCPIPD 147
NETWORK ACCESS INFORMATION
INBOUND: NO   OUTBOUND: YES
SAF NAME  NETWORK PREFIX AND PREFIX LENGTH
--------  -------------------------------
WORLD     DEFAULT
  PRFNM: <NONE>                                  SECLABEL: <NONE>
MYSUBNET  10.40.2.0/24
  PRFNM: <NONE>                                  SECLABEL: <NONE>
MYPC      10.40.2.119/32
  PRFNM: <NONE>                                  SECLABEL: <NONE>
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

Figure 9-1 NETSTAT command to display network access control in our test system

The TSO NETSTAT command does not display this information.

9.3.4 Working example of network access control

We implement network access control on the TCP/IP stack discussed in 9.3.1, “Network access control overview” on page 315. In that section, we show the three SAF profile names that need to be defined for the given NETACCESS block. Example 9-2 shows the configuration.

When the RACF profiles names shown in 9.3.1, “Network access control overview” on page 315, have been defined to RACF, we issue the TSO command PING 10.40.2.119 to send a packet to the workstation 10.40.2.119. Because we have not been permitted access to the 10.40.2.119 host (profile MYPC), you would expect an error. Figure 9-2 shows the error message from RACF indicating that access to the MYPC profile was not permitted.

```
 ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TESTTEST          )
   EZB.NETACCESS.SC30.TCPIPD.MYPC CL(SERVAUTH)
   INSUFFICIENT ACCESS AUTHORITY
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
 CS V1R8: Pinging host 10.40.2.119
 sendto(): EDC5111I Permission denied.
```

Figure 9-2 RACF error while attempting PING to host covered by host profile

Note: Even though both the host 10.40.2.119 and its subnet 10.40.2.0 are protected by RACF profiles, the RACF check is only performed on the *most specific* network/host entry. An easier way to say this is that the entries in the NETACCESS block are checked starting with those with the most bits specified in the subnet mask first, until a match is found.

If we now attempt to ping a new host 10.40.2.120, we would expect a RACF error when the TCP/IP address space does a RACF check for access to the MYSUBNET profile, as this is the most specific entry for that host address. We attempt a ping to 10.40.2.120 and get the error as expected. Figure 9-3 shows this.

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
  EZB.NETACCESS.SC30.TCPIPD.MYSUBNET CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS V1R8: Pinging host 10.40.2.120
sendto(): EDC5111I Permission denied.
```

Figure 9-3 RACF error while attempting to ping a host covered by subnet profile

Lastly, we show an example of the error you would receive when attempting network access to a host not specified on any NETACCESS statements. This assumes that you have coded a DEFAULT statement in the NETACCESS block. If you do not, access permission is not checked for any host not covered by any other NETACCESS statement. We tried to ping 10.40.5.10, an IP address that is not specifically stated in a host or subnet entry. The DEFAULT NETACCESS statement is therefore used for the SAF check that is mapped to profile WORLD. As shown in Figure 9-4, we got a RACF error message indicating that we did not have access to EZB.NETACCESS.SC63.TCPIPC.WORLD.

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
  EZB.NETACCESS.SC30.TCPIPD.WORLD CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS V1R8: Pinging host 10.40.5.10
sendto(): EDC5111I Permission denied.
```

Figure 9-4 RACF error while attempting to ping a host covered by the DEFAULT statement

9.4 Protecting your network ports

The ability of a server to bind to a specific port can be controlled in a number of ways using the UDPCONFIG, TCPCONFIG, and PORT (or PORTRANGE) TCP/IP profile statements.

The use of TCPCONFIG RESTRICTLOWPORTS and UDPCONFIG RESTRICTLOWPORTS is encouraged to enhance security. If these statements are present, low ports (< 1024) can only be bound when at least one of the following is true:

- The bind is issued from a process with a UNIX superuser (UID 0).
- The bind is issued from an APF-authorized application.
- The port is reserved for the application by job name, which may include *, OMVS, or TSO user ID.
- If an SAF resource name is used, the binding process's user ID must be permitted to the resource by the security product (described later).
- The RESERVED keyword will shut down a port from being used by any job name at all. The keyword can also be specified on the PORTRANGE statement. This readily allows an installation to clamp down very tightly on usage of ports if such control is desired.
- Specifying a job name on a PORT or PORTRANGE statement restricts the use of that port (and protocol) to the specified job name. Multiple PORT statements can be specified for a TCP port but not for UDP. Note that a job name of “*” (the wildcard character) is normally used with the SAF keyword, which is described next.
- Specifying the SAF keyword and profile name provides a mapping from a port and protocol to a SAF SERVAUTH class profile. A server attempting to bind to this port and

protocol is checked for SAF access to the profile named. The use of the SAF keyword is covered in this section.

9.4.1 The PORT/PORTRANGE SAF keyword

The SAF keyword can be specified along with all other valid options on the PORT and PORTRANGE statements. The special job name wildcard “*” is normally used with the SAF keyword so that access to the port is completely handled by the server's SAF profile rather than job name. Of course, a specific job name and the keyword SAF can still be coded together if you would like to secure not only the job name that can bind a socket, but also the SAF user associated with the job.

We show how to protect the FTP ports to illustrate the concept. Sample PORT statements for the system SC30 are shown in Figure 9-5.

```
PORT
    20 TCP *  NOAUTOLOG SAF FTP20 ; FTP Server
    21 TCP OMVS  BIND 10.40.1.230 SAF FTP21 ;control port
    23 TCP INTCLIEN            ; MVS Telnet Server
    512 UDP RESERVED           ; Shut down port 512
    23 TCP OMVS BIND 10.40.1.230; OE Telnet Server
   500 UDP IKED                ; IKE Daemon
  4500 UDP IKED                ; IKE Daemon
```

Figure 9-5 PORT statements for stack TCPIPC

Given the PORT statements in Figure 9-5, UDP port 512 is reserved, and therefore completely unavailable for use. Any process attempting to use TCP ports 20 or 21 have to be SAF authorized. The following SERVAUTH profiles have to be defined (assuming the system name is SC30 and the TCP/IP stack name is TCPIPD):

- EZB.PORTACCESS.SC30.TCPIPD.FTP20
- EZB.PORTACCESS.SC30.TCPIPD.FTP21

This form of port reservation might be used when a reserved low port needs to be accessed by many potential users via a client program that is not APF-authorized. All users requiring the ability to run this program have to be permitted to this RACF resource.

With FTP ports 20/21 (or any well-known port usually used for a system-type server), there is a possible security exposure when permitting port use by the SAF keyword. When a user is permitted to bind to, for example, port 20, the user can bind to that port using any program, not just through the FTP server. To prevent this, we recommend that you do not code a SAF keyword for port 20, but instead use the RESTRICTLOWPORTS parameter of the TCPCONFIG statement in conjunction with specifying “OMVS” as the job name for port 20. This restricts the use of port 20 to APF-authorized programs, UNIX superusers, or the FTP server.

The FTP daemon is the only user that needs to access the FTP control port 21, and hence should have access to the RACF SERVAUTH profile EZB.PORTACCESS.SC30.TCPIPD.FTP21. If the FTP server does not have access to the profile, you will get a RACF error similar to that shown in Figure 9-6.

```
S FTPDD
$HASP100 FTPDD    ON STCINRDR
IEF695I START FTPDD    WITH JOBNAME FTPDD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 FTPDD    STARTED
ICH408I USER(TCPIP   ) GROUP(TCPGRP  ) NAME(##################) 400
  EZB.PORTACCESS.SC30.TCPIPD.FTP21 CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
+EZY2714I FTP server shutdown in progress
+EZYFT59I FTP shutdown complete.
$HASP395 FTPDD    ENDED
```

Figure 9-6 FTP server unauthorized to use port 21

The FTP data connection port (20) is bound under the identity of the end user, not the FTP daemon. Therefore, if the PORT statement for port 20 is configured as shown in Figure 9-5 on page 319, then all end users (including the default user, if defined) who can potentially perform FTP need to be permitted to the port 20 RACF SERVAUTH profile EZB.PORTACCESS.SC30.TCPIPD.FTP20.

9.4.2 Using NETSTAT to display Port Access control

The TCPIPC stack contains the PORT statement shown in Figure 9-5 on page 319. The console command D TCPIP,stackname,Netstat,PORTlist in Figure 9-7 shows the configuration of the ports, and whether a SAF profile is associated with a port (the F in the FLAGS column).

```
D TCPIP,TCPIPD,NETSTAT,PORTLIST
EZD0101I NETSTAT CS V1R8 TCPIPD 602
PORT# PROT USER     FLAGS    RANGE       SAF NAME
00020 TCP  *        DF                   FTP20
00021 TCP  OMVS     DABFU                FTP21
      BINDSPECIFIC: 10.40.1.230
00023 TCP  OMVS     DABU
      BINDSPECIFIC: 10.40.1.230
00023 TCP  TCPIPD   DAU
00500 UDP  IKED     DA
00512 UDP  RESERVED DA
04500 UDP  IKED     DA
7 OF 7 RECORDS DISPLAYED
END OF THE REPORT
```

Figure 9-7 NETSTAT PORTLIST console command display for TCPIPD

9.5 Protecting the use of socket options

You can use RACF profiles to prevent the misuse of the sensitive SO_BROADCAST and IPv6 API socket options.

9.5.1 SO_BROADCAST Socket option access control

The SO_BROADCAST option provides control over the broadcast function, which can be prone to misuse if not restricted.

RACF profile EZB.SOCKOPT.*sysname.tcpname*.SO_BROADCAST controls this option.

Where:

- *sysname* is the z/OS system name. Our system name was SC30.
- *tcpname* is the jobname of the TCP/IP stack. Our stack was TCPIPD.

We show sample RACF commands to define this resource and then give access to OMPROUTE, which needs to use the SO_BRODCAST socket option. See Example 9-3.

Example 9-3 Sample RACF commands to protect SO_BROADCAST socket option

```
RDEFINE SERVAUTH EZB.SOCKOPT.SC30.TCPIPD.SO_BROADCAST UACC(NONE)
PERMIT EZB.SOCKOPT.SC30.TCPIPD.SO_BROADCAST CLASS(SERVAUTH) ACCESS(READ) ID(OMPROUT)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

If you want, you can protect this option on all your systems and on all the stacks using a single generic profile. The profile that you can use is:

```
EZB.SOCKOPT.*.*.SO_BROADCAST
```

9.5.2 IPv6 advanced socket API options

The z/OS Communications Server has a number of new advanced socket API options that need to be restricted to only authorized users. There is one SERVAUTH class profile to protect each of these socket options. These profiles are shown in the following list.

- IPV6_NEXTHOP - EZB.SOCKOPT.*sysname.tcpname*.IPV6_NEXTHOP
- IPV6_TCLASS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_TCLASS
- IPV6_RTHDR - EZB.SOCKOPT.*sysname.tcpname*.IPV6_RTHDR
- IPV6_HOPOPTS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_HOPOPTS
- IPV6_DSTOPTS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_DSTOPTS
- IPV6_RTHDRDSTOPTS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_RTHDRDSTOPTS
- IPV6_PKTINFO - EZB.SOCKOPT.*sysname.tcpname*.IPV6_PKTINFO
- IPV6_HOPLIMIT - EZB.SOCKOPT.*sysname.tcpname*.IPV6_HOPLIMIT

Note that the last qualifier of the profile is the same as the socket option itself.

9.6 Protecting sensitive network commands

This section discusses the ways to control the use of the TCP/IP system administration commands. These commands are categorized by where they originate.

From the z/OS console, you can use the DISPLAY and VARY commands for the TCP/IP address spaces. The VARY TCPIP command can be used to stop and start TCP/IP interfaces, reload configuration parameters, start and stop traces, drop TCP connections, and quiesce the TN3270 server. This command is very powerful and should only be authorized to operators or system administrators.

From TSO, there is the NETSTAT command, and from the UNIX shell there is the **onetstat** command. Both of these commands are used primarily to display information about the local TCP/IP environment. You may want to restrict these commands so that people cannot obtain information about your TCP/IP configuration, perhaps in preparation for an attack of some kind.

9.6.1 z/OS VARY TCPIP command security

This section describes the mechanisms by which you can limit users to the VARY TCPIP commands.

RACF profile details

The z/OS console VARY TCPIP commands are protected with RACF profiles defined in the resource class OPERCMDS. You can define a single profile to represent all VARY TCPIP commands or you can specify individual profiles for each VARY TCPIP command option.

The format of the profile name is:

`MVS.VARY.TCPIP.`*`command`*

Where:

- MVS.VARY.TCPIP is a constant.
- *command* is either a double asterisk (**), meaning all command options, or a specific VARY TCPIP option name, such as OBEYFILE.

An important thing to note about the profile name is that it does not specify a z/OS image name or TCP/IP stack name. Therefore, if there is more than one stack on your z/OS image or your SAF database is shared between multiple z/OS systems, granting access to a command enables that command to be performed by a user against any TCP/IP stack in any z/OS system that shares the database.

Protecting VARY TCPIP at the command level

To specify protection at the command level (any option), specify the generic OPERCMDS profile with a profile name of MVS.VARY.TCPIP.**. Figure 9-8 shows how this is done using RACF commands.

```
SETROPTS GENERIC(OPERCMDS)
RDEFINE OPERCMDS (MVS.VARY.TCPIP.**) UACC(NONE)
SETROPTS GENERIC(OPERCMDS) REFRESH
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure 9-8 Defining generic VARY TCPIP profile to protect command with all options

Note: If the MVS.VARY.TCPIP.** profile is defined, any user who needs to use any VARY TCPIP command must have CONTROL access to this profile, *regardless* of whether they have access to other MVS.VARY.TCPIP.command profiles.

Protecting VARY TCPIP at the command option level

You may decide to protect the VARY TCPIP command at a more granular level so that you can control who has authority to use the options of the VARY TCPIP command. To protect the command options, define the particular VARY TCPIP option that you want to protect as the last qualifier in the profile name. Exceptions to this rule are the VARY TCPIP START and VARY TCPIP STOP commands that are protected together with the profile named MVS.VARY.TCPIP.STRTSTOP. See Table 9-2 for a list of RACF profile names to protect the VARY TCPIP command options.

Table 9-2 List of RACF profiles to protect various VARY TCPIP console commands

RACF profile name	Command protected
MVS.VARY.TCPIP.**	All VARY TCPIP options
MVS.VARY.TCPIP.DROP	VARY TCPIP,,DROP
MVS.VARY.TCPIP.OBEYFILE	VARY TCPIP,,OBEYFILE
MVS.VARY.TCPIP.PKTTRACE	VARY TCPIP,,PKTTRACE
MVS.VARY.TCPIP.STRTSTOP	VARY TCPIP,,START or VARY TCPIP,,STOP

Figure 9-9 shows the VARY TCPIP,,STOP and VARY TCPIP,,START commands being protected.

```
RDEFINE OPERCMDS MVS.VARY.TCPIP.STRTSTOP UACC(NONE)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure 9-9 Defining specific VARY TCPIP profile to protect VARY TCPIP,,STOP/START commands

VARY TCPIP command security scenario

The command V TCPIP,TCPIPC,STOP,OSA22E0 was chosen to test the console RACF security profiles. The command output is shown in Figure 9-10 before any RACF security profiles were defined to the system. The SAF profile that controls the V TCPIP,,STOP command (in addition to the generic MVS.VARY.TCPIP.** if defined) is MVS.VARY.TCPIP.STRTSTOP.

```
V TCPIP,TCPIPC,STOP,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STOP,OSA2080
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA2080
```

Figure 9-10 VARY TCPIP,TCPIPC,STOP command output: No RACF profiles defined yet

For the first test, we only defined the generic MVS.VARY.TCPIP.** profile to protect all options of the V TCPIP command.

For the second test, we additionally defined the MVS.VARY.TCPIP.STRTSTOP profile to protect the V TCPIP,,STOP command. Both times, unauthorized use of the command caused the expected RACF error.

Defining the generic profile to protect all V TCPIP commands

The generic profile MVS.VARY.TCPIP.** was added to the OPERCMDS class with UACC(NONE), as shown in Figure 9-8 on page 322. At this stage, no user had any access to this profile. A user then attempted a V TCPIP,TCPIPC,START,OSA2080 command from the console, which resulted in the RACF error shown in Figure 9-11. Note that the profile being SAF checked was MVS.VARY.TCPIP.**

```
V TCPIP,TCPIPC,START,OSA2080
IEE345I VARY     AUTHORITY INVALID, FAILED BY SECURITY PRODUCT
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          ) 689
  MVS.VARY.TCPIP CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  FROM MVS.VARY.TCPIP.** (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

Figure 9-11 RACF error for VARY TCPIP STOP command showing generic profile violation

Defining the specific profile to protect the V TCPIP,,START command

The specific profile MVS.VARY.TCPIP.STRTSTOP was added to the OPERCMDS class with UACC(NONE), as shown in Figure 9-9 on page 323. At this stage, both the MVS.VARY.TCPIP.STRTSTOP and the generic MVS.VARY.TCPIP.** profiles were defined, and the user did not have access to either of them.

After the V TCPIP,TCPIPC,STOP,OSA2080 command was issued, Figure 9-12 shows that the profile name that is causing the access problems is MVS.VARY.TCPIP.**, even though the MVS.VARY.TCPIP.STRTSTOP profile is defined. This confirms what was said in “Protecting VARY TCPIP at the command level” on page 322, that is, the generic profile is always checked first, if defined.

```
V TCPIP,TCPIPC,START,OSA2080
IEE345I VARY     AUTHORITY INVALID, FAILED BY SECURITY PRODUCT
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          ) 699
  MVS.VARY.TCPIP CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  FROM MVS.VARY.TCPIP.** (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

Figure 9-12 RACF error for VARY TCPIP,,STOP command shows generic profile name

If the generic profile MVS.VARY.TCPIP.** is defined, any user who wants to enter a VARY TCPIP command must have CONTROL access to it. Example 9-4 shows the RACF commands to give such access to user CS09.

*Example 9-4 Giving CONTROL access to MVS.VARY.TCPIP.***

```
PE MVS.VARY.TCPIP.** CLASS(OPERCMDS) ID(CS09) ACCESS(CONTROL)
SETROPTS RACLIST(OPERCMDS) REFRESH
SETROPTS GENERIC(OPERCMDS) REFRESH
```

Now that we have CONTROL access to the generic profile MVS.VARY.TCPIP.**, the SAF check is for the profile that controls the VARY TCPIP,,STOP command, which is V TCPIP,TCPIPC,START,OSA2080. Figure 9-13 shows the RACF error resulting when the user does not have CONTROL access to the specific profile.

```
V TCPIP,TCPIPC,START,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,START,OSA2080
IEE345I VARY     AUTHORITY INVALID, FAILED BY SECURITY PRODUCT
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          ) 712
  MVS.VARY.TCPIP.STRTSTOP CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(CONTROL)  ACCESS ALLOWED(NONE   )
EZZ0059I MVS.VARY.TCPIP.STRTSTOP COMMAND FAILED: NOT AUTHORIZED
```

Figure 9-13 RACF error from unauthorized use of TSO NETSTAT command - Specific profile

Now the user CS09 was given CONTROL access to the specific profile MVS.VARY.TCPIP.STRTSTOP, as shown in Example 9-5.

Example 9-5 Give CONTROL access to MVS.VARY.TCPIP.STRTSTOP

```
PE MVS.VARY.TCPIP.STRTSTOP CLASS(OPERCMDS) ID(CS09) ACCESS(CONTROL)
SETROPTS RACLIST(OPERCMDS) REFRESH
SETROPTS GENERIC(OPERCMDS) REFRESH
```

The V TCPIP,TCPIPC,STOP,OSA22E0 command completed successfully, as shown in Example 9-6.

Example 9-6 Successful START command

```
V TCPIP,TCPIPC,START,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,START,OSA2080
EZZ0053I COMMAND VARY START COMPLETED SUCCESSFULLY
```

9.6.2 TSO NETSTAT and UNIX onetstat command security

The TSO NETSTAT command and the UNIX shell **`onetstat`** command can be protected from unauthorized use at both the command level (NETSTAT with any option) and the command option level. By defining a SAF profile to represent the NETSTAT command and option, you can grant permission by user or group to the NETSTAT command and its options.

RACF profile details

The SERVAUTH class is used to define a profile to protect the NETSTAT command. The format of the profile name is:

`EZB.NETSTAT.`*`sysname.tcpprocname.option`*

Where:

- EZB.NETSTAT is constant.
- *sysname* is the z/OS image name.
- *tcpprocname* is the TCP/IP stack name.
- *option* is either an asterisk (*), meaning all options, or a specific NETSTAT option name.

As mentioned, you can protect NETSTAT/**`onetstat`** at the command level and the command option level.

Protecting NETSTAT/onetstat at the command level

To specify protection at the command level, specify the SERVAUTH profile with a command option of an asterisk (*) and define the SERVAUTH profile to be generic. Example 9-7 shows how this is done using RACF commands, assuming that the system name is SC30 and the TCP/IP stack name is TCPIPD.

Example 9-7 Protecting NETSTAT/onetstat at the command level

```
SETROPTS GENERIC(SERVAUTH)
RDEFINE SERVAUTH (EZB.NETSTAT.SC30.TCPIPD.*) UACC(NONE)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Note that the SETROPTS GENERIC(SERVAUTH) needs to be done only once.

Protecting NETSTAT/onetstat at the command option level

You may decide to protect the NETSTAT/**onetstat** command at a more granular level so that you can control who has authority to use the options of the NETSTAT/**onetstat** command. To protect the command options, define the particular NETSTAT option that you want to protect as the last qualifier in the profile name. Example 9-8 shows the NETSTAT HOME or **onetstat -h** command being protected for TCP/IP system TCPIPC on z/OS system SC30.

Example 9-8 Defining NETSTAT profile to protect NETSTAT/onetstat command with home option

```
RDEFINE SERVAUTH (EZB.NETSTAT.SC30.TCPIPD.HOME) UACC(NONE)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Restriction: The NETSTAT DROP command is internally implemented as a z/OS console command VARY TCPIP,,DROP, and as such is protected by the SAF profile MVS.VARY.TCPIP.DROP, not by a NETSTAT SAF profile.

SAF checking

The NETSTAT SAF check is performed whenever a TSO NETSTAT or UNIX **onetstat** command is attempted. A SAF check is performed against the most specific profile name first. If a profile for the specific command option does not exist, then a check is made against the generic profile (the profile with a command option specified as an asterisk (*)).

NETSTAT security scenario

Our system name at the ITSO is SC30 and the TCP/IP stack name is TCPIPD. Our tests were to use the TSO NETSTAT HOME command. For the first test, we defined the generic NETSTAT profile to protect all options of the NETSTAT command. For the second test, we defined the profile to protect the NETSTAT HOME command. Both times, unauthorized use of the command caused the expected RACF error.

In the discussions that follow, wherever the TSO NETSTAT command is mentioned, the OMVS **onetstat** command is also implied.

Defining the NETSTAT generic profile to protect all NETSTAT commands

The generic profile EZB.NETSTAT.SC30.TCPIPD.* was added to the SERVAUTH class with UACC(NONE), as shown in Example 9-7. At this stage, no user had any access to this profile. A user then attempted a TSO NETSTAT HOME command, which resulted in the RACF error shown in Example 9-9. Note that the profile being RACF checked was EZB.NETSTAT.SC30.TCPIPD.HOME, but because that did not exist, the profile EZB.NETSTAT.SC30.TCPIPD.* was checked.

Example 9-9 Access denied by generic profile

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
   EZB.NETSTAT.SC30.TCPIPD.HOME CL(SERVAUTH)
   INSUFFICIENT ACCESS AUTHORITY
   FROM EZB.NETSTAT.SC30.TCPIPD.* (G)
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
 Access to Netstat HOME denied - SAF RC is 00000008
```

Defining the NETSTAT profile to protect the NETSTAT HOME command

The specific profile EZB.NETSTAT.SC30.TCPIPD.HOME was added to the SERVAUTH class with UACC(NONE), as shown in Example 9-8 on page 326. At this stage, no user had any access to this profile. User CS09 then attempted a TSO NETSTAT HOME command, which resulted in the RACF error shown in Example 9-10.

Note that the profile that has been SAF checked is now EZB.NETSTAT.SC30.TCPIPD.HOME rather than EZB.NETSTAT.SC30.TCPIPD.*.

Example 9-10 RACF error from unauthorized use of TSO NETSTAT command - Generic profile

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
  EZB.NETSTAT.SC30.TCPIPD.HOME CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
Access to Netstat HOME denied - SAF RC is 00000008
```

The OMVS command equivalent of TSO NETSTAT HOME is **`onetstat -h`**. The same user who attempted the TSO NETSTAT command in Example 9-10 used the command **`onetstat -h -p tcpipc`** (the -p parameter targets the TCP/IP stack named TCPIPD) and got the expected error, as shown in Example 9-11.

Example 9-11 OMVS error from unauthorized use of the onetstat command

```
CS09 @ SC30:/u/cs09>onetstat -h -p tcpipd
EZZ2385I Access to Netstat -h denied - SAF RC is 00000008
CS09 @ SC30:/u/cs09>
```

9.6.3 Policy Agent command security

The z/OS UNIX **`pasearch`** command queries information from the Policy Agent (PAGENT). The policy Agent contains sensitive information about the policies that control the security of your network such as IP Security (IPSec), Application Transparent Transport Layer Security (AT-TLS), Intrusion Detection, VPN tunnels, and so on. This command should be restricted to those network and security administrators who need to know policy settings.

You can define the following RACF profile EZB.PAGENT.*sysname.tcpname.policy_type* in SERVAUTH class to individually protect each policy type, where:

- *sysname* is the z/OS SMFID.
- *tcpname* is the TCP/IP stack name, which is the name of the TCP/IP started task.
- *policy_type* is the policy type, which can be one of the following:
  - QOS for Policy QoS
  - IDS for Policy IDS
  - IPSec for Policy IPSec
  - TTLS -for Policy AT-TLS

The policy_type can also be a wildcard character (*) to protect all the policy types with a single profile.

9.6.4 IPSec command access control

The **ipsec** commands are sensitive and are used to display and monitor IP Security management activities. The RACF profiles in the SERVAUTH class required to protect these commands are:

- EZB.IPSECCMD.*sysname*.*tcpprocname*.DISPLAY
- EZB.IPSECCMD.*sysname*.*tcpprocname*.CONTROL

You can also define a single generic profile EZB.IPSECCMD.*sysname.tcpprocname*.* to control both these commands.

9.6.5 For more information

For more information about the profile names used to protect the various commands and options, see *z/OS V1R8.0 Communications Server: IP User's Guide and Commands*, SC31-8780.

9.7 Protecting FTP-related resources

In this section, we discuss protecting FTP-related resources.

9.7.1 FTP SITE command control

FTP commands SITE DUMP and DEBUG generate a large amount of output and their use should be restricted. The SERVAUTH class profiles that protect these resources are:

- EZB.FTP.*sysname.ftpdname*.SITE.DUMP
- EZB.FTP.*sysname.ftpdname*.SITE.DEBUG, where:
  - *sysname* is the z/OS SMFID.
  - *ftpdname* is the name of the FTP started task.

9.7.2 FTP server access control

To control the ability to access the FTP server based on SAF user ID associated with TLS-authenticated X.509 client certificate, you need to define the profile EZB.FTP.*sysname.ftpdname*.PORTxxxxx in the SERVAUTH class.

9.7.3 FTP z/OS UNIX access control

This provides the ability to protect z/OS UNIX access by FTP users. The profile name is of the following form: EZB.FTP.*sysname.ftpdaemonname*.ACCESS.HFS. For example, the profile name for FTP daemon FTPD running on system MVSA would be EZB.FTP.MVSA.FTPD1.ACCESS.HFS.

9.7.4 RACF-delegation of cryptographic resources

If you are securing connections using TLS/SSL for your z/OS FTP server, then you need to permit each FTP client to the sensitive cryptographic resources such as CFSERV and CFSKEYS. You can avoid having to permit each FTP client individually by identifying these resources as 'RACF DELEGATED'. The RACF command to do this is:

```
RALTER CSFSERV CSFENV APPLDATA('RACF DELEGATED')
```

The FTP daemon will now access these resources on behalf of the user, though the user does not have explicit access to these sensitive resources.

9.8 Protecting network management resources

In this section, we discuss protecting network management resources.

9.8.1 SNMP agent control

You can control which of the SNMP subagents are permitted to connect to the SNMP agent. You need to define a SERVAUTH class profile EZB.SNMPAGENT.*sysname.tcpname* for this. After creating the profile, use the RACF PERMIT command to define the user IDs of those subagents that should be permitted to connect via TCP to the SNMP Agent.

9.8.2 TCP connection information service access control

The TCP connection information service allows network management applications to obtain information about TCP connection activity. Access to this information can be controlled by RACF SERVAUTH class profile EZB.NETMGMT.*sysname*.*tcpname*.SYSTCPCN. You also need to permit the user IDs of the applications authorized to access this resource.

9.8.3 CIM provider access control

The Common Information Model (CIM) provides a model for describing and accessing data across an enterprise. CIM providers gather the CIM data. You can control this function and restrict the collection of CIM data to authorized providers by defining the SERVAUTH class profile EZB.CIMPROV.*sysname*.*tcpname*. Access is granted if the user ID associated with the client of the z/OS CIM server is permitted (has read access) to this resource profile.

9.9 Protecting miscellaneous resources

In this section, we discuss protecting miscellaneous resources.

9.9.1 Digital Certificate Access Server access control

This controls the ability to access the Digital Certificate Access Server (DCAS) based on the SAF user ID associated with the TLS-authenticated X.509 client certificate. The profile that protects this resource is:

```
EZB.DCAS.cvtsysname
```

9.9.2 MODDVIPA utility program control

This restricts the usage of the MODDVIPA utility program (creates new DVIPA on system). The profile that protects this resource is:

`EZB.MODDVIPA.`*`sysname`*`.`*`tcpname`*

9.9.3 Fast Response Cache Accelerator access control

This controls the ability to create a Fast Response Cache Accelerator (FRCA) cache. (FRCA is used by Web servers for caching static Web pages in the stack.) The profile that protects this resource is:

`EZB.FRCAACCESS.`*`sysname`*`.`*`tcpname`*

9.9.4 Real-time SMF information service access control

This restricts access to select real-time SMF records accessible using the SMF information service. It is intended for network management applications. The profile that protects this resource is:

`EZB.NETMGMT.`*`sysname`*`.`*`tcpname`*`.SYSTCPSM`

9.9.5 TCP/IP packet trace service access control

This restricts access to select real-time packet trace records accessible using the TCP/IP packet trace service. It is intended for network management applications. The profile that protects this resource is:

`EZB.NETMGMT.`*`sysname`*`.`*`tcpname`*`.SYSTCPDA`

9.9.6 TCP/IP stack initialization access control

This controls the ability of applications to open a socket before the AT-TLS policy is loaded into the TCP/IP stack. Normally you cannot start any application before the POLICY AGENT address space starts. The profile that protects this resource is:

`EZB.INITSTACK.`*`sysname`*`.`*`tcpname`*

Part 3

Appendixes

Given that security technologies are complex and can be confusing, we have included helpful tutorial information in this part of the book. It includes information about the following topics:

- “Basic cryptography” on page 333
- “Tools for application security” on page 351
- “Certificate management in z/OS” on page 369
- “IPSec scenario policies” on page 421

© Copyright IBM Corp. 2007. All rights reserved.

A

Basic cryptography

The word *cryptography* has its roots in Greek, and means *secret writing*. One of the earliest uses of cryptography was for protecting military communications. In ancient times, a human messenger would be dispatched with a military order. If that messenger was caught, the message could be read by the enemy. A method had to be used to hide the meaning of the message from an interceptor but still allow the intended recipient to understand it.

The message (plain text) to be conveyed has to be encrypted by some formula (the *cipher*). The cipher normally has, as its inputs, the message to be encrypted and a *key*. By using a key, the cipher itself can be public knowledge but the key is kept (hopefully) private between the communicating parties. The text that is produced by the cipher is the *ciphertext*. The decryption process takes the ciphertext, runs it through the decryption cipher with the key, and produces the plain text again.

Cryptography has more uses than ensuring privacy through encrypting a message. Other uses for cryptography are to provide message integrity through the use of encrypted message hashes, and non-repudiation so that a sender cannot deny having sent a particular message. To ensure privacy, integrity, and non-repudiation in non-secure networks, cryptographic procedures need to be used.

Today, two distinct classes of cryptographic algorithms are in use:

- Secret key (or symmetric key)
- Public key (or asymmetric key)

They are fundamentally different in how they work, and thus in where they are used.

These are the basic building blocks for securing transactions over the Internet or some other untrusted network.

© Copyright IBM Corp. 2007. All rights reserved.

This appendix discusses the topics listed in Table A-1.

Table A-1 Topics discussed in this appendix

Section	Topic
“Potential problems with electronic message exchange” on page 334	We look at an example to illustrate the security issues with electronic message exchanges.
“Secret key cryptography” on page 337	This section discusses the basic concepts surrounding secret keys and the algorithms used for those keys.
“Public key cryptography” on page 339	We describe the basic concepts surrounding public keys where public and private keys are used. This section also covers digital certificates along with their role in the secure use of public key cryptography.
“Performance issues of cryptosystems” on page 345	Performance is always a concern when doing cryptography. Here we briefly look at some of the issues surrounding performance.
“Message integrity” on page 346	This topic discusses how cryptography can aid in asserting message integrity (ensuring a message has not been altered in transit). It also discusses how digital signatures can prove that the message sender actually sent the message.

Potential problems with electronic message exchange

Let us take an example of an electronic message exchange for a stock broker. Clients log on to the system and send buy and sell requests for shares electronically to the broker. Potential security problems involved with these message exchanges include:

- “The request is not really from your client”.
- “The order could have been intercepted and read”.
- “The order could have been intercepted and altered”.
- “An order is received from your client, but he denies sending it”.

Now we discuss each of these problems and show what can be done to resolve each.

The request is not really from your client

Figure A-1 shows a hacker posing as a legitimate client (Garth).

Figure A-1 Hacker posing as another genuine client

What is needed here is some way to ensure that the client is who he says he is. In some cases, this must involve some sort of shared secret, such as a password. This is called user authentication. “Authentication” on page 340 explains how this is done.

The order could have been intercepted and read

Figure A-2 shows a hacker intercepting and reading an order that the client has placed.

Figure A-2 Hacker intercepting the order

Assume that you have some way of knowing for sure that the order you have received originated from your client. How do you know that the order has not been read by anyone other than the two parties involved? You cannot be sure how many computers and links it has been across, and you do not know whether any intermediate link in the network has cached the message or logged it in any way, so what can you do?

The sender must alter the message so that its meaning is hidden to unauthorized parties using a process known as encryption. “Encryption” on page 339 explains this technology.

The order could have been intercepted and altered

Figure A-3 shows the hacker altering the original message.

Figure A-3 Hacker intercepting and altering the order

How do you know, when you receive a message, that the contents of the message have not been modified. What is needed is some form of message authentication.

A message authentication process takes a message block, or stream, and mathematically summarizes the bits in the message to produce a fixed length message digest that represents that message. No two messages should produce the same message digest, or at least, it should be computationally infeasible to find two that do. This message digest is normally appended to the original message, and transmitted along with it, to the destination. If the original message is altered, when the receiver recalculates the message digest, it will differ from the one in the message. This does not guard against someone intercepting the message, altering it, recalculating the digest, and replacing the original digest and sending it along. That is why digests are often encrypted, which is then termed a message authentication code (MAC).

If the encryption process is by a private key (nobody else knows the private key), then the MAC becomes a digital signature. A digital signature proves that one party, and one party alone, could have originated a particular message. If the message was intercepted and altered, the decryption process will yield rubbish and the receiver will know that the message should be retransmitted. These are explained in “Message authentication codes” on page 347, and “Digital signatures” on page 348.

An order is received from your client, but he denies sending it

Figure A-4 shows a hacker placing a false order and the client then denying that he placed the order.

Figure A-4 Hacker placing an order

What is needed is some method where the sender of a message cannot deny having sent it. This requirement is called non-repudiation. This is done by making sure that the sender sends his request along with his unique digital signature. This is described in “Digital signatures” on page 348.

Secret key cryptography

Secret key cryptography is so called because the key used to encrypt the message must be kept secret from everyone but the two communicating parties. Ensuring a key is secret seems obvious but is not necessary in public key systems; this is described in “Public key cryptography” on page 339. Another name for secret key encryption is symmetric encryption, so called because the same key that is used to encrypt the data is also used to decrypt the data and recover the clear text, as shown in Figure A-5.

Figure A-5 Symmetric encryption and decryption: Using the same key

Symmetric algorithms are usually efficient in terms of processing power, therefore they are ideal for encryption of bulk data. However, they have one major drawback, which is key management. The sender and receiver on any secure connection must share the same key; in a large network where thousands of users may need to communicate securely, it is extremely difficult to manage the distribution of keys so as not to compromise the integrity of any one of them. Public key encryption, described in “Public key cryptography” on page 339, can be used to exchange secret keys securely, and from then onward, the conversation can use the faster secret key encryption.

Frequently used symmetric algorithms include:

DES Data Encryption Standard. Developed in the 1970s by IBM scientists, it uses a 56-bit key. Stronger versions called Triple-DES have been developed that use three operations in sequence: *2-key Triple DES* encrypts with key 1, decrypts with key 2, and encrypts again with key 1. The effective key length is 112 bits. *3-key Triple-DES* encrypts with key 1, decrypts with key 2, and encrypts again with key 3. The effective key length is 168 bits.

CDMF Commercial Data Masking Facility. This is a version of the DES algorithm approved for use outside the U.S. and Canada (in times when export control was an issue). It uses 56-bit keys, but 16 bits of the key are known, therefore the effective key length is 40 bits.

RC2 Developed by Ron Rivest for RSA Data Security, Inc., RC2 is a block cipher with variable key lengths operating on 8-byte blocks. Key lengths of 40 bits, 56 bits, 64 bits, and 128 bits are in use.

RC4 Developed by Ron Rivest for RSA Data Security, Inc., RC4 is a stream cipher operating on a bit stream. Key lengths of 40 bits, 56 bits, 64 bits, and 128 bits are in use. The RC4 algorithm always uses 128-bit keys; the shorter key lengths are achieved by “salting” the key with a known, non-secret random string.

AES As a result of a contest for a follow-on standard to DES held by the National Institute for Standards and Technology (NIST), the Rijndael algorithm was selected. This is a block cipher created by Joan Daemen and Vincent Rijmen with variable block length (up to 256 bits) and variable key length (up to 256 bits).

IDEA The International Data Encryption Algorithm was developed by James Massey and Xueija Lai at ETH in Zurich. It uses a 128-bit key and is faster than triple DES.

DES is probably the most scrutinized encryption algorithm in the world. Much work has been done to find ways to break DES, notably by Biham and Shamir, but also by others. However, a way to break DES with appreciably less effort than a brute-force attack (breaking the cipher by trying every possible key) has not been found.

Both RC2 and RC4 are proprietary, confidential algorithms that have never been published. They have been examined by a number of scientists under non-disclosure agreements.

With all the ciphers listed above, it can be assumed that a brute-force attack is the only means of breaking the cipher. Therefore, the work factor depends on the length of the key. If the key length is n bits, the work factor is proportional to $2^{(n-1)}$.

Today, a key length of 56 bits is generally only seen as sufficiently secure for applications that do not involve significant amounts of money or critically secret data. If specialized hardware is built (such as the machine built by John Gilmore and Paul Kocher for the Electronic Frontier Foundation), the time needed for a brute-force attack can be reduced to about 100 hours or

less (see *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*, by Electronic Frontier Foundation, John Gilmore (Editor), 1988). Key lengths of 112 bits and above are seen as unbreakable for many years to come, because the work factor rises exponentially with the size of the key.

Note: NIST has named Advanced Encryption Standard (AES) as the replacement for DES as the standard encryption algorithm. The Internet Engineering Task Force (IETF) IPSec Working Group intends for AES to eventually be adopted as the default IPSec Encapsulating Security Payload (ESP) cipher; therefore, AES must be included in compliant IPSec implementations.

Public key cryptography

Public key cryptography implements encryption and decryption using two different keys, which is why it is also termed *asymmetric encryption*. These two keys are known as a public key and a private key.

The beauty of asymmetric algorithms is that they are not subject to the key management issues that beset symmetric algorithms. Your public key is freely available to anyone, and if someone wants to send you a message he or she encrypts it using that key. Only you can understand the message, because only you have the private key.

Important: Public and private keys, if implemented in a reversible scheme such as RSA, (described in the following section) yield extremely important properties:

- If the public key is used to encrypt the data, the private key must be used to recover the clear text.
- If the private key is used to encrypt the data, the public key must be used to recover the clear text.

Encryption

Figure A-6 shows an exchange where one party (on the left) uses the second party's public key to encrypt a message.

Figure A-6 Public-key cryptography: Encryption using a public key

The ciphertext created by this encryption process is only decipherable by using the private key, which in turn is only known by the second party. There is no way for any other party to decipher this message. This type of encryption is used when you want the receiver to be the only person capable of understanding the message. This message flow can also be used to securely exchange a secret key between the conversation partners so that the faster secret key encryption can be used instead of public key.

Authentication

Asymmetric keys are also very useful for authentication. Look at Figure A-7. What happens if you encrypt a message using your own private key?

Figure A-7 Public-key cryptography: Encryption using a private key results in authentication

As stated earlier, this indicates that anybody with access to your public key (and that should be anyone) would be able to decipher the message. This type of encryption, therefore, is obviously of no use to hide a message. By encrypting a message with your private key, *a receiver must use your public key to decipher it*, and that is the point. This proves that the message could *only* have come from you.

Public key algorithms

Asymmetric encryption algorithms, commonly called Public Key Cryptography Standards (PKCS), are based on mathematical algorithms. The basic idea is to find a mathematical problem that is very hard to solve. The algorithm in most widespread use today is RSA. However, some companies have begun to implement public-key cryptosystems based on so-called *elliptic curve* algorithms. With the growing proliferation of IPSec, the Diffie-Hellman algorithm is gaining popularity. A brief overview of all three methods follows:

RSA Invented in 1977 by Rivest, Shamir, and Adleman (who formed RSA Data Security, Inc.). The idea behind RSA is that integer factorization of very large numbers is extremely hard to do. Key lengths of public and private keys are typically 512 bits, 768 bits, 1024 bits, or 2048 bits. The work factor for RSA with respect to key length is sub-exponential, which means that the effort does not rise exponentially with the number of key bits. It is roughly $2^{(0.3*n)}$.

Elliptic Curve Public-key cryptosystems based on elliptic curves use a variation of the mathematical problem of finding discrete logarithms. It has been stated that an elliptic curve cryptosystem implemented over a 160-bit field has roughly

the same resistance to attack as RSA with a 1024-bit key length. Properly chosen elliptic curve cryptosystems have an exponential work factor (which explains why the key length is so much smaller). Elliptic curve cryptosystems are now standardized by FIPS PUB 186-2, the digital signature standard (January 2000).

Diffie-Hellman W. Diffie and M.E. Hellman, the inventors of public key cryptography, published this algorithm in 1976. The mathematical problem behind Diffie-Hellman is computing a discrete logarithm. Both parties have a public-private key pair each; they are collectively generating a key only known to them. Each party uses its own private key and the public key of the other party in the key generation process. Diffie-Hellman public keys are often called *shares*.

Digital certificates

Digital certificates are used to publish a public key with a certainty that the public key is genuine, according to the Certificate Authority (CA) that digitally signs the certificate. First we discuss what can happen when a public key is used for communication, and that key is not genuine. We then cover what can be done about authenticating a public key by using digital certificates.

How can I trust a published public key

If we want to communicate with XYZ Corporation, and we have found a public key published on the Internet, how can we use that public key? The two uses of another person's or entity's public key are:

- To decrypt a message originating from that person, who has encrypted with his private key
- To encrypt a message to be sent to that person so that only he can decrypt it with his private key

As already mentioned, we have found XYZ Corporation's public key on the Internet. How do we know it is genuine? A malicious third party could have put his own public key on the Internet and now can intercept all communications from you to XYZ Corporation, acting as a sort of "relay" on the way.

Figure A-8 Scenario where public key being used by good guy is really a hacker's key

In Figure A-8, the "good guy" assumes that he has obtained a public key for XYZ Corporation from the Internet, but in reality, it was a hacker's public key. We further assume that the hacker has some way of removing your messages from the network, and injecting his own. The last assumption is that we are using XYZ's public key (or at least we think we are) to encrypt messages to XYZ Corporation and the response will be encrypted by XYZ Corporation with its private key.

You can see what can happen when a public key is used for communication when you do not know if it is genuine. Your messages to XYZ Corporation will be encrypted using the hacker's public key, because you thought it was XYZ Corporation's public key. The hacker then uses his private key to decrypt the message, make any changes he feels is necessary, and then encrypt the message with XYZ Corporation's real public key. When XYZ Corporation receives the message, it will decrypt using its private key, process the message, and send a message back, encrypting with its own private key. The hacker then receives the response and decrypts using XYZ Corporation's public key, makes more changes, if necessary, and encrypts with his own (hacker's) private key. Lastly, you receive the hacker's message, decrypt with what you think is XYZ Corporation's public key, and now your communication to XYZ has been totally compromised.

The problem of securely storing and retrieving public keys is dealt with by what is known as a *Public Key Infrastructure* (*PKI*), discussed in the following section. A good reference work on PKI can be found in the IBM Redbook *Deploying a Public Key Infrastructure*, SG24-5512, at:

http://www.redbooks.ibm.com

Public Key Infrastructure

A PKI offers the basis for practical usage of public key cryptography. A PKI defines the rules and relationships for certificates and Certificate Authorities (CAs). It defines the fields that can or must be in a certificate, the requirements and constraints for a CA in issuing certificates, and how certificate revocation is handled.

When using a PKI, the user must be confident that the obtained public key belongs to the correct remote person (or system) with which the digital signature mechanism is to be used. This confidence is obtained through the use of public key digital certificates. A digital certificate is analogous to a passport: The passport certifies the bearer's identity, address, and citizenship. The concepts behind passports and other identification documents (for instance, drivers' licenses) are very similar to those that are used for digital certificates.

Passports are issued by a trusted authority, such as a government passport office. A passport will not be issued unless the persons who request it have proven their identity and citizenship to the authority. Specialized equipment is used in the creation of passports to make it very difficult to alter the information in it or to forge a passport altogether. Other authorities, for instance, the border police in other countries, can verify a passport's authenticity. If they trust the authority that issued the document, they implicitly trust the passport.

A digital certificate serves two purposes: It establishes the owner's identity and it makes the owner's public key available. Similar to a passport, a certificate must be issued by a trusted authority, the CA, and, like a passport, it is issued only for a limited time. When its expiration date has passed, it must be replaced.

Trust is a very important concept in passports, as well as in digital certificates. In the same way as, for instance, a passport issued by the governments of some countries, even if recognized to be authentic, will probably not be trusted by the government authorities of another country. Each organization or user has to determine whether a CA can be accepted as trustworthy.

As an example, a company might want to issue digital certificates for its own employees from its own CA. This can ensure that only authorized employees are issued certificates, as opposed to certificates being obtained from other sources such as a commercial entity such as VeriSign.

The information about the certificate owner's identity is stored in a format that follows RFC 2253 and the X.520 recommendation, for instance, CN=Ulrich Boche, O=IBM Corporation. The complete information is called the owner's distinguished name (DN). The owner's distinguished name and public key and the CA's distinguished name are digitally signed by the CA. That is, a message digest is calculated from the distinguished names and the public key. This message digest is encrypted with the private key of the CA.

Figure A-9 shows a simplified layout of a digital certificate.

Figure A-9 Simplified layout of a digital certificate

The digital signature of the CA serves the same purpose as the special measures taken for the security of passports, such as laminating pages with plastic material. It allows others to verify the authenticity of the certificate. Using the public key of the CA, the message digest can be decrypted. The message digest can be recreated. If it is identical to the decrypted message digest, the certificate is authentic.

Security considerations for certificates

If I send my certificate with my public key in it to someone else, what keeps this person from misusing my certificate and posing as myself? The answer is: My private key.

A certificate alone can never be proof of anyone's identity. The certificate just allows the identity of the certificate owner to be verified by providing the public key that is needed to check the certificate owner's digital signature. Therefore, the certificate owner must protect the private key that matches the public key in the certificate. If the private key is stolen, the thief can pose as the legitimate owner of the certificate. Without the private key, a certificate cannot be misused.

An application that authenticates the owner of a certificate cannot accept just the certificate. A message signed by the certificate owner should accompany the certificate. This message should use elements such as sequence numbers, time stamps, challenge-response protocols, or other data that allow the authenticating application to verify that the message is a "fresh" signature from the certificate owner and not a replayed message from an impostor.

Certificate Authorities and trust hierarchies

Before we discuss what is termed a *certificate hierarchy*, let us look at an analogous example of trusted hierarchies. If you were selling a car, and a buyer asked you if it was okay to pay by personal check, then you have to decide whether you trust the buyer. If you do, end of story, the car is sold. If you do not trust him, you may ask someone whom you both trust to countersign the check.

In digital certificate trust hierarchies, similar considerations to the car-buying example apply. In the end, it boils down to the fact that you have to trust somebody. You will have digital certificates in your database, and those certificates will either be set to *trusted* or *untrusted* status. A CA is a company that is considered trustworthy, and produces digital certificates for

other individuals and companies (called *subjects*) bearing that subject's public key. This certificate is signed with a message hash that is encrypted using the CA's private key. To verify that the certificate is authentic, the receiver needs the public key of the CA that issued the certificate.

Most Web browsers come preconfigured with the public keys of common CAs (such as VeriSign). However, if the user does not have the public key of the CA that signed the certificate, an additional certificate would be needed in order to obtain that public key. In general, a chain of multiple certificates may be required, comprising a certificate of the public key owner signed by a CA, and possibly additional certificates of CAs signed by other CAs. Many applications that send a subject's certificate to a receiver send not only just that certificate, but also all the CA certificates necessary to verify the certificate up to the root.

Obtaining and storing certificates

As we have discussed, certificates are issued by a CA. If you do not want to use a CA, you can use utilities to issue your own certificates. These are called *self-signed* certificates and will only be accepted by people who trust you. If you use an external CA, you request certificates by visiting the CA's Web site. After verifying the validity of the request, the CA sends back the certificate in an e-mail message or allows it to be downloaded.

In the case of obtaining a certificate for a server, whether you use a self-signed or external CA signed certificate is dependent on the server environment. In an intranet environment, it is generally appropriate to use self-signed certificates. In an environment where external users are accessing the server over the Internet, it is usually advisable to acquire a server certificate from a well-known CA, because the steps needed to import a self-signed certificate might seem obscure, and most users will not have the ability to discern whether the action they are performing is of trivial consequence. It should also be noted that a root CA certificate received over an untrusted channel, such as the Internet, does not deserve any kind of trust.

Certificate management in z/OS

To manage certificates on a z/OS system, you can use either the UNIX program **gskkyman** to create and manage certificates, or you can use the RACF database and RACDCERT command. This topic is covered in detail in Appendix C, "Certificate management in z/OS" on page 369.

Performance issues of cryptosystems

Elliptic curve cryptosystems are said to have performance advantages over RSA in decryption and signing. While the possible differences in performance between the asymmetric algorithms are somewhere in the range of a factor of 10, the performance differential between symmetric and asymmetric cryptosystems is far more dramatic.

For instance, it takes about 1000 times as long to encrypt the same data with RSA (an asymmetric algorithm) as it takes with DES (a symmetric algorithm), and implementing both algorithms in hardware does not change the odds in favor of RSA.

As a consequence of these performance issues, the encryption of bulk data is usually performed using a symmetric cryptosystem, while asymmetric cryptosystems are used for electronic signatures and in the exchange of key material for secret-key cryptosystems. With these applications, only relatively small amounts of data need to be encrypted and decrypted, and the performance issues of public key systems are less important.

Message integrity

Message integrity is the ability to assert that a message received has not been altered in any way from the time that it was sent. In a networked environment, a message could have been altered by a third party intercepting it, or by some other means, such as electromagnetic interference (although in the latter case the transmission protocol normally handles a retransmission). To provide message integrity, you provide a message digest along with the text of your message. Note that the message being authenticated may or may not also be encrypted.

Message digest (or hash)

A message digest algorithm takes a message as input, and produces a small, fixed length *digest* string (usually 128 bits or 160 bits) often referred to as a *hash*. This hash can be thought of as a mathematical summary of a message. There are two important things to note about a message digest algorithm:

- The algorithm is a *one-way* function. This means that there is absolutely no way you can recover a message, given the hash of that message.
- It should be computationally infeasible to produce another message that would produce the same message digest as another message.

Figure A-10 is a graphical representation of appending a message digest to a message. When a message digest is appended to a message en route to its destination, the message cannot be tampered with, because a recalculation of the hash at the receiver's end will show the message digest received is invalid.

Figure A-10 Message digest

The message digest should not be sent in the clear: Because the digest algorithms are well-known and no key is involved, a man-in-the-middle could not only forge the message but also replace the message digest with that of the forged message. This will make it impossible for the receiver to detect the forgery. The solution for this is to use a message digest algorithm that uses cryptography when creating the message digest, that is, to use a message authentication code, described in "Message authentication codes" on page 347.

Message digest algorithms

Common message digest algorithms are:

MD2 Developed by Ron Rivest of RSA Data Security, Inc., this algorithm is mostly used for Privacy Enhanced Mail (PEM) certificates. MD2 is fully described in RFC 1319. Since weaknesses have been discovered in MD2, its use is discouraged.

MD5 Developed in 1991 by Ron Rivest, the MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest of the input. The MD5 message digest algorithm is specified in RFC 1321, The MD5 Message-Digest Algorithm. Collisions have been found in MD5 (see *Cryptanalysis of MD5 Compress*, by Hans Dobbertin available at: `http://www.cs.ucsd.edu/users/bsy/dobbertin.ps`).

SHA-1 Developed by the National Security Agency (NSA) of the U.S. Government, this algorithm takes as input a message of arbitrary length and produces as output a 160-bit hash of the input. SHA-1 is fully described in standard FIPS PUB 180-1, also called the Secure Hash Standard (SHS). SHA-1 is generally recognized as the strongest and most secure message digesting algorithm.

SHA-256, SHA-512 Developed by the NSA of the U.S. Government. The security of a hash algorithm against collision attacks is half the hash size, and this value should correspond with the key size of encryption algorithms used in applications together with the message digest. Because SHA-1 only provides 80 bits of security against collision attacks, this is deemed inappropriate for the key lengths of up to 256 bits planned to be used with AES. Therefore, extensions to the SHS have been developed. SHA-256 provides a hash size of 256 bits, while SHA-512 provides a hash size of 512 bits.

Message authentication codes

Figure A-11 shows a message authentication code (MAC) being created for a message. The first step is to use a hashing algorithm, such as MD5, to compute a message digest. That message digest is then encrypted with a key, and appended to the original message. Both the message and the associated MAC are then sent to the recipient. The assumption here is that the recipient shares the same key so that he may recompute the message digest and encrypt it with the shared key. This result should match the MAC sent on the message.

Figure A-11 Message digest for data integrity

Secret-key cryptographic algorithms, such as DES, can be used for encryption with message digests. A disadvantage of using a secret-key algorithm is that because the receiver has the key that is used in MAC creation, this system does not offer a guarantee of non-repudiation. That is, it is theoretically possible for the receiver to forge a message and claim it was sent by the sender. Therefore, message authentication codes are usually based on public key/private key encryption in order to provide for non-repudiation. When a MAC is encrypted with a sender's private key, rather than a secret (symmetric) key, that MAC becomes a *Digital Signature*. This is discussed further in "Digital signatures".

Keyed hashing for message authentication (HMAC)

H. Krawczyk and R. Canetti of IBM Research and M. Bellare of UCSD invented a method to create a message authentication code called HMAC, which is defined in RFC 2104 as a proposed Internet standard. A simplified description of how to create the HMAC is as follows. The key and the data are concatenated and a message digest is created. The key and this message digest are again concatenated for better security, and another message digest is created, which is the HMAC.

HMAC can be used with any cryptographic hash function. Typically, either MD5 or SHA-1 is used. In the case of MD5, a key length of 128 bits is used (the block length of the hash algorithm). With SHA-1, 160-bit keys are used. Using HMAC actually improves the security of the underlying hash algorithm. For instance, some collisions (different texts that result in the same message digest) have been found in MD5. However, they cannot be exploited with HMAC; therefore, the weakness in MD5 does not affect the security of HMAC-MD5.

HMAC is now a PKCS#1 V.2 standard for RSA encryption (proposed by RSA, Inc., after weaknesses were found in PKCS#1 applications). For further details, see:

http://www.ietf.org/rfc.html

HMAC is also used in the Transport Layer Security (TLS) protocol, the successor to SSL.

Digital signatures

Digital signatures are an additional means of securing data integrity. While data integrity only ensures that the data received is identical to the data sent, digital signatures go a step further: They provide non-repudiation. This means that the sender of a message (or the signer of a document) cannot deny authorship, similar to signatures on paper. As illustrated in Figure A-12, the creator of a message or electronic document that is to be signed uses a message digesting algorithm such as MD5 or SHA-1 to create a message digest from the data. The message digest and some information that identifies the sender are then encrypted with an asymmetric algorithm using the sender's private key. This encrypted information is sent together with the data.

Figure A-12 Digital signature creation

The receiver, as shown in Figure A-13, uses the sender's public key to decrypt the message digest received. Then he or she will use the message digesting algorithm to compute the message digest from the data received. If the computed message digest is identical to the one recovered after decrypting the digital signature, the signature is recognized as valid proof of the authenticity of the message.

Figure A-13 Digital signature verification

With digital signatures, only public-key cryptosystems can be used. If secret-key cryptosystems are used to encrypt the signature, it will be very difficult to make sure that the receiver (having the key to decrypt the signature) cannot misuse this key to forge a signature of the sender. The private key of the sender is known to nobody else, therefore nobody is able to forge the sender's signature.

Note the difference between encryption using public-key cryptosystems and digital signatures:

- With encryption, the sender uses the receiver's public key to encrypt the data, and the receiver decrypts the data with his private key. This means everybody can send encrypted data to the receiver that only the receiver can decrypt. See Figure A-14 for a graphical representation.

Figure A-14 Encrypting data with the receiver's public key

- With digital signatures, the sender uses his private key to encrypt his signature, and the receiver decrypts the signature with the sender's public key. This means that only the sender can encrypt the signature, but everybody who receives the signature can decrypt and verify it.

The tricky part with digital signatures is the trustworthy distribution of public keys, because a genuine copy of the sender's public key is required by the receiver. A solution to this problem is provided by digital certificates, which we discussed in "Digital certificates" on page 341.

B

Tools for application security

This appendix discusses ways to secure application traffic. You will find that each of these protocols is used by many applications. For instance, Secure Sockets Layer (SSL) is used by TN3270, File Transfer Protocol (FTP), Policy Agent (PAGENT), Lightweight Directory Access Protocol (LDAP), and so on. Each protocol is explained in as much detail as needed to understand the function, and references are given for more advanced study.

SSL and Transport Layer Security (TLS) use public key cryptography to establish a secret key, which is then used for secret key (or symmetric) cryptography. These protocols require digital certificates for the server, and optionally for the client. For information about the SSL and TLS protocols, see “Secure Sockets Layer” on page 352.

For a brief overview of the differences between SSL and TLS, see “TLS protocol” on page 357.

The Kerberos system is a secret key system that uses symmetric keys, one at the client and another at what is known as a Key Distribution Center (KDC). z/OS applications that can make use of Kerberos include FTP (server and client), UNIX Telnet, and UNIX rsh. See “Kerberos-based security system” on page 358.

© Copyright IBM Corp. 2007. All rights reserved.

Secure Sockets Layer

The first version of the Secure Sockets Layer protocol was developed by Netscape Communications Corporation in 1994 to enable secure Web transactions. Since then, the SSL protocol has been widely deployed to protect traffic for a number of different applications. In 1996, Netscape Communications handed the responsibility for SSL over to the Internet Engineering Task Force (IETF), who enhanced the protocol and released it as TLS V1.0. TLS is discussed in “TLS protocol” on page 357. Figure B-1 shows the evolution of SSL.

Figure B-1 Evolution of SSL

SSL-enabled applications on z/OS include:

- IBM HTTP Server for z/OS
- TN3270 Server
- z/OS LDAP server
- z/OS Firewall Configuration Server
- IBM CICS® Web Interface
- z/OS UNIX policy agent
- Digital Certificate Access Server (DCAS) used in the Express Logon Feature

TLS-enabled applications on z/OS include FTP server and client.

SSL relies on digital certificates and a hierarchy of trusted authorities, as described in “Digital certificates” on page 341, to ensure authentication of clients or servers.

SSL protocol description

The SSL protocol defines the partners of a conversation as either a *client* or a *server*. This terminology is used because a client must send certain sets of messages and the server responds with another set. The SSL protocol begins with a *handshake* initiated by the client. During the handshake, the client authenticates the server, the server optionally authenticates the client, and the client and server agree on encryption and authentication algorithms. See Figure B-2.

Figure B-2 Overview of SSL handshake protocol

The process is:

1. First, the client sends a `client hello` message, which lists the cryptographic capabilities of the client (sorted in client preference order) and contains the SSL/TLS protocol version desired. It also contains a random number used later to generate a secret key by both server and client, a session ID (used for resumed sessions, not discussed here), and a list of cipher suites that the client can support. A cipher suite is an entry indicating an encryption algorithm and a message hashing algorithm.
2. The server responds with a `server hello` message, which contains the cipher suite selected by the server, the session ID, another random number, and the acceptable SSL/TLS protocol version. The client and server must support at least one common cipher suite or the handshake will fail.
3. Following the `server hello` message, the server sends its certificate. This message contains the server's digital certificate and all other certificates up to the *root*. The whole chain of certificates is included because the client must match the issuers of the certificates all the way up to the root certificate to find a match with an issuer that it trusts. In a z/OS system server, the certificate is obtained from either a key ring database stored in a z/OS UNIX or MVS data set (which is created with the **`gskkyman`** utility) or from the RACF database using the DIGTCERT class.
4. If SSL version 3 or later (TLS) is used and the server application requires a certificate for client authentication, the server sends a `certificate request` message. In the certificate request message, the server sends a list of the types of certificates supported and the distinguished names of acceptable certification authorities.
5. The server then sends a `server hello done` message and waits for a client response. Upon receipt of the `server hello done` message, the client verifies the validity of the server's certificate and checks that the server hello parameters are acceptable.
6. If the server requested a client certificate, the client sends a certificate or, if no suitable certificate is available, a `no certificate` alert. This alert is only a warning, but the server application can fail the session if client authentication is mandatory. If a certificate is available, this message contains the client's digital certificate and all other certificates up to the *root*. The whole chain of certificates is included because the server must match the issuers of the certificates all the way up to the root certificate to find a match with an issuer that it trusts.

7. The client then sends a `client key exchange` message. This message contains the so-called pre-master secret, a 46-byte random number that is used in the generation of the symmetric encryption keys and the message authentication code (MAC) keys, encrypted with the public key of the server.
8. If the client sent a certificate to the server, the client will now send a `certificate verify` message, which is signed with the client's private key. By verifying the signature of this message, the server can explicitly verify the ownership of the client certificate.

   A similar process to verify the server certificate is not necessary. If the server does not have the private key that belongs to the certificate, it cannot decrypt the pre-master secret nor create the correct keys for the symmetric encryption algorithm, and the handshake must fail.
9. Now the client uses a series of cryptographic operations to convert the pre-master secret into a master secret, from which all key material required for encryption and message authentication is derived. Then the client sends a `change cipher spec` message to make the server switch to the newly negotiated cipher suite.
10. The `finished` message that immediately follows is the first message encrypted with this cipher method and keys.
11. After the server responds with a `change cipher spec` and a `finished` message of its own, the SSL handshake is completed and encrypted application data can be sent.

The SSL Record Protocol transfers application data using the encryption algorithm and keys agreed upon during the handshake phase. As explained in "Performance issues of cryptosystems" on page 345, symmetric encryption algorithms are used, because they provide much better performance than asymmetric algorithms.

Certificates for SSL

To conduct commercial business on the Internet, you might use a widely known Certificate Authority (CA), such as VeriSign, to get a high assurance server certificate. For a relatively small private network within your own enterprise or group, you can issue your own server certificates, called self-signed certificates, using the z/OS UNIX **`gskkyman`** utility or the RACF RACDCERT command.

In SSL, servers are always authenticated by the client. This means that the client must have access to a CA certificate that can verify the server's certificate.

Client authentication, which is optional, provides additional authentication and access control by checking client certificates at the server. This support prevents a client from obtaining a connection without an installation approved certificate. There are three levels of client authentication:

- Level 1

  The authentication is performed by system SSL and ensures that the server's key ring contains a CA certificate that can verify the client certificate. For information about digital certificates, see "Digital certificates" on page 341.
- Level 2

  The authentication provides, in addition to level 1 support, that the client certificate be registered with RACF (or another SAF-compliant security product) and mapped to a RACF user ID in the RACF database.

- Level 3

  The authentication provides, in addition to level 1 and level 2 support, the capability to restrict access to a server (and port number) based on a profile that can be set up in RACF. The user ID that is associated with the client certificate is tested for access rights to the server and port represented by the RACF profile.

Figure B-3 shows an example of a z/OS server (TN3270 in this case) using SSL.

Figure B-3 SSL protocol

The client must verify the server's certificate based on the certificate of the CA that signed the certificate or based on a self-signed certificate from the server. The server must verify the client's certificate (if client authentication has been configured in the server) using the certificate of the CA that signed the client's certificate. The client and the server then use the negotiated session keys and begin encrypted communications.

A program may require a certificate associated with itself depending on what side of the SSL connection the program is running. This requirement also depends on whether client authentication is requested as part of the SSL handshake. Programs acting as SSL servers (act as the server side of the SSL handshake protocol) must have a certificate to use during the handshake protocol. A program acting as an SSL client requires a certificate in the key database if the SSL server requests client authentication as part of the SSL handshake operation.

If the organization chooses to use a CA (within the organization or outside of the organization), then you must generate a certificate request. If only self-signed server certificates are used, you do not have to formulate a certificate request to be sent to an external CA for approval. However, in this case SSL clients do have to import the server's self-signed certificate so that it can be verified during SSL handshake processing.

Additional information about the concepts of cryptography and SSL can be found at the following Web site:

http://www.verisign.com/repository/crptintr.html

Refer to Appendix C, “Certificate management in z/OS” on page 369, for steps regarding the creation of certificates with **gskkyman** and RACF.

System SSL

System SSL is a common set of libraries for use by clients and servers in a z/OS system. An Application Programming Interface (API) is provided by System SSL in order to use the SSL code library.

System SSL is part of the System SSL Cryptographic Services Base element of z/OS. The z/OS Communications Server uses the System SSL APIs to create and manage SSL connections. X.509 certificates are used by both the client and server when securing communications using System SSL.

System SSL supports the following two methods for managing Public Key Infrastructure (PKI) private keys and digital certificates:

- A z/OS shell-based program called **gskkyman**. **gskkyman** creates, fills in, and manages a z/OS UNIX file that contains PKI private keys, certificate requests, and certificates. This z/OS UNIX file is called a *key database* and, by convention, has a file extension of *.kdb*.
- The IBM z/OS SecureWay® Security Server (RACF) RACDCERT command. RACDCERT installs and maintains PKI private keys and certificates in RACF. RACF supports multiple PKI private keys and certificates to be managed as a group. These groups are called *key rings*. RACF key rings are the preferred method for managing PKI private keys and certificates for System SSL.

Table B-1 shows the encryption capabilities of each of the z/OS V1R8 System SSL FMIDs.

Table B-1 SSL encryption capabilities

Encryption type/key sizes	Base security level FMID HCPT320	Security level 3 FMID JCPT321
512-bit keys	X	X
1024-bit keys	X	X
1 - SSL V2.0 RC4 US		X
2 - SSL V2.0 RC4 Export	X	X
3 - SSL V2.0 RC2 US		X
4 - SSL V2.0 RC2 Export	X	X
6 - SSL V2.0 DES 56-Bit	X	X
7 - SSL V2.0 Triple DES US		X
01 - SSL V3.0 NULL MD5	X	X
02 - SSL V3.0 NULL SHA	X	X
03 - SSL V3.0 RC4 MD5 Export	X	X
04 - SSL V3.0 RC4 MD5 US		X
05 - SSL V3.0 RC4 SHA US		X
06 - SSL V3.0 RC2 MD5 Export	X	X
09 - SSL V3.0 DES SHA Export	X	X
0A - SSL V3.0 Triple DES SHA US		X

Note: The encryption level used in an SSL connection depends on the client and server encryption level capability. In the SSL handshake, after server or client authentication, both server and client exchange their cipher capabilities and agree on the best cipher algorithm for the session. Therefore, be aware that your TN3270 client must support at least the same level of encryption as your server to have the level of encryption that you want.

System SSL supports both the TLS and SSL protocols.

To implement SSL connections, TCP/IP must have APF-authorized access to the System SSL DLLs. The System SSL DLLs are located in SYS1.SIEALNKE by default. System SSL uses the C runtime library (SCEERUN) and the C/C++ IBM Open class library (SCLBDLL), which must also be accessible to TCP/IP. To access these libraries, either add them to the linklist or specify them in the TCP procedure's STEPLIB. If accessed via the linklist, the linklist must be authorized (LNKAUTH=LNKLST specified in the IEASYSxx parmlib member) or the libraries explicitly APF authorized. If accessed via a STEPLIB, the libraries must be APF authorized and DISP=SHR specified.

SSL considerations

As discussed, security functions such as SSL are needed to send sensitive data safely if you connect your system to an insecure network such as the Internet. On the other hand, using such security functions has performance impacts, including utilizing additional CPU cycles and degrading server performance.

To maintain SSL security you have to manage the keys carefully, especially when using self-certification, because the whole system environment is affected by the security of the CA's key database. On z/OS, the key database or key ring file, including the server key pair, may be stored in a file in the z/OS UNIX file system if you use the **gskkyman** utility to manage certificates and keys. In this case, the file may be accessible by users of the z/OS UNIX shell unless you are very careful about setting the UNIX file permission bits on the z/OS UNIX files and you do not allow users to enter the superuser state. However, RACF is a more secure environment to store certificates and keys, and should be used if possible.

TLS protocol

SSL 3.0 has outgrown the scope of being a Netscape standard. Continued development of the protocol became the responsibility of the Internet Engineering Task Force in 1996. As a result, SSL 3.0 evolved into the proposed standard for Transport Layer Security, RFC 2246.

TLS is the latest in the continuing evolution of SSL. TLS 1.0 might as readily have been titled SSL 3.1. In fact, when negotiating a TLS handshake, the client and server hello messages will use version specification 3.1 (SSL 3.0 uses version specification 3.0).

Enhancements from SSL V3.0 to TLS V1.0 include:

- Additions to the number of *alert* messages defined in the protocol
- Standardized method of calculating message authentication codes (MAC)
- Simplified CertificateCertify message
- Simplified Finished message

Kerberos-based security system

Kerberos is a network authentication protocol that was developed in Project Athena at the Massachusetts Institute of Technology, in cooperation with IBM and Digital Equipment Corporation in the 1980s. DES cryptography is used to provide data privacy, especially for sensitive data such as passwords to log into a server.

Kerberos version 5 is the latest release and has been implemented in SecureWay Security Server Network Authentication and Privacy Service for z/OS, and chosen by Microsoft Corporation as their preferred authentication technology in Windows 2000.

The Kerberos system is an encryption-based security system that provides mutual authentication between the users and the servers in a network environment. The assumed goals for this system are:

- Authentication to prevent fraudulent requests and responses between users and servers that must be confidential and on groups of at least one user and one server.
- Authorization can be implemented independently from the authentication by each service that wants to provide its own authorization system. The authorization system can assume that the authentication of a user/client is reliable.
- Message confidentiality may also be used that provides assurance to a data sender that the message's content is protected from access by entities other than the context's named peer.

Kerberos authentication is based on shared secrets, which are passwords stored on the Kerberos server and client. Those passwords are encrypted with a symmetric cryptographic algorithm, which is DES in this case, and decrypted when needed. This fact implies that a decrypted password is accessed by the Kerberos server, which is not usually required in an authentication system that exploits public key cryptography. Therefore, the servers must be placed in locked rooms that are physically secure to prevent an attacker from stealing a password.

For the complete description about the Kerberos Version 5 protocol, refer to RFC 1510 - The Kerberos Network Authentication Service (V5).

Kerberos protocol overview

The Kerberos system consists of three components: A client, a server, and a trusted third party, which is also known as a *Key Distribution Center* (*KDC*). KDC interacts with both a client and a server to accept the client's request, authenticate its identity, and issue tickets to it.

The domain served by a single KDC is referred to as a *realm*. A *principal identifier* is used to identify each client and server in a realm. The principal name is uniquely assigned for all clients and servers by the Kerberos administrator. All principals must be known to the KDC.

Although the Kerberos protocol consists of several subprotocols, three particular exchanges provide the fundamental foundation (see Figure B-4 on page 359). The first phase exchange takes place between a client and the authentication server (AS), in which a client asks the AS that knows secret keys of all clients in the realm to authenticate himself and give it a *ticket-granting ticket* (TGT) to be used to get a service ticket for an application server it wants to access.

Upon receiving the TGT, the client sends a request, which contains the TGT, for a service ticket to the ticket-granting server (TGS), and waits until a service ticket is returned. Having the session ticket ready, the client can then communicate with the server that is providing a service he wants to use. Optionally, the application server can perform further authentication processes against the client.

Note: In most Kerberos implementations, the AS and the TGS are the same server.

Figure B-4 Kerberos protocol overview

Message encoding defined in Kerberos version 5 is described using the Abstract Syntax Notation 1 (ASN.1) syntax in accordance with ISO standards 8824 and 8825.

In the following sections, we discuss the interactions in more detail using the following notations:

- K_x: X's symmetric encryption key
- $K_{x,y}$: Encryption key shared by X and Y (for example, a session key)
- K_x{data}: A message that contains data encrypted with X's key

Phase 1: Authentication service exchange

The authentication service exchange is initiated by a client when it wants to get authentication credentials for an application server but it currently holds no credentials. Two messages are exchanged between the client and the Kerberos authentication server; then credentials for a TGS are given to the client, which is called the ticket-granting ticket (TGT) and will subsequently be used to obtain credentials for other services.

This exchange is also used for other services, such as the password-changing service. As seen in Figure B-5, the client's secret key is used exclusively in this phase.

Figure B-5 Simplified authentication service exchange

When a user logs into a client system and enters her password, a client sends the Kerberos AS a message that includes a user name in plain text (for example, Alice), the current time encrypted with her secret key, and the identity of the server for which the client is requesting credentials (TGS in Figure B-5).

Upon receiving the request from the client, the AS looks up the client name and the service name (the TGS in this case) in the Kerberos database, and then obtains an encryption key of each of them, K_{Alice} and K_{KDC}.

The AS then generates a response back to the client, which contains the TGT and a session key $K_{Alice,KDC}$, which is used in the subsequent secure communication between the client and KDC. The TGT includes the session key $K_{Alice,KDC}$, the identities of the server and the client, lifetime, and some other information. The AS then encrypts the ticket using its own key K_{KDC}. This produces a *sealed ticket*. The session key $K_{Alice,KDC}$ is also encrypted using the client's key K_{Alice} with some other information, such as nonce.

The encrypted current time is also known as the *authenticator*, because the receiver can assure that the sender knows the correct shared secret K_{Alice}, which is the client's encryption key derived from her password (this key is also referred as Alice's *long-term key*), by decrypting it and validating what is inside. Because the AS knows Alice's secret key, it can evaluate the time decrypted from the received authenticator. As you might have noticed, the clocks on the client system and the KDC must be reasonably synchronized with each other. A network time service may be used for this purpose.

An authenticator is also used to help the server detect the message replays.

A *nonce* is information to identify a pair of Kerberos requests and responses. A time stamp or a random number generated by a client may be used.

TGS is the server's identification, which is the Kerberos ticket-granting server in this case.

Since K_{Alice} is known exclusively by Alice and KDC, no one but Alice can extract the critical information from the response message, such as the session key $K_{Alice,KDC}$ to be used in the next phase.

When the client receives the AS's response, it decrypts it using its secret key K_{Alice} and checks to see if the nonce matches the specific request. If the nonce matches, the client caches the session key $K_{Alice,KDC}$ for future communications with the TGS.

Phase 2: Ticket-granting service exchange

The next phase is used for a client to obtain credentials for services that it wants to use. This exchange is also initiated by the client, and two messages are exchanged between the client and the TGS. The protocol and message format used in this exchange is almost identical to those for the AS exchange. The primary difference is that the client's key is never used in this exchange, but the session key obtained from the preceding AS exchange is used.

The request message that the client sends to the TGS contains several pieces of information including:

- Information to authenticate the client, which includes a new authenticator and the TGT obtained from the preceding AS exchange
- Identity of the service for which the client is requesting credentials
- Nonce to identify this request

Figure B-6 shows a simplified ticket-granting service exchange.

Figure B-6 Simplified ticket-granting service exchange

When the TGS receives the above message from the client, it first deciphers the sealed ticket using its encryption key K_{KDC}. From the deciphered ticket, the TGS obtains the session-key $K_{Alice,KDC}$. It uses this session key to decipher the authenticator. The validity checks performed by the TGS include:

- If the client name and its realm in the ticket match the same fields in the authenticator.
- If the address from which this message is originated is found in the address field in the ticket, which specifies addresses from which the ticket can be used.
- If the user-supplied checksum in the authenticator matches the contents of the request. This procedure guarantees the integrity of the message.

Finally, it checks the current time in the authenticator to make sure that the message is recent. Again, this requires that all the clients and servers maintain their clocks within some prescribed tolerance.

Note: By checking the time stamp in the nanoseconds scale, the replay attacks can be detected.

The TGS now looks up the server name from the message in the Kerberos database, and obtains the encryption key K_{Bob} for the specified service.

The TGS forms a new random session key $K_{Alice,Bob}$ for the benefit of the client (Alice) and the server (Bob), and then creates a new ticket tkt_to_Bob containing:

- The session key $K_{Alice,Bob}$
- Identities of the service and the client
- Lifetime

Note: The format of the ticket for a particular service is identical to one of the TGT.

It then assembles and sends a message to the client.

Phase 3: The client/server authentication exchange

The client/server authentication exchange is performed by the client and the server to authenticate each other. The client must have obtained credentials for the server using the AS or TGS exchange before the client/server authentication exchange is initiated.

After receiving the TGS exchange response from the TGS, the client deciphers it using the TGS session key $K_{Alice,KDC}$ that is exclusively known by the client and the TGS. From this message, it extracts a new session key $K_{Alice,Bob}$ that is shared with the server (Bob) and the client (Alice). The sealed ticket included in the response from the TGS cannot be deciphered by the client because it is enciphered using the server's secret key KBob.

Then the client builds an authenticator and seals it using the new session key $K_{Alice,Bob}$. At last, it sends a message containing the sealed ticket and the authenticator to the server (Bob) to request its service.

When the server (Bob) receives this message, it first deciphers the sealed ticket using its encryption key K_{Bob}, which is kept in secret between Bob and the KDC. It then uses the new session key $K_{Alice,Bob}$ contained in the ticket to validate the authenticator in the same way as the TGS does in the TGS exchange.

Figure B-7 shows a simplified client/server authentication exchange.

Figure B-7 Simplified client/server authentication exchange

When the server has authenticated a client, an option exists for the client to validate the server (this procedure is called *mutual authentication*). This prevents an intruder from impersonating the server.

If mutual authentication is required by the client, the server has to send a response message back to the client. The message has to contain the same time stamp value as one in the client's request message. This message is enciphered using the session key $K_{Alice,Bob}$ that was passed from the client to the server.

If the response is returned, the client decrypts it using the session key $K_{Alice,Bob}$ and verifies that the time stamp value matches one in the authenticator that was sent by the client in the preceding client/server authentication exchange. If it matches, then the client is assured that the server is genuine.

When the client/server authentication exchange has completed successfully, an encryption key is shared by the client and server and can be used for the ongoing application protocol to provide the data confidentiality.

Inter-realm operation

The Kerberos protocol is designed to operate across organizational boundaries. Each organization wanting to run a Kerberos server establishes its own realm. The name of the realm in which a client is registered is part of the client's name and can be used by the application server to decide whether to honor a request.

By establishing inter-realm keys, the administrators of two realms can allow a client authenticated in one realm to use its credentials in the other realm. The exchange of inter-realm keys registers the ticket-granting service of each realm as a principal in the other realm. A client is then able to obtain a ticket-granting ticket for the remote realm's ticket-granting service from its local ticket-granting service. Tickets issued to a service in the remote realm indicate that the client was authenticated from another realm.

This method can be repeated to authenticate throughout an organization across multiple realms. To build a valid authentication path to a distant realm, the local realm must share an inter-realm key with the target realm or with an intermediate realm that communicates with either the target realm or with another intermediate realm.

Realms are typically organized hierarchically. Each realm shares a key with its parent and a different key with each child. If an inter-realm key is not directly shared by two realms, the hierarchical organization allows an authentication path to be easily constructed. If a hierarchical organization is not used, it may be necessary to consult some database in order to construct an authentication path between realms.

Although realms are typically hierarchical, intermediate realms may be bypassed to achieve cross-realm authentication through alternate authentication paths. It is important for the end-service to know which realms were transited when deciding how much faith to place in the authentication process. To facilitate this decision, a field in each ticket contains the names of the realms that were involved in authenticating the client.

Some assumptions

The following limitations are applied to the Kerberized security environment:

- *Denial-of-service (DoS)* attacks are not addressed by Kerberos. There are places in these protocols where an intruder can prevent an application from participating in the proper authentication steps. Detection and solution of such attacks (some of which can appear to be “usual” failure modes for the system) is usually best left to human administrators and users.

  Note: Intrusion Detection Services (IDS) may be used to protect against such attacks as resource hogging.

- The secret key must be kept in secret by each principal (each client and server). If an attacker steals a principal's key, it can then masquerade as that principal or impersonate any server of the legitimate principal.
- Kerberos does not address *password-guessing* attacks. If a poor password is chosen, an attacker may be able to mount an offline dictionary attack by repeatedly attempting to decrypt messages that are encrypted with a key derived from the user's password.
- Kerberos assumes a loosely synchronized clock in the whole system. Workstations may be required to have a synchronization tool such as the time server provided.
- Principal identifiers should not be reused on a short-term basis. Access control lists (ACLs) may be used to grant permissions to particular principals.

Kerberos implementation in z/OS

The Kerberos version 5 server was introduced in IBM OS/390® V2R10 and implemented in SecureWay Security Server Network Authentication Service for z/OS. Kerberos provides strong authentication and encryption for the following applications:

- The UNIX Telnet server: Authentication support provided by the Kerberos 5 protocol
- The UNIX remote shell execution (rsh) server: Authentication support provided by the Kerberos 5 protocol and the GSSAPI protocol
- The FTP client and FTP server: Authentication support provided by the GSSAPI protocol

Restriction: The zSeries KDC is incompatible with Windows 2000 Kerberos applications. Windows 2000 applications must use the Windows KDC. To support Windows 2000 applications, a cross-realm connection between the zSeries KDC and the Windows KDC is required.

The following section provides a brief overview of how Kerberos is set up in z/OS.

RACF support for Kerberos

The Kerberos realm and its trust relationships with other realms is defined using the general resource class REALM. To define the local realm, set up a REALM class profile named KERBDFLT. Figure B-8 on page 365 shows a local realm ZOS18.RAL.IBM.COM being defined with a minimum ticket lifetime of 30 seconds, a default ticket lifetime of 10 hours, a maximum ticket lifetime of 24 hours, and a password of NEW1PW. All of the ticket lifetimes are specified in seconds. The administrator then lists the new REALM profile with the RACF RLIST command.

```
 RDEFINE REALM KERBDFLT KERB(KERBNAME(ZOS18.RAL.IBM.COM)   -
      PASSWORD(kerberos) MINTKTLFE(15) DEFTKTLFE(36000)    -
      MAXTKTLFE(86400))

     RLIST REALM KERBDFLT KERB NORACF
CLASS      NAME
-----      ----
REALM      KERBDFLT

KERB INFORMATION
----------------
KERBNAME= ZOS18.RAL.IBM.COM
MINTKTLFE= 0000000015
MAXTKTLFE= 0000086400
DEFTKTLFE= 0000036000
KEY VERSION= 001
KEY ENCRYPTION TYPE= DES DES3 DESD
***
```

Figure B-8 Setting up the local Kerberos realm using the RACF REALM class

A Kerberos principal is defined in the KERB segment of a user profile (in the same way that the UNIX information for a user is stored in the OMVS segment). You can use the RACF ADDUSER (for new users) or ALTUSER (for existing users) commands to add the KERB segment for a user ID. In the KERB segment, the KERBNAME parameter identifies the local principal name. Local principal names may contain embedded blanks and lowercase characters, and must be unique. For instance, the following associates the RACF user cs09 with Kerberos principal name CS09:

```
alu CS09 password(kerbpass) noexpired kerb(kerbname(CS09))
```

When you add a KERB segment to a user profile, RACF automatically sets up a profile in the KERBLINK class named with the KERBNAME parameter from the user's KERB profile. This enables RACF to have a mapping to a RACF user ID from a Kerberos principal name (which may or may not be the same). When you use ALTER NOKERB to remove a KERB segment from a user, or you use DELUSER to delete a user with a KERB segment, the KERBLINK profile is automatically deleted.

Note: Do not execute the DELUSER command, or an ALTUSER command with the NOKERB option, for a user profile that contains a KERB segment from RACF systems that do not support the KERBLINK class. These systems do not automatically manage KERBLINK profiles. You will inadvertently leave residual mapping profiles in the KERBLINK class. For information about recovery procedures, see z*/OS V1R8.0 Security Server RACF System Programmer's Guide*, SA22-7681.

Basic steps to follow to configure RACF to support Kerberos are:

- Customizing the local environment:
  - Defining your local RRSF (RACF remote sharing facility) node
  - Defining your local realm
  - Defining local principals
- Defining your foreign environment:
  - Defining foreign realms
  - Mapping RACF user IDs for foreign principals

The z/OS Kerberos KDC

The Kerberos KDC is implemented by started task SKRBKDC, as shown in Figure B-9. The RACF user ID associated with the started task must have a UNIX UID of UID(0) (a superuser).

```
//*********************************************************************
//*                                                                   *
//* Procedure for starting the Kerberos Security Server               *
//*                                                                   *
//*********************************************************************
//SKRBKDC  PROC  REGSIZE=256M,OUTCLASS='S'
//*-------------------------------------------------------------------
//GO       EXEC  PGM=EUVFSKDC,REGION=&REGSIZE,TIME=1440,
//  PARM=('ENVAR("LANG=En_US.IBM-1047"),TERM(DUMP)                    X
//             / 1>DD:STDOUT 2>DD:STDERR')
//STDOUT   DD  SYSOUT=&OUTCLASS,DCB=LRECL=250,
//  FREE=END,SPIN=UNALLOC
//STDERR   DD  SYSOUT=&OUTCLASS,DCB=LRECL=250,
//  FREE=END,SPIN=UNALLOC
//SYSOUT   DD  SYSOUT=&OUTCLASS,
//  FREE=END,SPIN=UNALLOC
//CEEDUMP  DD  SYSOUT=&OUTCLASS,
//  FREE=END,SPIN=UNALLOC
```

Figure B-9 Started task for Kerberos server

The SKRBKDC started task reads the Kerberos server configuration file from the SKRBKDC RACF user's OMVS home directory. The Kerberos configuration file specifies which IP host and port the KDC server should be started on for the local realm as well as the IP host and port numbers for KDCs in other realms.

```
;-----------------------------------------------------
;  Sample Kerberos configuration file
;-----------------------------------------------------

[libdefaults]

default_realm = ZOS18.RAL.IBM.COM 1
kdc_default_options = 0x00000010
use_dns_lookup = 0

; Default encryption types if DES3 is not supported
default_tkt_enctypes = des-cbc-crc
default_tgs_enctypes = des-cbc-crc
; Default encryption types if DES3 is supported
;default_tkt_enctypes = des3-cbc-sha1,des-cbc-crc
;default_tgs_enctypes = des3-cbc-sha1,des-cbc-crc

[realms]

ZOS18.RAL.IBM.COM = {
  2 KDC = WTSC63C.ZOS18.RAL.IBM.COM:88
  3 KPASSWD_SERVER = WTSC63C.ZOS18.RAL.IBM.COM:464
}

[domain_realm]

.ZOS18.RAL.IBM.COM = ZOS18.RAL.IBM.COM
```

Figure B-10 Sample Kerberos configuration file

Figure B-10 shows an example of a Kerberos server configuration file.

1. The default_realm statement specifies the realm name that is used when a principal wants to start communicating with another principal, and does not specifically state the realm. This should be the DNS root of your system that the KDC will run on.
2. The KDC statement for a realm specifies the host name and port number of the KDC server for that realm.
3. The KPASSWD_SERVER statement for a realm specifies the host name and port number of the Password Change server for that realm.

It should be noted that in the above example, the local realm is ZOS18.RAL.IBM.COM (as set up in Figure B-8 on page 365), and that the host name on the local realm points to where the KDC will be opening a socket. These sockets must be reserved for job name OMVS in the TCP/IP stack that the server will be running on.

Verifying correct KDC startup

After the SKRBKDC started task has successfully started, check to ensure that the KDC server is listening on the correct sockets in the TCP/IP stack that you have targeted using the **onetstat -s -p *stackname*** OMVS command.

In your TSO logon proc, ensure that the Kerberos REXX data set EUVF.SEUVFEXC is in the SYSEXEC concatenation. In OMVS, ensure that the PATH variable has subdirectory /usr/lpp/skrb/bin before any other bin library for the user.

```
CS09 @ SC63:/cs09>kinit CS09
EUVF06017R Enter password:

CS09 @ SC63:/cs09>klist
 Ticket cache: FILE:/var/skrb/creds/krbcred_cf635eb0
 Default principal: CS09@ZOS18.RAL.IBM.COM

Server: krbtgt/ZOS18.RAL.IBM.COM@ZOS18.RAL.IBM.COM
  Valid 2002/05/30-10:23:39 to 2002/05/30-20:23:39
CS09 @ SC63:/cs09>
```

Figure B-11 Kerberos installation verification procedure

To get an initial ticket from Kerberos, enter the **kinit** command. The first parameter is the principal name. In Figure B-11, the principal name that we are getting a ticket for is CS09 (note that the case is exactly the same as that entered on the RACF command used to add the KERB segment for the user).

Note: When using the **kinit** command, the password that you enter must be uppercase. This is because when you add it with the RACF ALTUSER command, RACF translates the password to uppercase.

The password was then entered (it must be in uppercase) and the ticket was received from the KDC. The **klist** command shows a list of *credentials* that the current user has. In this case, there is only one, for the default realm ZOS18.RAL.IBM.COM.

The user is now ready to log on to a Kerberized server, such as otelnetd or FTP.

For information about how to Kerberize a server application, see that server's documentation.

For further information about the Kerberos server in z/OS, refer to *z/OS V1R7.0 Integrated Security Services Network Authentication Service Administration*, SC24-5926.

C

Certificate management in z/OS

Digital certificates have to be created and maintained within a central repository. In this chapter, we discuss the use of RACF and the **gskkyman** utility to provide this function. Using these utilities, we discuss how digital certificates and key rings are created and maintained.

In this chapter, we cover the topics listed in Table C-1.

Table C-1 Topics covered in this appendix

Section	Topic
“Digital certificates” on page 370	This section discusses the basic concepts surrounding digital certificates.
“How to generate digital certificates in z/OS” on page 371	We tell you how to generate digital certificates in z/OS.
“Digital certificate field formats” on page 372	This section looks at how digital certificates are structured.
“RACF RACDCERT command use” on page 374	This section discusses using RACF to manage digital certificates.
“RACF key rings” on page 375	This section discusses using RACF to define key rings.
“gskkyman command use” on page 377	This section discusses using **gskkyman** to manage digital certificates.
“Client certificates” on page 379	We discuss the use of client certificates.
“Server certificates” on page 379	We discuss the use of server certificates.
“Self-signed certificates” on page 380	We discuss the use of self-signed certificates.
“Obtaining certificates” on page 380	In this section, we cover how to obtain digital certificates in a z/OS environment.
“Certificate locations example” on page 413	This topic covers where certificates are stored in RACF and gsyyyman z/OS UNIX files.

© Copyright IBM Corp. 2007. All rights reserved.

Digital certificates

In the z/OS environment, digital certificates are used by SSL/TLS to authenticate and encrypt the protocol handshaking messages. An SSL/TLS server must send its certificate to the client, and a server can optionally request a certificate from the client. For the purposes of this appendix, SSL and TLS are equivalent unless stated otherwise.

There are two ways for you to obtain a certificate. One is to request a Certificate Authority (CA) to create your certificate. If you are requesting a certificate for a server, and you plan to make your server available to the public or your business partners, you should get your certificate from a trusted CA such as VeriSign, Inc., or any other CA whose root certificate is contained in the key database of the clients who use your server.

The second way for you to obtain a certificate is to generate one yourself. This type of certificate is called a *self-signed* certificate, because the issuer of the certificate is the same as the subject of the certificate. This type of certificate might be useful for testing purposes or for securing TLS connections within your intranet.

To validate a certificate, the receiver checks its key database for a trusted CA certificate that has the same distinguished name as that of the received certificate's certifier. Thus the CA certificates must be located in the client's and server's local database (or *key ring*) and marked as trusted. See Figure C-1.

Figure C-1 SSL certificate management: CA-signed certificates

If you choose to use a self-signed certificate instead of a CA-signed certificate (shown in Figure C-2 on page 371), the CA certificate that should be trusted is identical to the server/client certificate itself. If the server itself has signed its certificate and client authentication is not required, the server certificate must be exported and stored in the client's local database as a trusted CA certificate, because the server certificate *is* the issuer's certificate for itself.

Figure C-2 SSL certificate management: Self-signed certificate without client authentication

How to generate digital certificates in z/OS

Most SSL-enabled applications in z/OS make use of the System SSL toolkit as described in “System SSL” on page 356. For certificate storage and management, two command utilities exist:

- The RACF command RACDCERT, which creates and maintains certificates and key rings that are stored in the RACF database. This command can also be used to create self-signed certificates and certificate requests for other CAs.
- The **`gskkyman`** utility, which creates and maintains a key database as a file in the z/OS UNIX file system. It can also create self-signed certificates and certificate requests for other CAs.

Using RACF key rings is the preferred method because it provides better security for the certificates and their private keys. With RACF key rings, stash files containing key database passwords are not used and access to key rings and certificates is controlled by RACF. In this appendix, we show both methods of creating and managing certificates.

For detailed information about the creation and maintenance of digital certificates in z/OS, see Chapter 9, “Certificate/Key Management,” in *z/OS V1R8.0 Cryptographic Services System SSL Programming*, SC24-5901. For a reference on the RACDCERT command, see *z/OS V1R8.0 Security Server RACF Command Language Reference*, SA22-7687.

Table C-2 summarizes all applications that make use of the certificate management tools in z/OS V1R8.

Table C-2 Applications that use digital certificates in z/OS V1R8

		RACDCERT	gskkyman
SSL	TN3270 server	X	X
	HTTP Server	X	X
	PAGENT Client		X
	Lightweight Directory Access Protocol (LDAP) server	X	X
	Policy Agent		X
	DCAS server	X	X
	Firewall configuration client	X	X
TLS	FTP Server	X	X
IKE	IKE server	X	
	AT-TLS applications	X	X
	IPSec/IKED	X	X

Digital certificate field formats

When you create a digital certificate, whether using **gskkyman** or RACDCERT, certain fields are required and others are optional. We cover the most important fields here.

- Certificate Version Number: This is always 3. **gskkyman** will ask for the number, while RACDCERT sets it automatically.
- Distinguished Name: The issuer of a certificate and the subject of a certificate are both represented by a *distinguished name*. This name takes the form of a hierarchy, although different certificate issuers treat the format differently. For a self-signed certificate, the issuer's distinguished name will be copied from the subject's distinguished name. A distinguished name contains the following subfields (with RACDCERT parameter names in parentheses):
  - Common Name: (CN). For a server certificate, this field normally contains the server's DNS name. For a client certificate, this will identify the individual or computer.
  - Organization-name: (O). Company name or similar.
  - Title: (T). Salutation for an individual.
  - Organizational-unit: (OU). Used for classification within the *Organization-name*, listed above.
  - Locality: (L). City or town.
  - State-or-province: (SP).
  - Country: (C). Two-character ISO code for country.

    The only compulsory subfields of the distinguished name are the common name, the organization name, and the country.
- Period of validity: The **gskkyman** utility asks for the number of days from today that the certificate is valid for, while RACDCERT sets the lower and upper dates with the NOTBEFORE(DATE(yyyy-mm-dd) and the NOTAFTER(DATE(yyyy-mm-dd) parameters.

The Label field is also needed. This field is not part of the X509 specification, but it is used to organize certificates in the key database. You can use the label to list, alter, and delete individual certificates. A label must be unique except for storage within RACF, where labels can be duplicated as long as they are associated with different RACF user IDs (with the ID(user..) parameter).

Figure C-3 shows a batch job used to create a self-signed digital certificate using RACDCERT.

```
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add the top-level self-signed certificate for the certificate
//*   authority (ourselves)
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT CERTAUTH GENCERT                        -
1 SUBJECTSDN( O('I.B.M Corporation')               -
                CN('server.raleigh.ibm.com')       -
                 C('US'))                          -
      NOTAFTER(DATE(2002-08-22))                   -
      SIZE(512)                                    -
      WITHLABEL('Label for RACDCERT cert')
/*
```

Figure C-3 Setting up a test self-signed certificate, for a gskkyman comparison

1 The SUBJECTSDN parameter encloses all the Distinguished Name subfields for the subject.

Figure C-4 on page 374 shows the same certificate being created (with a different label) using **gskkyman** in order to show how the certificate fields are specified in each utility. The required fields of Common Name, Organization, and Country were specified; the key size was set to 512 bits; and a 100-day period of validity was set.

```
Current key database is /example.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 5
Enter version number of the certificate to be created (1, 2, or 3) [3]:  3
Enter a label for this key...............> Label for gskkyman cert
Select desired key size from the following options (512):
    1:    512
    2:    1024
Enter the number corresponding to the key size you want: 1
Enter certificate subject name fields in the following.
    Common Name (required)................> server.raleigh.ibm.com
    Organization (required)...............> I.B.M Corporation
    Organization Unit (optional)..........>
    City/Locality (optional)..............>
    State/Province (optional).............>
    Country Name (required 2 characters)..> US
Enter number of valid days for the certificate [365]:  100
Do you want to set the key as the default in your key database? (1 = yes, 0 = no
) [1]: 0
Do you want to save the certificate to a file? (1 = yes, 0 = no) [1]: 0

Please wait while self-signed certificate is created...
```

Figure C-4 Example of setting up a certificate in gskkyman

RACF RACDCERT command use

RACF can be used to create, register, store, and administer digital certificates and the private keys associated with the certificates. RACF can also be used to create and manage key rings of stored digital certificates. Certificates are stored in the RACF database, while private keys may be stored in the ICSF Public Key Data Set (PKDS), encrypted under a 168-bit Triple-DES key.

RACF distinguishes three types of digital certificates:

- Certificate Authority certificates: These certificates are associated with CAs and are used to verify signatures in other certificates.
- Site certificates: These certificates are associated with servers or network entities in locations other than the local system.
- User certificates: These certificates are associated with a RACF user ID and are used to authenticate a user's identity.

A user certificate or a certificate that has been connected to a key ring with USAGE(PERSONAL) is the only type of certificate whose private key can be used to create signatures. Therefore, all server certificates for local servers need to be user certificates or they need to be connected to an appropriate key ring with USAGE(PERSONAL).

The RACF ISPF panels can be used to maintain the digital certificates if you do not choose to use the TSO RACDCERT command. Our examples show the TSO commands as they can be submitted in a batch job.

RACF key rings

A RACF key ring is a way to logically group together a number of certificates. Certificates can be “connected” to one or more key rings.

Each key ring is associated with only one user, but the certificates that are connected to that key ring may or may not be the key ring owner’s certificate.

Figure C-5 Example showing how key rings contain pointers to certificates

Figure C-5 shows the logical relationship between RACF key rings and digital certificates stored in the RACF database. There can be more than one key ring in the database with the same name, but each must be assigned to a different user ID.

Typically, a z/OS server that uses digital certificates will have a configuration parameter where the RACF key ring name is specified. TN3270 and FTP are examples of servers that use key rings. During SSL/TLS handshaking, the server sends its certificate to the client. The server will get its certificate from the RACF key ring specified in the server’s configuration file and that is associated with the server’s RACF user ID. A server also looks at the certificates in its key ring for a CA certificate with which to validate the client certificate, if client authentication is configured.

A z/OS client that uses digital certificates (such as FTP) will have a configuration parameter where the RACF key ring name is specified. During SSL/TLS handshaking, the server sends its certificate to the client, and the client looks at the certificates in its key ring for a CA certificate with which to validate the server certificate. If the server requests the client certificate, the client will get its certificate from the RACF key ring specified in the client's configuration file and that is associated with the client's RACF user ID. Note that a client certificate must be in TRUSTED status in the RACF database.

RACDCERT command security

Authority to the IRR.DIGTCERT.function resource in the FACILITY class allows a user to issue the RACDCERT command. To issue the RACDCERT command, users must have one of the following RACF authorities:

- The SPECIAL attribute
- Sufficient authority to resource IRR.DIGTCERT.function in the FACILITY class
- READ access to IRR.DIGTCERT.function to issue the RACDCERT command for themselves
- UPDATE access to IRR.DIGTCERT.function to issue the RACDCERT command for others
- CONTROL access to IRR.DIGTCERT.function to issue the RACDCERT command for SITE and CERTAUTH certificates (This authority also has other uses.)

Important: Any z/OS-based client or server that uses a RACF key ring issues an internal RACDCERT LIST and RACDCERT LISTRING command. The RACF user ID associated with the server must therefore be granted READ access to the RACF profiles controlling these commands, which are IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING. For a list of servers that use RACF key rings, see Table C-2 on page 372.

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(TCPIPA) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(TCPIPA) ACCESS(READ)
SETR RACLIST(FACILITY) REFRESH
```

Figure C-6 RACF commands to the TCP/IP user ID

Figure C-6 shows the RACF commands needed to permit a user (TCPIPA in this case) to issue the RACDCERT LIST and RACDCERT LISTRING commands.

For more information, see *z/OS V1R8.0 Security Server RACF Security Administrator's Guide*, SA22-7683.

RACDCERT command format

```
RACDCERT [ID(user) | SITE | CERTAUTH] command-options
```

The RACDCERT command can be directed to a RACF user ID's digital certificates or key rings by the ID(user) parameter, to a CA's resources by the CERTAUTH parameter, and to a site's resources by the SITE parameter. If no ID, SITE, or CERTAUTH parameter is included, the command issuer's ID is used.

For instance, the command **`racdcert certauth list`** will list all CA certificates in the RACF database, while **`racdcert list`** shows all of your (the command issuer's) certificates.

There is also a *multiid* parameter for mapping functions. This and other parameters are explained fully in *z/OS V1R8.0 Security Server RACF Security Administrator's Guide*, SA22-7683.

gskkyman command use

The **gskkyman** UNIX command is used to create and maintain digital certificate key databases in a z/OS UNIX file system. This is an alternative to storing digital certificates in the RACF database. Note that if you are using SSL/TLS client authentication to map a digital certificate to a RACF user ID, then you must use the RACF RACDCERT command to store the client certificate, not **gskkyman**.

In the examples shown later in this appendix, we assume that a key database has been set up. The procedure to set up a new key database (and stash file) is as follows:

1. Set up access to the **gskkyman** command from your UNIX shell. This is covered in *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775.
2. From the UNIX shell, enter the **gskkyman** command. Figure C-7 shows the initial panel. This example shows how to create a new key database in the z/OS UNIX file system. The database will be created in the subdirectory from which you entered the **gskkyman** command. The password you enter here will be used to open the database in the future.

```
FOCAS @ SC63:/>gskkyman


            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 1
Enter key database name or press ENTER for "key.kdb": example.kdb
Enter password for the key database.......>
Enter password again for verification.....>
Should the password expire? (1 = yes, 0 = no) [1]: 0

The database has been successfully created, do you want to continue to work with
 the database now? (1 = yes, 0 = no) [1]: 0
```

Figure C-7 Setting up a new key database in a z/OS UNIX using gskkyman

Because the key database has a password, there must be a mechanism for a server to supply it to read the contents. This mechanism is implemented by using a stash file, which is a file using the same name as the key database, but with a suffix of .sth rather than .kdb. This file contains the key database password in encrypted form, and is created from the **gskkyman** panel.

3. Create the password stash file. This is shown in Figure C-8.

```
FOCAS @ SC63:/>gskkyman


            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 2
Enter key database name or press ENTER for "key.kdb": example.kdb
Enter password for the key database.......>


            Key database menu

Current key database is /example.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 11

The encrypted password has been stored in file /example.sth

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 1
FOCAS @ SC63:/>ls -la example.*
-rw-------   1 HAIMO    SYS1       65080 May 15 17:57 example.kdb
-rw-------   1 HAIMO    SYS1          80 May 15 17:57 example.rdb
-rw-------   1 HAIMO    SYS1         129 May 15 18:06 example.sth
FOCAS @ SC63:/>
```

Figure C-8 Creation of the stash file using gskkyman

Note that after the stash file was created, the UNIX file attributes were displayed with the UNIX **ls** command. As you can see in Figure C-8, the file attributes of the key database and the stash file are both “-rw-------", which means only the creator of the database (the user of the **gskkyman** command) can read and write to this file. You should use the UNIX **chmod** command to set the permission bits so that the server’s UNIX UID is able to read both the key database and the stash file. An example command to allow the owner read/write access and the owners group to have read access is **chmod 640 example.***.

Client certificates

An SSL/TLS-enabled server *may* request that the client produce a digital certificate to verify the client's identity. The server must then validate the client certificate by checking the trusted CA hierarchy in its own key database, to ensure the digital signature on the certificate is from a trusted CA. The server does not make use of the client's public key contained in the certificate for communications; the request is for identification purposes only.

If the client passes a self-signed certificate (one that the client has generated and signed itself), then the server must check to ensure that it has a copy of the same certificate in its key database and that the certificate is marked as trusted.

When client authentication is requested by the server, the server will be configured to authenticate to a particular level. These levels are:

Level 1 The server ensures that the signer of the client's certificate is trusted by checking the trusted CA certificates that are in the server's key ring.

Level 2 The authentication requires that the client certificate be registered with RACF (or another SAF-compliant security product) and that it be in "TRUSTED" status. The RACF user ID that the certificate is associated with is that given in the ID() parameter of the RACFDCERT ID() ADD command when the client certificate was added to RACF. The CA that issued the client certificate must have a CA certificate connected to the server's key ring. Note that this level cannot be used if the z/OS server is using a key database created by using **gskkyman**.

Level 3 The authentication provides, in addition to level 1 and level 2 support, the capability to restrict access to the server based on the user ID returned from RACF. This level is implemented entirely in RACF, that is, a server only selects level 2 authentication, and if the appropriate profiles for the server are defined in RACF, the authentication level is upgraded to level 3. Note that this level cannot be used if the z/OS server is using a key database that is created by using **gskkyman**.

Server certificates

As discussed in "SSL protocol description" on page 352, the SSL/TLS protocol requires a server to supply a digital certificate to a client. The client must then validate the server certificate by checking the trusted CA hierarchy in its own key database, to ensure the digital signature on the certificate is from a trusted CA. Then the client can use the server's public key from the certificate to communicate the rest of the SSL handshake.

> **Important:** To implement SSL in any form, you must have a server certificate available to the server and client. This is a prerequisite for implementing any client authentication that is discussed in this appendix.

If the server passes a self-signed certificate (one that the server has digitally signed itself), then the client must check to ensure that it has a copy of the same certificate in its key database, and that the certificate is marked as "TRUSTED".

Self-signed certificates

The server or client certificate may be self-signed. This means that the digital signature on the certificate can only be verified by the public key given on the same certificate. The certificate is not authenticated by any CA and must be taken at face value by the client or server receiving it.

The normal validation procedure for a certificate is still performed for a self-signed certificate. This means that the receivers check their key database for the CA that signed the certificate, but, as already mentioned, the CA is represented by the certificate received. Therefore, the certificate must have been previously received by some other means, and placed in the receiver's key database as a trusted certificate.

Obtaining certificates

This section shows the practical steps necessary to obtain digital certificates in a z/OS environment. If you choose not to use self-signed certificates, you will need to request your client/server certificates from a CA. That CA can be either an external organization such as VeriSign, or you can create a CA internally by generating a CA certificate yourself, and using that to sign other certificates. You can also generate self-signed certificates where the CA is the certificate itself. This is the simplest form of certificate usage.

In all the examples that follow, the server runs on z/OS under the RACF user ID "STC", and the end-user's RACF user ID is "FOCAS". The end user's user ID is only needed when you are storing client certificates in RACF using RACDCERT.

Procedures for obtaining and storing self-signed certificates can be found in "Self-signed certificates". Procedures for obtaining and storing internal CA signed certificates can be found in "Internal Certificate Authority" on page 396. Procedures for obtaining and storing external CA signed certificates can be found in "External Certificate Authority" on page 401.

Self-signed certificates

The aim of this section is to show how to use the TSO RACDCERT command and the UNIX **gskkyman** command to store and use self-signed certificates. For the purposes of the examples, it is assumed the server is on z/OS and the client is not.

The procedure to use RACDCERT to generate and manage a self-signed server certificate is shown in "Self-signed server certificate RACDCERT procedure". The procedure to use RACDCERT to import and manage a self-signed client certificate is shown in "Self-signed client certificate RACDCERT procedure" on page 385. The procedure to use **gskkyman** to generate and manage a self-signed server certificate is shown in "Self-signed server certificate gskkyman procedure" on page 388. The procedure to use **gskkyman** to import and manage a self-signed client certificate is shown in "Self-signed client certificate gskkyman procedure" on page 392.

Self-signed server certificate RACDCERT procedure

This procedure is basically the same for any z/OS server. In this example, we are generating and storing a certificate for use by a TN3270 server.

Once generated, the server certificate is placed in the server's RACF key ring and also exported to the client to be placed in the client's key database as a trusted CA certificate.

Here are the steps required:

1. Generate a self-signed certificate for the server, as shown in Figure C-9.

```
//FOCAS1   JOB 'SET UP TN3270 CERT','PETER FOCAS',CLASS=A,MSGCLASS=X
//SERVCRT  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* set up the TN3270 server certificate, and self-sign it.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
1 RACDCERT ID(STC) GENCERT SUBJECTSDN(CN('ITSO.RALEIGH.IBM.COM') -
         O('IBM Corporation')                                    -
        OU('ITSO Raleigh TN3270 Server')                         -
         C('US'))                                                -
 WITHLABEL('TN3270 Server')
/*
```

Figure C-9 Batch job to create self-signed server certificate

The ID(STC) parameter 1 associates the certificate being generated with the RACF user “STC”. This is the user ID that the server in our example is running under. Yours will probably be different. For an explanation of the rest of the RACDCERT parameters, see “Digital certificate field formats” on page 372.

Note that because there is no RACDCERT SIGNWITH parameter specified on the GENCERT command, the certificate will be digitally signed by the private key owned by the subject of the certificate. This is the definition of a self-signed certificate. Make sure that the common name (CN) is the same as the host or domain name of the server.

2. Create a RACF key ring for the server.

```
//FOCAS1   JOB 'SET UP TN3270 CERT','PETER FOCAS',CLASS=A,MSGCLASS=X
//KEYRING  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add a new Keyring to the TN3270 servers RACF ID (STC), then....
//* Add TN3270 server certificate to the user 'STC's keyring. the
//*   Keyring name is from the TN3270 configuration statement as below
//*      'KEYRING SAF TN3270Ring'
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(STC) ADDRING(TN3270Ring)
  RACDCERT ID(STC) CONNECT(ID(STC)                -
                           LABEL('TN3270 Server') -
                           RING(TN3270Ring)       -
                           DEFAULT                -
                           USAGE(PERSONAL))
/*
```

Figure C-10 Batch job to add a key ring for the self-signed certificate

Figure C-10 shows the two steps necessary to create the key ring for the server:

a. Create a new RACF key ring using the RACDCERT ADDRING command.

b. Connect the self-signed servers certificate to the new key ring using the RACDCERT CONNECT command.

Note that the RING parameter specifies the same ring name as what you have configured into the server. In the TN3270 server, this is specified on the KEYRING SAF *ringname* statement, and on the FTP server, it is on the KEYRING statement in FTP.DATA. The DEFAULT statement is needed, because there may be more than one certificate in the key ring, and System SSL needs to know which certificate to pass to the client.

3. Export the self-signed server certificate to an MVS database.

```
//FOCAS1   JOB 'EXPORT SERVER CERT','PETER FOCAS',CLASS=A,MSGCLASS=X
//EXPORT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Export the Self-signed Server certificate from the RACF database
//*   in base-64 encoded format. This is then FTP'd to the TN3270
//*      client so that it can verify the same certificate
//*      when passed in the SSL exchange.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(STC) EXPORT(label('TN3270 Server'))  -
              FORMAT(CERTB64) DSN('FOCAS.RACDCERT.TN32CERT')
/*
```

Figure C-11 Batch job to write the internal CA certificate to an MVS data set

Figure C-11 shows the RACDCERT EXPORT command being used to export the self-signed server certificate to an MVS data set.

4. FTP the certificate exported to the MVS data set in step 3 to the client that will use it.

   This step is not shown, because any FTP client will be able to perform this step. Note that in the example, the exported certificate in the MVS data set is in EBCDIC format. Therefore, the FTP must perform EBCDIC-to-ASCII translation if the client is on an ASCII host. The MVS data set will be sent via FTP to any client that needs to use that certificate to validate the same certificate when presented by a server in an SSL exchange. Depending on the number of clients in an enterprise, this may result in a large number of transfers. One way to reduce the number of file transfers to clients is for all clients to pick up their key database from a LAN drive.

Figure C-12 Personal Communications client certificate management window

5. At the client, the certificate received from step 4 must be imported into the key database as a trusted certificate.

   Depending on the type of client, there are a number of different ways to do this. In the case of a Windows Personal Communications client (a TN3270 client), select the Certificate Management or Certificate Wizard icon from the Utilities folder. This displays the window (after the key database file is opened) shown in Figure C-12. You now import the certificate by clicking the **Add** button. When the certificate is added, the window seen in Figure C-13 on page 384 shows the detail display from the certificate, showing the key size of 1024 (set by default with the RACDCERT GENCERT command), the certificate version, and the *Issued To* name the same as the *Issued B*y (this is a self-signed certificate).

Figure C-13 Personal Communications client: Display of imported self-signed server certificate

6. Test the client-to-server connection.

   The Personal Communications client was instructed to connect to the TN3270 server using SSL. Figure C-14 on page 385 shows Personal Communications displaying the server certificate (by clicking **Communication** → **Security** → **Server**), which shows that the certificate subject is the TN3270 server, and the certificate issuer is the same. This is the self-signed server certificate set up in step 1.

Figure C-14 Personal Communications client: server certificate being used and signer's details

While this discussion showed the certificate being generated for a TN3270 server, followed by an export to the client and placement in the client's key database as a trusted certificate, the procedure for any client/server is basically the same.

Self-signed client certificate RACDCERT procedure

A client certificate must be added to RACF and associated with the appropriate RACF user ID using the RACFDCERT ID(clients-user ID) ADD.... statement. The client certificate's CA must be connected to the server's RACF key ring using the RACFDCERT ID(servers-user ID) CONNECT. The basic procedure to follow is:

1. Get the client certificate. For a self-signed certificate, this is normally generated at the client end.

   Because different client programs have different ways to generate a client certificate, this should be thought of as a generic example. Most clients will have some way to generate a certificate. In the case of Personal Communications, which provides a TN3270 client, use the Windows menu (click **Start** → **Programs** → **IBM Personal Communications** → **Utilities** → **Certificate Management**) to open the client's key database. Then click **Create** → **New Self-signed Certificate** to generate the certificate. See Figure C-15 on page 386 for an example of a self-signed client certificate that has just been generated by Personal Communications into the client key database.

Figure C-15 Personal Communications client: Newly added self-signed client certificate

2. Export the certificate from the client's key database to a certificate file.

   Export the client certificate to a file. Most certificate management utilities allow the export of a certificate in at least two formats. The most common is Base64-encoded ASCII, but there is also the binary DER format. The choice depends on what your certificate management utility at the server end can use as an import format. We will use Base64 ASCII format. At the lower right-hand side of the window shown in Figure C-15 is the Extract Certificate... button. This is used to write a certificate to a data set. The window that is presented is shown in Figure C-16 on page 387. Note that the Data Type field specifies Base64-encoded ASCII and that the certificate will be written to the cert.arm file.

Figure C-16 Exporting a client certificate to a file

3. FTP the certificate file from the client to the z/OS server side.

   This does not need to be shown here. You just need to FTP the certificate file created in step 2 (cert.arm in this example) to an MVS data set at the server side as a text file.

4. Add the client certificate into the RACF database and associate with an end user.

   Figure C-17 shows the batch job used to add the client certificate to the RACF database and associate it with the RACF user ID “FOCAS” (the owner of the client certificate) with the ID() parameter. The RACDCERT ADD statement specifies the MVS data set name that contains the client certificate; this was the data set name that was created by the FTP in step 3. The certificate is set to TRUSTED status, which means that any certificate that is signed with this certificate will pass authentication.

```
//FOCAS1   JOB 'EXPORT SERVER CERT','PETER FOCAS',CLASS=A,MSGCLASS=X
//CLIENT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Import the Self-signed Client certificate into the RACF database
//*   This was FTP's from the workstation TN3270 client, that
//*      generated it with the local PCOMM utility. It must be
//*      imported as a trusted CA certificate
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(FOCAS) ADD('FOCAS.RACDCERT.CLIENT.CERT') -
               TRUST WITHLABEL('TN3270 client certificate PF')
/*
```

Figure C-17 Batch job to add a client certificate into the RACF database as TRUSTED

5. Connect the client certificate to the server's RACF key ring.

   Figure C-18 shows the batch job used to connect the client certificate, added in step 4, to the server's key ring. In the RACDCERT CONNECT statement, the key ring name is whatever is coded in the server's configuration file (in TN3270's case, it is specified on the KEYRING SAF statement). This assumes that the key ring is already set up (probably because you have the server's certificate there). If the ring is not set up, you must create it before this job is run with the RACDCERT ID(STC) ADDRING(TN3270Ring) command (assuming the server's RACF ID is *STC* and key ring name is *TN3270Ring*).

```
//FOCAS1   JOB 'EXPORT SERVER CERT','PETER FOCAS',CLASS=A,MSGCLASS=X
//CLIENT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Import the Self-signed Client certificate into the RACF database
//*   This was FTP's from the workstation TN3270 client, that
//*      generated it with the local PCOMM utility. It must be
//*      imported as a trusted CA certificate
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
RACDCERT ID(STC) CONNECT(ID(FOCAS)              -
                         LABEL('TN3270 client certificate PF') -
                         RING(TN3270Ring)       -
                         USAGE(PERSONAL))
/*
```

Figure C-18 Batch job to import client certificate into RACF

After you complete these steps (assuming that you have followed the steps to obtain and store the server's certificate) and both the client and server are configured with the appropriate parameters for client authentication, the connection can be made.

Self-signed server certificate gskkyman procedure

It is assumed that you have set up a key database and produced a stash file for the server as discussed in "gskkyman command use" on page 377, that you have set the correct UNIX permission bits so that the server can read the files, and that the database is named tn32v1r8.kdb. After the database is created, generate a self-signed certificate into it, and set it as the default certificate. The self-signed certificate must then be exported to the client workstation, where it is imported into the client's key database as a trusted CA certificate.

Figure C-19 shows **gskkyman** being used to open the existing database in preparation for generating the certificate:

1. Open the key database using **gskkyman**.

```
FOCAS @ SC63:/>gskkyman


            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 2
Enter key database name or press ENTER for "key.kdb": tn32v1r8.kdb
Enter password for the key database.......>


            Key database menu

Current key database is /tn32v1r8.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu):
```

Figure C-19 Opening the key database

Now that the key database is opened, you are presented with a list of options. Option 5 creates a self-signed certificate by prompting for details. The generation of the self-signed certificate is shown in Figure C-20 on page 390, which carries on from Figure C-19 by selecting option 5.

2. Create the self-signed certificate and export it to a z/OS UNIX file.

```
Enter option number (or press ENTER to return to the parent menu): 5
Enter version number of the certificate to be created (1, 2, or 3) [3]:
Enter a label for this key................> TN3270 gskkyman certificate
Select desired key size from the following options (512):
    1:    512
    2:    1024
Enter the number corresponding to the key size you want: 2
Enter certificate subject name fields in the following.
    Common Name (required)................> ITSO.RALEIGH.IBM.COM
    Organization (required)...............> IBM
    Organization Unit (optional)..........> ITSO Raleigh TN3270 Server
    City/Locality (optional)..............> Raleigh
    State/Province (optional).............> NC
    Country Name (required 2 characters)..> US
Enter number of valid days for the certificate [365]:
Do you want to set the key as the default in your key database? (1 = yes, 0 = no
) [1]: 1
Do you want to save the certificate to a file? (1 = yes, 0 = no) [1]: 1
Should the certificate binary data or Base64 encoded ASCII data be saved? (1 = A
SCII, 2 = binary) [1]:
Enter certificate file name or press ENTER for "cert.arm": tn32v1r8.arm

Please wait while self-signed certificate is created...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 1
FOCAS @ SC63:/>
```

Figure C-20 Generating a self-signed certificate using gskkyman

Figure C-20 shows the dialog between the user and **gskkyman** to set up a self-signed certificate. The key size is requested; then details about the certificate subject are entered, whether you want to set the certificate as the default certificate, and whether you want to produce an exported ASCII file of the certificate. This exported certificate is what is sent via FTP to the client for importation into the client's key database. In this example, the exported file is called tn32v1r8.arm. The contents of the exported file are shown in Figure C-21.

```
FOCAS @ SC63:/>cat tn32v1r8.arm
-----BEGIN CERTIFICATE-----
MIICczCCAdygAwIBAgIEPOOwjjANBgkqhkiG9w0BAQQFADB+MQswCQYDVQQGEwJV
UzELMAkGA1UECBMCTkMxEDAOBgNVBAcTB1JhbGVpZ2gxDDAKBgNVBAoTAOlCTTEj
MCEGA1UECxMaSVRTTyBSYWxlaWdoIFROMzI3MCBTZXJ2ZXIxHTAbBgNVBAMTFElU
UO8uUkFMRUlHSC5JQkOuQO9NMB4XDTAyMDUxNTEzMTM1MFoXDTAzMDUxNjEzMTM1
MFowfjELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5DMRAwDgYDVQQHEwdSYWxlaWdo
MQwwCgYDVQQKEwNJQkOxIzAhBgNVBAsTGklUUO8gUmFsZWlnaCBUTjMyNzAgU2Vy
dmVyMROwGwYDVQQDExRJVFNPLlJBTEVJROguSUJNLkNPTTCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEAvuHpLXympFCoT1Q3jZ5E+EveDyued1RUo+BgCWOEErB/
6rKLrnlVwEU8w/nnTyApBW19IEbITrJ3YFGa4tIJVlleCQpGj5yQJNyPj6MIYOzv
9xDD8TgJu61zciJWLN6cnC7sygHiC+gEhCVAs+LR2wspfzOv8ebQZpujQDr2ZTOC
AwEAATANBgkqhkiG9w0BAQQFAAOBgQCzEB3UQoAUrAsyCYFrZwxRgS86RiNHU5iA
F+eHCwDSHzo6JI1q2l/JklbH/A4d9/ftNOrHOTS4rDl/U/izJR7tMSbJ/7kSAeZj
NXJDgQOIjpkMyZS8FQR9+BCRTD9EhDmaJGzxPxQ7UOF9KthOc87NkMKaO6BNhUml
5DzQ9Vjiag==
-----END CERTIFICATE-----
FOCAS @ SC63:/>
```

Figure C-21 An exported certificate from the gskkyman utility

3. FTP the file exported in step 2 to the client that will be using it.

   Any FTP client at the workstation can be used for this transfer. Ensure that the transfer type is *text* and that you can view the file at the workstation as a text file.

Figure C-22 Personal Communications client certificate management window

4. Import the certificate into the client's key database.

At the client, the certificate received from step 3 must be imported into the key database as a trusted certificate. Depending on the type of client, there are a number of different ways to do this. In the case of a Windows Personal Communications client (a TN3270 client), select the **Certificate Management** or **Certificate Wizard** icon from the Utilities folder. This displays the window (after the key database file is opened) shown in Figure C-22. Now import the certificate by clicking the **Add** button. When the certificate is added, the window shown in Figure C-23 shows the detail display from the certificate, showing the key size of 1024 (set by **gskkyman** when creating the certificate in Figure C-20 on page 390), the certificate version, and the *Issued To* name the same as the *Issued By* (this is a self-signed certificate).

Figure C-23 Personal Communications client: Display of imported self-signed server certificate

Now the client should be able to connect to the server. When the server passes its certificate to the client during the SSL exchange, the client will be able to validate it using the same certificate that is now stored in the client's key database as a CA certificate.

Self-signed client certificate gskkyman procedure

It is assumed that you have set up a key database (tn32v1r8.kdb) and the server's certificate is in the key database. The steps to implement client authentication in a **gskkyman** environment are basically the same as in the RACDCERT environment, except that the certificates are stored in a z/OS UNIX key database rather than the RACF database. These steps are:

1. Get the client certificate. For a self-signed certificate, this is normally generated at the client end.

   Because different client programs have different ways to generate a client certificate, this should be thought of as a generic example. Most clients will have some way to generate a certificate. In the case of Personal Communications, which provides a TN3270 client, use the Windows menu (click **Start** → **Programs** → **IBM Personal Communications** →

Utilities → **Certificate Management**) to open the client's key database. Then click **Create** → **New Self-signed Certificate** to generate the certificate. See Figure C-24 for an example of a self-signed client certificate that has just been generated by Personal Communications into the client key database.

Figure C-24 Personal Communications client: Newly added self-signed client certificate

2. Export the client certificate to a file.

   Most certificate management utilities allow the export of a certificate in at least two formats. The most common is Base64-encoded ASCII, but there is also the binary DER format. The choice depends on what your certificate management utility at the server end can use as an import format. We will use Base64 ASCII format. At the lower right-hand side of the window shown in Figure C-24 is the Extract Certificate... button. This is used to write a certificate to a file. The window that is presented is shown in Figure C-25 on page 394. Note that the Data Type field specifies Base64-encoded ASCII and that the certificate will be written to the cert.arm file.

Figure C-25 Exporting a client certificate to a file

3. FTP the certificate file from the client to the z/OS server side.

   This does not need to be shown here. You just need to FTP the certificate file created in step 2 (cert.arm in this example) to a z/OS UNIX file at the server side as a text file.

4. Add the client certificate into the server's z/OS UNIX key database using **gskkyman**.

   Figure C-26 on page 395 shows **gskkyman** being used to open the existing database in preparation for importing the client's certificate.

```
FOCAS @ SC63:/>gskkyman


            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 2
Enter key database name or press ENTER for "key.kdb": tn32v1r8.kdb
Enter password for the key database.......>


            Key database menu

Current key database is /tn32v1r8.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu):
```

Figure C-26 Opening the key database

Now that the key database is opened, you are presented with a list of options. Choose option **6 Store a CA certificate** to receive a certificate and store it as a trusted CA certificate.

```
            Key database menu

Current key database is /tn32v1r8.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 6
Enter certificate file name or press ENTER for "cert.arm": cert.arm
Enter a label for this key................> TN3270 Client Cert for User1

Please wait while certificate is stored...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 1
FOCAS @ SC63:/>
 ===>
```

Figure C-27 Adding a self-signed client certificate as a CA certificate using gskkyman

Figure C-27 shows the dialog between the user and **gskkyman** to add the client certificate as a CA certificate. The client certificate file in the example is named *cert.arm* and was placed there in the FTP transfer in step 3.

Now the client should be able to connect to the server (assuming that you have followed the steps to obtain and store the server's certificate). The server will be able to validate the client certificate with its copy of the client's certificate in the server's key database.

Internal Certificate Authority

One possibility in setting up your certificate management scheme is to set up as a CA. This means that you will be signing digital certificates for other entities, and anybody who uses the certificates that you sign will have to have a copy of your certificate in their key databases as a trusted CA.

You might choose to be a CA if you have multiple SSL/TLS-enabled servers in your system. When you have more than one server with its own self-signed certificate, each certificate must be exported to the clients that will use them. Therefore, if you had an FTP server, a TN3270 server, and an LDAP server, each using self-signed certificates and all being used from one workstation, that workstation will need all three certificates in its key database. With an internal CA, you can sign each of the server's certificates, and export the one certificate (the internal CA certificate) to the clients. Note that this assumes that a client's key database can be shared between client programs. This is mostly not the case, but a saving can still be made in that only the one key needs to be distributed.

In this section, we cover setting up an internal CA and creating and signing a server certificate with that CA certificate. Only one example is given, that is signing a server certificate and using RACDCERT for storing the certificates. You will see that the process is similar to the self-signing process described in “Self-signed certificates” on page 380, except that the certificate being distributed to the clients is the CA certificate, not the server certificate.

Internal-CA signed server certificate RACDCERT procedure

This procedure is basically the same for any z/OS server. In this example, we are producing a certificate for use by a TN3270 server. This certificate is needed by TN3270 clients using SSL. For the client to validate the server certificate, the internal CA certificate will be needed at the client:

1. Create a self-signed certificate for the local (internal) CA, as shown in Figure C-28.

```
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add the top-level self-signed certificate for the certificate
//*   authority (ourselves)
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  racdcert certauth gencert                        -
     subjectsdn( o('IBM Corporation')              -
                ou('ITSO Certificate Authority')   -
                 C('US'))                          -
      WITHLABEL('ITSO Certificate Authority')
/*
```

Figure C-28 Batch job to create internal CA certificate in RACF database

2. Generate a certificate for the server, as shown in Figure C-29.

```
//SERVCRT  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* set up the TN3270 server certificate, and sign it with the
//*   self-signed certificate-authority certificate set up previously
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(STC) GENCERT SUBJECTSDN(CN('ITSO.RALEIGH.IBM.COM') -
         O('IBM Corporation')                                    -
        OU('ITSO Raleigh TN3270 Server')                         -
         C('US'))                                                -
 WITHLABEL('TN3270 Server')                                      -
  SIGNWITH(CERTAUTH LABEL('ITSO Certificate Authority'))
/*
```

Figure C-29 Batch job to create server certificate and sign with internal CA certificate

Note that the RACDCERT SIGNWITH parameter specifies the LABEL of the internal CA certificate we set up in step 1. This indicates that the server certificate should be digitally signed with the internal CA’s private key. The ID(STC) parameter is used in this example because the TN3270 server, for whom the certificate is being generated, is associated with RACF user ID *STC*. Make sure that the common name (CN) is the same as the host or domain name of the server.

3. Create a RACF key ring for the server.

```
//KEYRING  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add a new Keyring to the TN3270 servers RACF ID (STC), then....
//* Add TN3270 server certificate to the user 'STC's keyring. the
//*   Keyring name is from the TN3270 configuration statement as below
//*      'KEYRING SAF TN3270Ring'
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(STC) ADDRING(TN3270Ring)
  RACDCERT ID(STC) CONNECT(CERTAUTH              -
                           LABEL('ITSO Certificate Authority') -
                           RING(TN3270Ring)      -
                           USAGE(CERTAUTH))
  RACDCERT ID(STC) CONNECT(ID(STC)               -
                           LABEL('TN3270 Server') -
                           RING(TN3270Ring)      -
                           DEFAULT               -
                           USAGE(PERSONAL))
/*
```

Figure C-30 Batch job to add a key ring

Figure C-30 shows the three steps necessary to create the key ring for the server:

a. Create a new RACF key ring using the RACDCERT ADDRING command.

b. Connect the internal CA certificate to the new key ring using the RACDCERT CONNECT command.

c. Connect the server's certificate (which was signed by the internal CA certificate) to the new key ring using the RACDCERT CONNECT command.

4. Export the internal CA certificate to an MVS data set.

```
//EXPORT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Export the Self-signed Certificate Authority certificate from the
//*   RACF database in base-64 encoded format. This is then FTP'd to
//*      the TN3270 client so that it can verify the TN3270 server's
//*      certificate when passed in the SSL exchange.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT CERTAUTH EXPORT(label('ITSO Certificate Authority')) -
              FORMAT(CERTB64) DSN('FOCAS.RACDCERT.SERVCERT')
/*
```

Figure C-31 Batch job to write the internal CA certificate to an MVS data set

Figure C-31 shows the RACDCERT EXPORT command being used to export the internal CA certificate to an MVS data set. The MVS data set will be ASCII sent via FTP to any client that needs to use that certificate to validate a server (in this case, a TN3270 client). One way to reduce the number of file transfers to clients is for them to pick up their key databases from a LAN drive, if practicable.

5. FTP the certificate exported in step 4 to the client that will use it.

This step is not shown, because any FTP client will be able to perform this step.

Figure C-32 Personal Communications client certificate management window

6. At the client, the certificate received from step 5 must be imported into the key database as a trusted certificate.

   Depending on the type of client, there are a number of different ways to do this. In the case of a Windows Personal Communications client (a TN3270 client), select the **Certificate Management** or **Certificate Wizard** icon from the Utilities folder. This displays the window (after the key database file is opened) in Figure C-32. Now import the certificate using the **Add** button. When the certificate is added, the window shown in Figure C-33 on page 400 shows the detail display from the certificate, showing the key size of 1024 (set by default in the RACDCERT GENCERT command), the certificate version, and the Issued To name the same as the Issued By (this is a self-signed certificate for a CA).

Figure C-33 Personal Communications client: Display of imported server certificate

7. Test the client-to-server connection.

   The Personal Communications client was instructed to connect to the TN3270 server using SSL. Figure C-34 on page 401 shows Personal Communications displaying the server certificate (by clicking **Communication** → **Security** → **Server**), which shows that the certificate subject is the TN3270 server, and the certificate issuer is the internal CA set up in step 1.

Figure C-34 Personal Communications client: Server certificate and signer's details

This discussion showed certificates being generated for a TN3270 server, and the internal CA certificate being exported to the client, then placed in the client's key database as a trusted certificate; the procedure for any client/server is basically the same.

External Certificate Authority

The aim of this section is to show how to use the TSO RACDCERT command and the UNIX **gskkyman** command to store and use CA-signed certificates. For the purposes of the examples, it is assumed that the server is on z/OS and the client is not.

The following step-by-step examples are generic in nature. They can be used to create a z/OS UNIX key database or RACF certificate/key ring for the IBM HTTP Server for z/OS, the TN3270 server, or other servers that are SSL enabled.

The procedure to use RACDCERT to request and manage a CA-signed server certificate is shown in "External CA-signed server certificate RACDCERT procedure" on page 402.

The procedure to use RACDCERT to request and manage a CA-signed client certificate is shown in "External CA-signed client certificate RACDCERT procedure" on page 407. The procedure to use **gskkyman** to request and manage a CA-signed server certificate is shown in "External CA-signed server certificate gskkyman procedure" on page 403. The procedure to use **gskkyman** to request and manage a CA-signed client certificate is shown in "External CA-signed client certificate gskkyman procedure" on page 413.

External CA-signed server certificate RACDCERT procedure

This section presents the steps required to implement the SSL environment for IBM HTTP Server with a server certificate signed by a public CA. A similar procedure can be used for other SSL-enabled application servers. The assumption in the examples is that the Web server's RACF user ID is "WEBSRV".

1. Generate a self-signed certificate.

   We will use this certificate as a base for the certificate request we will be creating.

   ```
   RACDCERT  ID(WEBSRV)  GENCERT
         SUBJECTDSN(CN('itso.raleigh.ibm.com')
                 O('IBM Corporation')
                 OU('ITSO Raleigh Webserver')
                 C('US'))
         WITHLABEL('Web Server Certificate')
   ```

   Make sure that the common name (CN) is the same as the host or domain name of the server.

2. Create a certificate request for the CA.

   The certificate request will be stored in an MVS data set named BOCHE.WEBSERV.GENREQ.

   ```
   RACDCERT  ID(WEBSRV)  GENCERT
         GENREQ(LABEL('Web Server Certificate'))
         DSN('BOCHE.WEBSERV.GENREQ')
   ```

   This certificate request needs to be sent to the CA. The format of the request is Base64-encoded text. The data set can be transmitted to a PC with FTP and pasted into the appropriate field in the certificate request. Alternatively, cutting and pasting between a host emulator window and the Web browser can be used.

3. Store the returned certificate in an MVS data set.

   The CA usually returns the certificate using e-mail or similar means. The certificate is in Base64-encoded text format. Use FTP to transfer the certificate received from the CA into an MVS data set named, for instance, BOCHE.WEBSERV.CERT.

4. Replace the self-signed certificate with the certificate received from and signed by the CA.

   ```
   RACDCERT  ID(WEBSRV)
         ADD('BOCHE.WEBSERV.CERT')
         WITHLABEL('Web Server Certificate')
   ```

5. Create a key ring for the server.

   This key ring must not already exist for this user. Key ring names become names of RACF profiles in the DIGTRING class, and can contain only characters that are allowed in RACF profile names. Although asterisks (*) are allowed in key ring names, a single asterisk is not allowed.

   ```
   RACDCERT ID(WEBSRV) ADDRING(WEBSERVER)
   ```

6. Connect the certificate to the key ring.

   Now we can create the connection between the digital certificate and the key ring with the RACDCERT CONNECT command and associate it with the HTTP started task user ID.

   ```
   RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Web Server Certificate')
            RING(WEBSERVER) DEFAULT USAGE(PERSONAL))
   ```

External CA-signed server certificate gskkyman procedure

The following step-by-step example shows how to create a key database in z/OS UNIX and how to create certificate requests for external CAs for the IBM HTTP Server for OS/390. A similar procedure may be used for other server applications, including the TN3270 server and the Policy Agent.

In our test environment, we elected to have our certificate issued by the public CA company, VeriSign. In the following discussion, we show how we created the certificate request for our server certificate, and how we received it into the key database.

It is assumed that you have set up a key database and produced a stash file for the server, as discussed in “gskkyman command use” on page 377; that you have set the correct UNIX permission bits so that the server can read the files; and that the database is named /u/takada/sslkey/server.kdb.

1. Create a public-private key pair and certificate request.

   Figure C-35 shows the menu screen of the **gskkyman** utility. To create a public-private key pair and a certificate request, select **3 - Create new key pair and certificate request** from this screen.

```
Key database menu

Current key database is /u/takada/sslkey/server.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 3 1
Enter certificate request file name or press ENTER for "certreq.arm": server.arm 2
Enter a label for this key...............> ITSO Raleigh Webserver Cert 3
Select desired key size from the following options (512):
    1:    512
    2:    1024
Enter the number corresponding to the key size you want: 1 4
Enter certificate subject name fields in the following. 5
Common Name (required)................> mvs03c.itso.ral.ibm.com
    Organization (required)...............> IBM Corp.
    Organization Unit (optional)..........> ITSO Raleigh
    City/Locality (optional)..............> Research Triangle Park
    State/Province (optional).............> North Carolina
    Country Name (required 2 characters)..> US

Please wait while key pair is created...

Your request has completed successfully, exit gskkyman? (1 = yes, 0 = no) [0]: 0
```

Figure C-35 Create a new key pair and certificate request

1 Select option **3 - Create new key pair and certificate request** to create a new key pair and certificate request. If you want to create a self-signed certificate, select option **5 - Create a self-signed certificate**.

2 Specify a file name for the certificate request. Later you have to send the contents of this file to the CA you selected.

3 Enter a label related to this key and certificate.

4 Select the key size that you want. However, the key size depends on the location. In our test environment (ITSO Raleigh), we installed the strong crypto version of the **gskkyman** utility. In almost all cases, you would want to install the strong crypto version and use a key size of 1024 bits.

5 Enter the certificate subject name fields. Common Name should be your server's host name. If you specify another name, a user will receive the window shown in Figure C-36 when accessing this server via a browser.

Figure C-36 Netscape Navigator's window checking certificate name

The only way a Web browser can check the server's identity is to compare the host name in the URL with the host name in the Common Name attribute in the certificate. If they do not match, Netscape Navigator will display the warning window shown in Figure C-36. Whether Internet Explorer® (IE) will display a warning window or simply terminate the connection depends on the release level of IE and the security level chosen.

2. Request the certificate from the CA.

   After step 1, you have three files in addition to the key database:

   – A certificate request file (*.arm)
   – A stash file (*.sth)
   – A key pair file (*.rdb)

   Figure C-37 on page 405 shows the contents of the certificate request file. You send this request to the CA to be signed. We sent this certificate request to VeriSign. As shown in Figure C-38 on page 405, you can copy and paste the contents of the request file into the VeriSign form.

```
TAKADA @ RA03:/u/takada/sslkey>ls -l
total 192
rw-r--r--   1 TAKADA   OMVSGRP      513 Aug  6 18:56 server.arm
-rw-r--r--   1 TAKADA   OMVSGRP    65080 Aug  6 18:54 server.kdb
-rw-r--r--   1 TAKADA   OMVSGRP     5080 Aug  6 18:56 server.rdb
-rw-------   1 TAKADA   OMVSGRP      129 Aug  6 18:56 server.sth
TAKADA @ RA03:/u/takada/sslkey>cat server.arm
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBNTCB4AIBADB7MQswCQYDVQQGEwJVUzELMAkGA1UECBMCTkMxDTALBgNVBAcT
BENhcnkxGDAWBgNVBAoTD0lCTSBDb3Jwb3JhdGlvbjEUMBIGA1UECxMLSVRTTyBS
YWxpZ2gxIDAeBgNVBAMTF212czAzYy5pdHNvLnJhbC5pYm0uY29tMFwwDQYJKoZI
hvcNAQEBBQADSwAwSAJBAJaDyGjFOxIvb3FXm68t66tDQ+dn9B/zLthCS7dc7nor
KT6YpfjnI7duvw/zXXMrrJP99y4oLIGafHIZq1qAHoOCAwEAAaAAMA0GCSqGSIb3
DQEBBAUAA0EAc7OFskVCHrzZXkyoIa6NnDdrtt6CHhMKLJKtIitStFPXZVIMQxPK
1ER2vdsdzpQtIqgTromX2Jf4l4qm47gcWA==
-----END NEW CERTIFICATE REQUEST----
```

Figure C-37 The content of the certificate request file

Figure C-38 Certificate request submit form at VeriSign Web site

After you send a certificate request to an external CA, it can take several days before the request is processed and the certificate is returned. We used the trial Secure Server ID from VeriSign to test the IBM HTTP Server for z/OS SSL function in the ITSO Raleigh test environment. This provides a temporary certificate that is valid for two weeks from the date of issuance. Because the certificate is temporary, it does not require the extensive checking that a real certificate would, therefore we received it almost immediately after submitting the certificate request.

While you are waiting for the CA to process your certificate request, it is a good idea to exploit a trial certificate to test SSL sessions. Alternatively (or in addition), you can use the **gskkyman** utility to create a self-signed certificate to enable SSL sessions between clients and the server. For detailed information regarding how to make a self-signed certificate, see *HTTP Server Planning, Installing and Using, V5.3*, SC31-8690, or *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074.

3. Receive the certificate from the CA.

   After receiving a certificate from the CA via e-mail, copy and paste it to a z/OS UNIX file. In Figure C-39, we created a file server.cert and put our certificate into this file.

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       /u/takada/sslkey/server.cert                   Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 -----BEGIN CERTIFICATE-----
000002 MIICJTCCAc8CEA35fK/Qad35oFktNSeSsS8wDQYJKoZIhvcNAQEEBQAwgakxFjAU
000003 BgNVBAoTDVZlcmlTaWduLCBJbmMxRzBFBgNVBAsTPnd3dy52ZXJpc2lnbi5jb2Ov
000004 cmVwb3NpdG9yeS9UZXNOQ1BTIEluY29ycC4gQnkgUmVmLiBMaWFiLiBMVEQuMUYw
000005 RAYDVQQLEz1Gb3IgVmVyaVNpZ24gYXVOaG9yaXplZCBOZXNOaW5nIG9ubHkuIE5v
000006 IGFzc3VyYW5jZXMgKEMpVlMxOTk3MB4XDTk5MDgwNjAwMDAwMFoXDTk5MDgyMDIz
000007 NTk1OVowgYExCzAJBgNVBAYTAlVTMRcwFQYDVQQIEw5Ob3JOaCBDYXJvbGluYTEN
000008 MAsGA1UEBxQEQ2FyeTESMBAGA1UEChQJSUJNIENvcnAuMRQwEgYDVQQLFAtJVFNP
000009 IFJhbGlnaDEgMB4GA1UEAxQXbXZzMDNjLml0c28ucmFsLmlibS5jb2OwXDANBgkq
000010 hkiG9wOBAQEFAANLADBIAkEA4P/8r7jWD27V1XWTP1l2GgOqcakpxrTaXZ78x/Sr
000011 EMydBymOnxhrRzK21DFbpTlbM9mT+juOav9mKiUxf19WswIDAQABMAOGCSqGSIb3
000012 DQEBBAUAAOEAr2OtpJvdpN4NcR6Lzx3eBGUZ4VtwtwkvKeU2AU6N9/JXOMGS2r+m
000013 IckUeu4+pRF+cHZY8uLjL1hA+cOBux4RKA==
000014 -----END CERTIFICATE-----
****** **************************** Bottom of Data ****************************

 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel .
```

Figure C-39 The content of the server certificate issued by the trusted CA

   Because the **gskkyman** utility accepts only z/OS UNIX files, you have to create a z/OS UNIX file for your certificate.

4. Import the CA-signed server certificate into the key database.

Figure C-40 shows **gskkyman** being used to import the certificate into your key database.

```
 IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number: 2 1
Enter key database name or press ENTER for "key.kdb": server.kdb
Enter password for the key database.......>******

            Key database menu

Current key database is /u/takada/sslkey/server.kdb

     1  - List/Manage keys and certificates
     2  - List/Manage request keys
     3  - Create new key pair and certificate request
     4  - Receive a certificate issued for your request
     5  - Create a self-signed certificate
     6  - Store a CA certificate
     7  - Show the default key
     8  - Import keys
     9  - Export keys
    10  - List all trusted CAs
    11  - Store encrypted database password

     0  - Exit program

Enter option number (or press ENTER to return to the parent menu): 4 2
Enter certificate file name or press ENTER for "cert.arm": server.cert 3
Do you want to set the key as the default in your key database? (1 = yes, 0 = no) [1]:
1 4
Please wait while certificate is received.......
```

Figure C-40 Store the server certificate into the key database

1. Open the key database. Because the **gskkyman** command was entered from the subdirectory /u/takada/sslkey, there was no need to specify the subdirectory in the key database name.
2. Select option **4 - Receive a certificate issued for your request** to store your server certificate file.
3. Specify your server certificate file created in Figure C-39 on page 406.
4. You have to select 1. If you set another key as the default, server authentication will fail.

External CA-signed client certificate RACDCERT procedure

The procedure for a client to request a CA-signed certificate is dependent on the type of client software being used. Most SSL/TLS-enabled clients will have a method to create a file with a certificate request for submission to a CA.

A client certificate, in addition to being stored in the client's key database, must be added to RACF and associated with the appropriate RACF user ID using the RACFDCERT ID(clients-user ID) ADD.... statement. The client certificate's CA must be connected to the server's RACF key ring using the RACFDCERT ID(servers-user ID) CONNECT. The basic procedure to follow is:

1. Generate a certificate request at the client.

   Clients have different ways to generate a certificate request from an external CA. For this example, we will be requesting a client certificate for a Personal Communications (TN3270) client. Figure C-41 shows the Personal Communications window to request a certificate.

Figure C-41 Using a client to request a certificate

See Figure C-42 for an example of the certificate request file that is created.


```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqDCCARECAQAwaDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5DMRAwDgYDVQQH
EwdSYWxlaWdoMRgwFgYDVQQKEw9JQkOgQ29ycG9yYXRpb24xDTALBgNVBAsTBElU
UO8xETAPBgNVBAMTCGOyMzhwNHlrMIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQC81RfEw/spXrdZY/eSES6kFkrI+BvOllVhYQ+X/+lgsA/Bbb85e75hsPAHU/+q
xeDC2JDqJrjPIChbwxbOMRofxwYhSpu51grQJIYYMehbWlmz9BvF3V+I8SV2fp+A
uPXtjwl7cTC1tNO+mbBnlxgYVDaygOgkh8XhlM4QMderIwIDAQABoAAwDQYJKoZI
hvcNAQEEBQADgYEAHdJbb3R3i7a2WJgQKnl+TdbeJxX9D8bdufXfzwCRckLqBPNi
kVeh6Hg5z+UeLX7O+Cr3TsPmYJHAXZYQCNATCsHIRj1p5XC5OVDrckEG/RpVLvfO
36Y2fYOT4f86sOy8L2RwhRSm3V2mC5vG9JjlB1MS2hkQ13ZWFkYrFMvwczo=
-----END NEW CERTIFICATE REQUEST-----
```


Figure C-42 Certificate request generated by client for external CA

2. Send the certificate request to the CA.

E-mail the certificate request output from the client, either by using cut and paste, or as an attachment. The CA may take a number of days to generate and send the certificate and private key back to you.

3. Receive the certificate from the CA.

   Depending on the CA, you may have to go to a secure Web site to download your client certificate and key or they may send it to you in a secure e-mail. Whatever method is chosen, you must end up with a file, probably in PKCS12 format, which contains both a digital certificate and a private key. This file will be password protected.

4. Import the client certificate (and private key) into the client key database.

   In step 3, the certificate and key were received and detached into a workstation file. That certificate and key must now be imported into your client's key database. As an example, we show how to import the client certificate and key into the Personal Communications key database.

   First, start the Certificate Management utility. Click **Start** → **Programs** → **IBM-Personal-Comm** → **Utilities** → **Certificate Management**.

   Open the Personal Communications key database and give the password (the default installation password is *PCOMM*).

Figure C-43 Personal Communications certificate management window after key database is opened

Figure C-43 shows the window that is displayed after the Personal Communications key database is opened. The highlighted certificate is the server certificate that we have already assumed to have been set up.

Just below the heading `Key database content`, there is a drop-down box containing the words `Signer Certificates`, so the certificates listed are the CA certificates. Click the drop-down box and select **Personal Certificates**. You will then be presented with a list of personal certificates. If you have never imported any, the list will be blank. In any case, then click the **Import** button and select the certificate file that you exported in step 1, and click **OK**. You will be asked for a password, which will have been provided by your CA.

Figure C-44 Personal Communications certificate dialog after importing a client certificate and key

Figure C-44 shows the window after you have imported the key/certificate into the key database. As the file received from the CA not only contained the client certificate, but also the signer's certificate, the import function added two certificates to the client key database, one in the Personal Certificates section and one in the Signer Certificates section, as indicated in the informational window.

5. Export the client certificate to a workstation file.

   This step is not shown for brevity. It is a matter of highlighting the client certificate in the Personal Certificates section and clicking the **Export/Import...** button, then following the instructions to create a Base64-encoded ASCII certificate file. Note that the file received from the CA should not be sent to the server, because it contains the client's private key. That is why it is first imported into the client's key database; then just the certificate is exported, not the private key.

   Note: This same step might have to be followed for the CA certificate as well. If the CA that signed your client certificate does not have a CA certificate at the server's key database, then the server has no way of validating the client certificate. You will need to export the CA certificate, add it into the RACF database, and connect it as a CA certificate into the server's key ring. Use the RACDCERT CERTAUTH LIST command to see all CA certificates in your system.

6. FTP the client (and CA certificate if needed as per the note in step 4) exported in step 5 to the z/OS system as MVS data sets.

   This step does not need to be shown. Ensure that the transfer is a text-type transfer to enable ASCII/EBCDIC translation.

   Important: The RACF RACDCERT command requires a certificate file that it will be importing to be in variable blocked format. To do this from a workstation FTP client, use the **`quote site recfm=vb`** command or, if using a GUI FTP client, consult the help files.

7. Add the client certificate into the RACF database as a TRUSTED certificate for the RACF ID that you want to associate the certificate with (the user of the workstation).

   In our case, the client certificate was issued for user ID *FOCAS*, and the FTP in step 6 transferred the client certificate to an MVS data set named FOCAS.RACDCERT.THAWTE.CLIENT.CERT.

```
//CLIENT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Import the Client certificate issued by Thawte, and set the
//*   TRUST flag on.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(FOCAS) ADD('FOCAS.RACDCERT.THAWTE.CLIENT.CERT') -
               WITHLABEL('Thawte Certificate for FOCAS') TRUST
/*
```

Figure C-45 Batch job to import client certificate and associate with a RACF user ID

   Figure C-45 shows the RACDCERT ADD command being used to add the client certificate issued by the CA into the RACF database. The TRUST flag is set on, and the ID(FOCAS) parameter associated this certificate with the user ID FOCAS. This certificate does not need to be added to the server's key ring.

8. Connect the CA certificate of the client certificate's issuer into the RACF database and connect it to the server's key ring.

   If the CA certificate is already in the RACF database, the ADD step can be skipped, but you must still connect the CA certificate to the server's key ring with USAGE(CERTAUTH), which sets the TRUST status on for use in the server's key ring.

```
//CACERT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Import the CA Certificate that signed the client certificate
//*   into the RACF database. This certificate was FTP'd from the
//*   TN3270 client PCOMM key database from the 'Servers Certifictes'
//*   section.If the CA certificate is already in the RACF database
//*   the 'ADD' step can be skipped. You must still add the CA
//*   certificate to the servers keyring with USAGE(CERTAUTH)
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT CERTAUTH ADD('FOCAS.RACDCERT.FREEMAIL.RSA.CERT') -
              WITHLABEL('Thawte Freemail RSA 2000') notrust
  RACDCERT ID(STC) CONNECT(CERTAUTH              -
                           LABEL('Thawte Freemail RSA 2000') -
                           RING(TN3270Ring)      -
                           USAGE(CERTAUTH))
/*
```

Figure C-46 Batch job to import CA certificate and add to server's RACF key ring as a CA

Figure C-46 shows the RACDCERT ADD command being used to add the CA certificate into the RACF database. In this example, we added the certificate as a CA certificate (CERTAUTH on the ADD), but set the NOTRUST flag. This is overridden in any case when you connect the CA certificate to the server's key ring as USAGE(CERTAUTH). This was just an example to show how the TRUST status of a certificate can be overridden for a particular server's use by adding that certificate to the server's key ring as a CA certificate by specifying USAGE(CERTAUTH).

Table C-3 Example certificate locations

Certificate	z/OS RACF database	Personal Communications Client Key database
Client Certificate	ADDed with ID(FOCAS) status TRUST. If self-signed, must also CONNECT to server's key ring as USAGE(CERTAUTH).	In the Personal Certificates section as *default* certificate
Client Certificate's CA certificate	ADDed with CERTAUTH (trust status immaterial). CONNECT to server's key ring as USAGE(CERTAUTH).	In the Signer Certificates section marked as *trusted root*
Server Certificate	ADDed with ID(STC). CONNECT to server's key ring as USAGE(PERSONAL) and DEFAULT.	Not needed unless the server certificate is a self-signed, in which case it is added to the Signer Certificates section as a *trusted root*
Server Certificates CA certificate (non-self-signed server certificate only)	ADDed with CERTAUTH (trust status immaterial). CONNECT to server's key ring as USAGE(CERTAUTH).	In the Signer Certificates section as a *trusted root*

Table C-3 shows the various locations of the certificates for client authentication to happen. The assumption is that user FOCAS is the client user and STC is the RACF user ID associated with the server. If the server uses a self-signed certificate, ignore the Server

Certificate CA row. If the client uses a self-signed certificate, ignore the Client Certificate CA row.

> **Note:** The RACF database comes preconfigured with CA certificates from some major CAs. These companies are Thawte and VeriSign. If you delete these CA certificates from the RACDCERT database, they will be reinstated automatically after an IPL.

After these steps are followed, the connection can be established using SSL/TLS.

External CA-signed client certificate gskkyman procedure

The basic procedure to follow for **gskkyman** is:

- Steps 1 through 4 are exactly the same as those outlined in "External CA-signed client certificate RACDCERT procedure" on page 407:
  a. Generate a certificate request at the client.
  b. Send the certificate request to the CA.
  c. Receive the client certificate from the CA.
  d. Import the certificate into the client's key database.
- Step 5: When you use **gskkyman**, you do not store the client's certificate in the server's key ring. However, you need to ensure that the CA that issued the client certificate has a CA certificate in the server's key ring. This is needed so that the server can validate the client's certificate when the client presents it during the SSL handshake. The CA certificate should already be in the key database as a CA. The **gskkyman** database comes preconfigured with CA certificates from some major CAs such as Thawte and VeriSign. If your client certificate was issued by some other CA that does not have a CA certificate in the server's z/OS UNIX key database, you should export it from the client, where it would have been also added when you imported your certificate from the CA, and then add it to the **gskkyman** database by selecting the option **6 - Store a CA certificate**.

Certificate locations example

This section briefly summarizes the locations of certificates using each of the utilities.

RACF certificates

The following example might help you visualize where client and server certificates must be located in order for SSL/TLS to function correctly. Both user1 and user2 are clients of the FTP and TN3270 servers on z/OS. Both clients use a workstation-based TN3270 client. user1 uses a z/OS-based FTP client and user2 has a workstation FTP client.

Table C-4 Example list of certificates for RACF database

Owner	Certificate number	Signer	Description	Trusted?
ID(user1)	1	Self	FTP Client Certificate	Yes
ID(user1)	2	VeriSign	TN3270 Client Certificate	Yes
ID(user2)	3	VeriSign	FTP Client Certificate	Yes
ID(user2)	4	Self	TN3270 Client Certificate	Yes
ID(FTP)	5	VeriSign	FTP Server Certificate	Does not matter

Owner	Certificate number	Signer	Description	Trusted?
ID(TN3270)	6	Self	TN3270 Server Certificate	Does not matter
CERTAUTH	7	Self	VeriSign CA Certificate	Does not matter

Certificate table description

In Table C-4, the Owner column identifies the certificate owner. This is the parameter used on the RACDCERT command when you add, delete, or change this certificate entry.

user1 is an FTP and TN3270 client's RACF user ID. His FTP client is on z/OS and his TN3270 client is on a workstation. The FTP client has a self-signed certificate and the TN3270 workstation client has a certificate issued by a well-known CA, VeriSign.

user2 is an FTP and TN3270 client's RACF user ID. His FTP client and TN3270 client are on a workstation. The TN3270 client has a self-signed certificate and the FTP client has a certificate issued by a well-known CA, VeriSign.

FTP is the RACF user ID that the TLS-enabled FTP server on z/OS runs under. The FTP server has a server certificate issued by a well-known CA, VeriSign.

TN3270 is the RACF user ID that the SSL-enabled TN3270 server on z/OS runs under. The TN3270 server has a self-signed server certificate.

CERTAUTH is that part of the RACF database reserved for CA certificates. The VeriSign certificate is shown as being self-signed, although it could also be signed by a higher-level authority. For the purposes of this discussion, that is immaterial.

Key rings needed for example

If we discuss each user in turn, we can see where the digital certificates for the clients and the servers need to be located. We are assuming that both the TN3270 and FTP servers are configured for client authentication (server authentication is mandatory).

user1

Table C-5 RACF key ring connections needed for user1

Key ring name	Key ring association	Certificate number	Certificate association	Default	Usage()
FTPClientRing	ID(user1)	1 **1**	ID(user1)	Yes	PERSONAL
FTPClientRing	ID(user1)	7 **2**	CERTAUTH	No	CERTAUTH

user1 is an FTP and TN3270 client user. The FTP client is on z/OS and will be communicating with the z/OS-based TLS-enabled FTP server. Therefore, the client needs a key ring to store the certificates that will be used in the SSL exchange. Table C-5 shows the RACF key ring with example name FTPClientRing set up for the user. There are two certificates in the key ring:

1 The client certificate (certificate number 1 in Table C-4 on page 413) is needed by the client to pass to the FTP server when requested as part of client authentication. The client certificate must be in TRUSTED status and must be set to the DEFAULT certificate in the key ring. This is how the FTP client knows which certificate to pass in the TLS exchange.

2 The server that is being connected to has a server certificate (certificate number 5 in Table C-4 on page 413) signed by an external CA, VeriSign (certificate number 7 in Table C-4 on page 413). The CA certificate is needed by the client to validate the server certificate passed by the FTP server during the TLS exchange.

user1's TN3270 client is not on z/OS and therefore does not require a RACF key ring. However, the TN3270 client on the workstation will require its key database to contain the client certificate and a CA certificate to verify the TN3270 server (because the TN3270 server certificate is self-signed, the server certificate itself *is* the CA certificate).

user2

user2 is an FTP and TN3270 client user, with both clients on a workstation. Therefore, user2 does not require a RACF key ring. However, the workstation key databases of both clients need the following:

- The TN3270 client on the workstation will require its key database to contain user2's TN3270 client certificate and a CA certificate to verify the TN3270 server (because the TN3270 server certificate is self-signed, the server certificate itself *is* the CA certificate).
- The FTP client on the workstation will require its key database to contain the user2's FTP client certificate and a CA certificate to verify the FTP server (because the FTP server certificate was signed by VeriSign, you should ensure that the VeriSign CA certificate is present and set as TRUSTED).

FTP

Table C-6 RACF key ring connections needed for user FTP

Key ring name	Key ring association	Certificate number	Certificate association	Default	Usage()
FTPServerRing	ID(FTP)	5 1	ID(FTP)	Yes	PERSONAL
FTPServerRing	ID(FTP)	1 2	ID(user1)	No	CERTAUTH
FTPServerRing	ID(FTP)	7 3	CERTAUTH	No	CERTAUTH

FTP is the RACF user ID that the z/OS-based FTP server runs under. Any TLS-enabled server requires a key ring to refer to certificates that it will be using in the TLS exchange. A certificate is required for the server, and a certificate is required for every different CA that has issued the FTP client certificates. Table C-6 shows the RACF key ring with example name FTPServerRing. There are three certificates in the key ring:

1 When the server sends its certificate to the client at the beginning of the TLS handshake, it looks in the key ring (whose name is in the server's configuration file) for a DEFAULT certificate. That is the server's certificate that is passed to the client (certificate number 5 in Table C-4 on page 413).

2 This is the CA certificate to validate user1. Because user1 had a self-signed client certificate for FTP, the client certificate (certificate number 1 in Table C-4 on page 413) is being used here in the context of a CA certificate, that is, it will be used to validate the client certificate when it is passed to the server.

3 This is the CA certificate to validate user2. Because user2 had a client certificate for FTP issued by an external company, VeriSign, this is the VeriSign CA certificate (certificate number 7 in Table C-4 on page 413).

TN3270

Table C-7 List of RACF key rings needed for example

Key ring name	Key ring association	Certificate number	Certificate association	Default	Usage()
TN3270ServerRing	ID(TN3270)	6 1	ID(TN3270)	Yes	PERSONAL
TN3270ServerRing	ID(TN3270)	4 2	ID(user2)	No	CERTAUTH
TN3270ServerRing	ID(TN3270)	7 3	CERTAUTH	No	CERTAUTH

TN3270 is the RACF user ID that the z/OS-based TN3270 server runs under. Any SSL-enabled server requires a key ring to refer to certificates that it will be using in the SSL exchange. A certificate is required for the server, and a certificate is required for every different CA that has issued the TN3270 client certificates. Table C-7 shows the RACF key ring with example name TN3270ServerRing. There are three certificates in the key ring:

1. When the server sends its certificate to the client at the beginning of the SSL handshake, it looks in the key ring (whose name is in the server's configuration file) for a DEFAULT certificate. That is the server's certificate that is passed to the client (certificate number 6 in Table C-4 on page 413).
2. This is the CA certificate to validate user2. As user2 had a self-signed client certificate for TN3270, the client certificate (certificate number 4 in Table C-4 on page 413) is being used here in the context of a CA certificate, that is, it will be used to validate the client certificate when it is passed to the server.
3. This is the CA certificate to validate user1. Because user1 had a client certificate issued by an external company, VeriSign, this is the VeriSign CA certificate (certificate number 7 in Table C-4 on page 413).

gskkyman z/OS UNIX certificates

The following example might help you visualize where client and server certificates must be located in order for SSL/TLS to function correctly. Both user1 and user2 are clients of the FTP and TN3270 servers on z/OS. Both clients use a workstation-based TN3270 client. user1 uses a z/OS-based FTP client and user2 has a workstation FTP client.

Table C-8 Example list of certificates for examples to follow

Certificate subject	Certificate number	Signer	Description	Trusted?
user1	1	Self	FTP Client Certificate	Yes
user1	2	VeriSign	TN3270 Client Certificate	Yes
user2	3	VeriSign	FTP Client Certificate	Yes
user2	4	Self	TN3270 Client Certificate	Yes
FTP	5	VeriSign	FTP Server Certificate	No
TN3270	6	Self	TN3270 Server Certificate	Yes
VERISIGN	7	Self	VeriSign CA Certificate	Yes

Certificate table description

Table C-8 on page 416 shows a list of certificates that we will be using to set up some different key databases. A key database will be set up for the RACF user IDs user1, FTP, and TN3270.

user1 is an FTP and TN3270 client's RACF user ID. His FTP client is on z/OS and his TN3270 client is on a workstation. The FTP client has a self-signed certificate and the TN3270 workstation client has a certificate issued by a well-known CA, VeriSign.

user2 is an FTP and TN3270 client's RACF user ID. His FTP client and TN3270 client are on a workstation. The TN3270 client has a self-signed certificate and the FTP client has a certificate issued by a well-known CA, VeriSign.

FTP is the RACF user ID that the TLS-enabled FTP server on z/OS runs under. The FTP server has a server certificate by a well-known CA, VeriSign.

TN3270 is the RACF user ID that the SSL-enabled TN3270 server on z/OS runs under. The TN3270 server has a self-signed server certificate.

VERISIGN is an external CA certificate. The VeriSign certificate is shown as being self-signed, although it could also be signed by a higher-level authority. As long as the *trusted root* status is set, that is okay.

z/OS UNIX key databases needed for example

If we discuss each user in turn, we can see where the digital certificates for the clients and the servers need to be located. We are assuming that both the TN3270 and FTP servers are configured for client authentication (server authentication is mandatory).

user1

Table C-9 Key database needed for user1 to use FTP client on z/OS

Certificate subject	Certificate number	Signer	Description	Default?
user1	1 1	Self	FTP Client Certificate	Yes
VERISIGN	7 2	Self	VeriSign CA Certificate	No

user1 is an FTP and TN3270 client user. The FTP client is on z/OS and will be communicating with the z/OS-based TLS-enabled FTP server. Therefore, the client needs a key ring to store the certificates that will be used in the SSL exchange. Table C-9 shows the contents of the z/OS UNIX key database set up for user1. There are two certificates in the key database:

1 The client certificate (certificate number 1 in Table C-8 on page 416) is needed by the client to pass to the FTP server when requested as part of client authentication. The client certificate must be in TRUSTED status and must be set to the DEFAULT certificate in the key database. This is how the FTP client knows which certificate to pass in the TLS exchange.

2 The server that is being connected to has a server certificate (certificate number 5 in Table C-8 on page 416) signed by an external CA, VeriSign (certificate number 7 in Table C-8 on page 416). The CA certificate is needed by the client to validate the server certificate passed by the FTP server during the TLS exchange.

user1's TN3270 client is not on z/OS, therefore does not require a z/OS UNIX key database. However, the TN3270 client on the workstation will require its key database to contain user1's TN3270 client certificate and a CA certificate to verify the TN3270 server (because the TN3270 server certificate is self-signed, the server certificate itself *is* the CA certificate).

user2

user2 is an FTP and TN3270 client user, with both clients on a workstation. Therefore, user2 does not require a z/OS UNIX key database. However, the workstation key databases of the FTP and TN3270 clients need the following:

- The TN3270 client on the workstation will require its key database to contain user2's TN3270 client certificate and a CA certificate to verify the TN3270 server (because the TN3270 server certificate is self-signed, the server certificate itself *is* the CA certificate).
- The FTP client on the workstation will require its key database to contain the user2's FTP client certificate and a CA certificate to verify the FTP server (because the FTP server certificate was signed by VeriSign, you should ensure that the VeriSign CA certificate is present and set as TRUSTED).

FTP

Table C-10 Key database needed for FTP server on z/OS

Certificate subject	Certificate number	Signer	Description	Default?
FTP	5 **1**	VeriSign	FTP Server Certificate	Yes
user1	1 **2**	Self	FTP Client Certificate	No
VERISIGN	7 **3**	Self	VeriSign CA Certificate	No

FTP is the RACF user ID that the z/OS based FTP server runs under. Any TLS-enabled server requires a key database to refer to certificates that it will be using in the TLS exchange. A certificate is required for the server, and a certificate is required for every different CA that has issued the FTP client certificates. Table C-10 shows the contents of the z/OS UNIX key database set up for user FTP. There are three certificates in the key ring:

1 When the server sends its certificate to the client at the beginning of the TLS handshake, it looks in the key database (whose name is in the server's configuration file) for a DEFAULT certificate. That is the server's certificate that is passed to the client (certificate number 5 in Table C-8 on page 416).

2 This is the CA certificate to validate user1. Because user1 had a self-signed client certificate for FTP, the client certificate (certificate number 1 in Table C-8 on page 416) is being used here in the context of a CA certificate, that is, it will be used to validate the client certificate when it is passed to the server.

3 This is the CA certificate to validate user2. Because user2 had a client certificate for FTP issued by an external company, VeriSign, this is the VeriSign CA certificate (certificate number 7 in Table C-8 on page 416).

TN3270

Table C-11 Example list of certificates for examples to follow

Certificate subject	Certificate number	Signer	Description	Default?
TN3270	6 1	Self	TN3270 Server Certificate	Yes
user2	4 2	Self	TN3270 Client Certificate	No
VERISIGN	7 3	Self	VeriSign CA Certificate	No

TN3270 is the RACF user ID that the z/OS-based TN3270 server runs under. Any SSL-enabled server requires a key database to refer to certificates that it will be using in the SSL exchange. A certificate is required for the server, and a certificate is required for every different CA that has issued the TN3270 client certificates. Table C-11 shows the contents of the z/OS UNIX key database set up for user TN3270. There are three certificates in the key ring:

1 When the server sends its certificate to the client at the beginning of the SSL handshake, it looks in the key database (whose name is in the server's configuration file) for a DEFAULT certificate. That is the server's certificate that is passed to the client (certificate number 6 in Table C-8 on page 416).

2 This is the CA certificate to validate user2. Because user2 had a self-signed client certificate for TN3270, the client certificate (certificate number 4 in Table C-8 on page 416) is being used here in the context of a CA certificate, that is, it will be used to validate the client certificate when it is passed to the server.

3 This is the CA certificate to validate user1. Because user1 had a client certificate issued by an external company, VeriSign, this is the VeriSign CA certificate (certificate number 7 in Table C-8 on page 416).

D

IPSec scenario policies

This appendix contains the policies used for our IPSec scenarios. They are as follows:

- 3.5, “Implementing IPSec between two z/OS systems” on page 95
  - The policy for TCPIPA on SC30 is shown in Example D-1.
  - The policy for TCPIPA on SC31 is shown in Example D-2 on page 426.

Example: D-1 TCPIPA policy for SC30

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC30
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Fri Aug 25 11:49:00 EDT 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}

IpGenericFilterAction          IpSec~LogYes
{
  IpFilterAction               IpSec
  IpFilterLogging              Yes
}

KeyExchangeOffer               KEO~5
{
  HowToEncrypt                 DES
  HowToAuthMsgs                SHA1
  HowToAuthPeers               PresharedKey
  DHGroup                      Group1
```

© Copyright IBM Corp. 2007. All rights reserved.

```
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~6
{
  HowToEncap                   Transport
  HowToEncrypt                 3DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~7
{
  HowToEncap                   Transport
  HowToEncrypt                 AES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpService                      OMPROUTE-IP_V4
{
  Protocol                     OSPF
  Type                         Any
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~1
{
  Protocol                     IGMP
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~2
{
  Protocol                     UDP
  SourcePortRange              520
  DestinationPortRange         1024 65535
  Direction                    BiDirectional
```

```
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~3
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~4
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

## NOTE -- Generated IpService IKE~Gen
IpService                       IKE~Gen
{
  Protocol                      UDP
  SourcePortRange               500
  DestinationPortRange          500
  Direction                     BiDirectional
  Routing                       Local
}

IpService                       All_other_traffic
{
  Protocol                      All
  Direction                     BiDirectional
  Routing                       Local
}

IpDynVpnAction                  IPSec__Silver
{
  Initiation                    Either
  VpnLife                       1440
  Pfs                           None
  IpDataOfferRef                IPSec__Silver~R
  IpDataOfferRef                IPSec__Silver~R~6
  IpDataOfferRef                IPSec__Silver~R~7
}
##
## Connectivity Rule Omproute combines the following items:
##   Local data endpoint       All4
##   Remote data endpoint      All4
##   Topology                  None (Permit/Deny only)
##   Requirement Map           Omproute
##     OMPROUTE-IP_V4            => Permit

IpFilterRule                    Omproute~1
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4
```

```
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~2
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~1
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~3
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~2
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~4
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~3
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   Omproute~5
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~4
  IpGenericFilterActionRef     Permit~LogNo
}
##
## Connectivity Rule All_Traffic_Silver combines the following items:
##   Local data endpoint       All_Traffic_Silver~ADR~1
##   Remote data endpoint      All_Traffic_Silver~ADR~2
##   Topology                  HH
##   Requirement Map           All_Traffic_Silver
##     All_other_traffic         => IPSec__Silver

IpAddr                         All_Traffic_Silver~ADR~1
{
  Addr                         10.10.1.230
}

IpAddr                         All_Traffic_Silver~ADR~2
{
  Addr                         10.10.1.241
}

LocalSecurityEndpoint          All_Traffic_Silver~LSE~4
{
  Identity                     IpAddr 10.10.1.230
  LocationRef                  All_Traffic_Silver~ADR~1
}

RemoteSecurityEndpoint         All_Traffic_Silver~RSE~3
{
```

```
  Identity                      IpAddr 10.10.1.241
  LocationRef                   All_Traffic_Silver~ADR~2
}

KeyExchangeRule                 All_Traffic_Silver~5
{
  LocalSecurityEndpointRef      All_Traffic_Silver~LSE~4
  RemoteSecurityEndpointRef     All_Traffic_Silver~RSE~3
  KeyExchangeActionRef          All_Traffic_Silver
  SharedKey                     Ebcdic abcde
}

KeyExchangeAction               All_Traffic_Silver
{
  HowToInitiate                 Main
  HowToRespond                  Either
  KeyExchangeOfferRef           KEO~5
  AllowNat                      No
}

## NOTE -- Generated IpFilterRule All_Traffic_Silver~6
IpFilterRule                    All_Traffic_Silver~6
{
  IpSourceAddrRef               All_Traffic_Silver~ADR~1
  IpDestAddrRef                 All_Traffic_Silver~ADR~2
  IpServiceRef                  IKE~Gen
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    All_Traffic_Silver~7
{
  IpSourceAddrRef               All_Traffic_Silver~ADR~1
  IpDestAddrRef                 All_Traffic_Silver~ADR~2
  IpServiceRef                  All_other_traffic
  IpGenericFilterActionRef      IpSec~LogYes
  IpDynVpnActionRef             IPSec__Silver
}

KeyExchangePolicy
{
  AllowNat                      No
  KeyExchangeRuleRef            All_Traffic_Silver~5
}

IpFilterPolicy
{
  PreDecap                      OFF
  FilterLogging                 ON
  IpFilterLogImplicit           Yes
  AllowOnDemand                 Yes
  IpFilterRuleRef               Omproute~1
  IpFilterRuleRef               Omproute~2
  IpFilterRuleRef               Omproute~3
  IpFilterRuleRef               Omproute~4
  IpFilterRuleRef               Omproute~5
  IpFilterRuleRef               All_Traffic_Silver~6
  IpFilterRuleRef               All_Traffic_Silver~7
}
```

Example: D-2 TCPIPA policy for SC31

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC31
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Fri Aug 25 13:13:41 EDT 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}

IpGenericFilterAction          IpSec~LogYes
{
  IpFilterAction               IpSec
  IpFilterLogging              Yes
}

KeyExchangeOffer               KEO~5
{
  HowToEncrypt                 DES
  HowToAuthMsgs                SHA1
  HowToAuthPeers               PresharedKey
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~6
{
  HowToEncap                   Transport
  HowToEncrypt                 3DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      120 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~7
```

```
{
  HowToEncap                    Transport
  HowToEncrypt                  AES
  HowToAuth                     ESP Hmac_Sha
  RefreshLifetimeProposed       240
  RefreshLifetimeAccepted       120 480
  RefreshLifesizeProposed       None
  RefreshLifesizeAccepted       None
}

IpService                       OMPROUTE-IP_V4
{
  Protocol                      OSPF
  Type                          Any
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~1
{
  Protocol                      IGMP
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~2
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~3
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~4
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

## NOTE -- Generated IpService IKE~Gen
IpService                       IKE~Gen
{
  Protocol                      UDP
  SourcePortRange               500
  DestinationPortRange          500
  Direction                     BiDirectional
  Routing                       Local
```

```
}

IpService                       All_other_traffic
{
  Protocol                      All
  Direction                     BiDirectional
  Routing                       Local
}

IpDynVpnAction                  IPSec__Silver
{
  Initiation                    Either
  VpnLife                       1440
  Pfs                           None
  IpDataOfferRef                IPSec__Silver~R
  IpDataOfferRef                IPSec__Silver~R~6
  IpDataOfferRef                IPSec__Silver~R~7
}
##
## Connectivity Rule Omproute combines the following items:
##   Local data endpoint        All4
##   Remote data endpoint       All4
##   Topology                   None (Permit/Deny only)
##   Requirement Map            Omproute
##     OMPROUTE-IP_V4             => Permit

IpFilterRule                    Omproute~1
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    Omproute~2
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4~1
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    Omproute~3
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4~2
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    Omproute~4
{
  IpSourceAddr                  All4
  IpDestAddr                    All4
  IpServiceRef                  OMPROUTE-IP_V4~3
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    Omproute~5
{
```

```
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4~4
  IpGenericFilterActionRef     Permit~LogNo
}
##
## Connectivity Rule All_Traffic_Silver combines the following items:
##   Local data endpoint       All_Traffic_Silver~ADR~1
##   Remote data endpoint      All_Traffic_Silver~ADR~2
##   Topology                  HH
##   Requirement Map           All_Traffic_Silver
##     All_other_traffic         => IPSec__Silver

IpAddr                         All_Traffic_Silver~ADR~1
{
  Addr                         10.10.1.241
}

IpAddr                         All_Traffic_Silver~ADR~2
{
  Addr                         10.10.1.230
}

LocalSecurityEndpoint          All_Traffic_Silver~LSE~4
{
  Identity                     IpAddr 10.10.1.241
  LocationRef                  All_Traffic_Silver~ADR~1
}

RemoteSecurityEndpoint         All_Traffic_Silver~RSE~3
{
  Identity                     IpAddr 10.10.1.230
  LocationRef                  All_Traffic_Silver~ADR~2
}

KeyExchangeRule                All_Traffic_Silver~5
{
  LocalSecurityEndpointRef     All_Traffic_Silver~LSE~4
  RemoteSecurityEndpointRef    All_Traffic_Silver~RSE~3
  KeyExchangeActionRef         All_Traffic_Silver
  SharedKey                    Ebcdic abcde
}

KeyExchangeAction              All_Traffic_Silver
{
  HowToInitiate                Main
  HowToRespond                 Either
  KeyExchangeOfferRef          KEO~5
  AllowNat                     No
}

## NOTE -- Generated IpFilterRule All_Traffic_Silver~6
IpFilterRule                   All_Traffic_Silver~6
{
  IpSourceAddrRef              All_Traffic_Silver~ADR~1
  IpDestAddrRef                All_Traffic_Silver~ADR~2
  IpServiceRef                 IKE~Gen
  IpGenericFilterActionRef     Permit~LogNo
}
```

```
IpFilterRule                    All_Traffic_Silver~7
{
  IpSourceAddrRef               All_Traffic_Silver~ADR~1
  IpDestAddrRef                 All_Traffic_Silver~ADR~2
  IpServiceRef                  All_other_traffic
  IpGenericFilterActionRef      IpSec~LogYes
  IpDynVpnActionRef             IPSec__Silver
}

KeyExchangePolicy
{
  AllowNat                      No
  KeyExchangeRuleRef            All_Traffic_Silver~5
}

IpFilterPolicy
{
  PreDecap                      OFF
  FilterLogging                 ON
  IpFilterLogImplicit           Yes
  AllowOnDemand                 Yes
  IpFilterRuleRef               Omproute~1
  IpFilterRuleRef               Omproute~2
  IpFilterRuleRef               Omproute~3
  IpFilterRuleRef               Omproute~4
  IpFilterRuleRef               Omproute~5
  IpFilterRuleRef               All_Traffic_Silver~6
  IpFilterRuleRef               All_Traffic_Silver~7
}
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 433. Note that some of the documents referenced here may be available in softcopy only.

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7340
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-based Network Security*, SG24-7342
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *IP Network Design Guide*, SG24-2580
- *Deploying a Public Key Infrastructure*, SG24-5512
- *OSA-Express Implementation Guide*, SG24-5948
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *z/OS Infoprint Server Implementation*, SG24- 6234
- *z/OS 1.6 Security Services Update*, SG24-6448
- *zSeries HiperSockets*, SG24-6816

Other publications

These publications are also relevant as further information sources:

- *z/OS V1R8.0 Migration*, GA22-7499
- *z/OS V1R8.0 UNIX System Services Planning*, GA22-7800
- *z/OS V1R8.0 Communications Server: New Function Summary*, GC31-8771
- *z/OS V1R8.0 Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS V1R8.0 Cryptographic Services ICSF Overview*, SA22-7519
- *z/OS V1R8.0 Cryptographic Services ICSF System Programmer's Guide*, SA22-7520
- *z/OS V1R8.0 Cryptographic Services ICSF Administrator’s Guide*, SA22-7521

© Copyright IBM Corp. 2007. All rights reserved.

- *z/OS V1R8.0 MVS Interactive Problem Control System (IPCS) Commands*, SA22-7594
- *z/OS V1R8.0 MVS System Commands*, SA22-7627
- *z/OS V1R8.0 Security Server RACF System Programmer's Guide*, SA22-7681
- *z/OS V1R8.0 Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS V1R8.0 Security Server RACF Command Language Reference*, SA22-7687
- *z/OS V1R8.0 TSO/E Command Reference*, SA22-7782
- *z/OS V1R8.0 UNIX System Services User's Guide*, SA22-7801
- *z/OS V1R8.0 UNIX System Services Command Reference*, SA22-7802
- *z/OS V1R8.0 UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS V1R8.0 UNIX System Services Programming Tools*, SA22-7805
- *z/OS V1R8.0 UNIX System Services Messages and Codes*, SA22-7807
- *z/OS V1R8.0 UNIX System Services File System Interface Reference*, SA22-7808
- *z/OS UNIX System Services Parallel Environment: Operation and Use*, SA22-7810
- *z/OS V1R8.0 XL C/C++ Run-Time Library Reference*, SA22-7821
- *System z9 and zSeries Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935
- *z/OS V1R8.0 Cryptographic Services System SSL Programming*, SC24-5901
- *z/OS V1R7.0 Integrated Security Services Network Authentication Service Administration*, SC24-5926
- *HTTP Server Planning, Installing and Using, V5.3*, SC31-8690
- *z/OS V1R8.0 Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS V1R8.0 Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS V1R8.0 Communications Server: SNA Operations*, SC31-8779
- *z/OS V1R8.0 Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS V1R8.0 Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS V1R8.0 Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS V1R8.0 Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS V1R8.0 Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS V1R8.0 Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *z/OS V1R8.0 Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *z/OS V1R8.0 Communications Server: IP Sockets Application Programming Interface Guide and Reference*, SC31-8788
- *z/OS V1R8.0 Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS V1R8.0 Communications Server: CSM Guide*, SC31-8808
- *z/OS V1R8.0 Communications Server: IPv6 Network and Application Design Guide*, SC31-8885
- *z/OS V1R8.0 Communications Server: Quick Reference*, SX75-0124
- *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*, by Electronic Frontier Foundation, John Gilmore (Editor), 1988

Online resources

These Web sites and URLs are also relevant as further information sources:

- Mainframe networking

  http://www.ibm.com/servers/eserver/zseries/networking/

- z/OS Communications Server product overview

  http://www.ibm.com/software/network/commserver/zos/

- z/OS Communications Server publications

  http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/r7pdf/commserv.html

- IBM Configuration Assistant for z/OS Communications Server

  http://www-1.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=D400&uid=swg24013160&loc=en_US&cs=UTF-8&lang=en

- z/OS downloads

  http://www.ibm.com/servers/eserver/zseries/zos/downloads/

- The default behavior of IPsec NAT traversal (NAT-T) is changed in Windows XP Service Pack 2

  http://support.microsoft.com/kb/885407/en-us

- TOOLS - IDSAUTO

  http://www.ibm.com/support/docview.wss?uid=swg24001743

- Manage Intrusion Detection Services with Tivoli Risk Manager

  http://www-1.ibm.com/support/docview.wss?uid=swg24006973

- *Cryptanalysis of MD5 Compress*, by Hans Dobbertin

  http://www.cs.ucsd.edu/users/bsy/dobbertin.ps

- IETF RFC Page

  http://www.ietf.org/rfc.html

- VeriSign

  http://www.verisign.com/repository/crptintr.html

How to get IBM Redbooks

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

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A
Abstract Syntax Notation (ASN) 359
access control 86, 314, 330, 354
access control list (ACL) 24, 310, 364
additional information 19, 24, 226, 275, 277–278, 286, 328
    on RACF 85
    on zSeries hardware cryptography 85
Advanced Encryption Standard (AES) 338
ALTUSER command 365
application programming interface (API) 356
application security 225, 351
Application Transparent Transport Layer Security (AT-TLS) 13, 22, 223
associating user ID with started task (STC) 310
attack categories
    ICMP redirect restriction 256
    inbound fragment restrictions 255
    IP option restriction 255
    IP protocol restrictions 255
    malformed packets 255
    outbound raw restrictions 256
    TCP SYNflood 256
    UDP perpetual echo 256
attack policy 255
    notification 256
    statistics 257
    tracing 257
attempt to FTP to system A23 from remote workstation (should fail) 74
AT-TLS
    application
        restriction 223
        types 225
    implementation 227
    importance 226
    operability verification 245
    policy 5, 225, 243, 330
Authentication Header (AH) 76
authentication server (AS) 358
authenticator 360
authorization code (AC) 309
Authorized Program Facility (APF) 309
authorized users access to start and stop PAGENT 12
authorizing use of hardware cryptographic encryption 84
available management tools 301

B
basic concept 3–4, 21–22, 75, 223–224, 251, 279, 305–306, 334, 369
basic configuration 14
basic services traffic 29
batch job 229, 373

C
CA certificate 243, 345, 354, 370, 410
    server certificate 397
Certificate Authority (CA) 82, 312, 341–342, 345, 354–355, 370
    direct request 144
certificate management 305, 370
    RACDCERT 371
    RACF common key ring 374
    SSL 402–403
certificate request 353, 371, 374
    appropriate field 402
Certificate Version Number 372
CIM provider access control 329
Cisco
    QoS Device Manager 302
    QoS MIB 302
    QoS Policy Manager 302
client authentication 225, 353, 370, 414
    appropriate parameters 388
client certificate 163, 225, 328, 353, 370, 380
    CA certificate 411
client policy 229
client policy configuration 240
client/server (CS) 362, 380
coding policy definition in configuration file 17
Commercial Data Masking Facility (CDMF) 338
Common Information Model (CIM) 329
common mistakes 278
Common Name (CN) 372
    required fields 373
Condition Set 266, 293
configuration client 372
configuration file 4, 25, 79, 86, 260, 282, 286, 375
    coding policy definitions 17
    dynamic monitoring 18
    IKE daemon obtains operational parameters 79
    modification time 18
Conjunctive Normal Form (CNF) 301
    policies 301
connect the certificates to IKED's key ring 83
connections 257
connectivity rule 36, 105
constrained state 258
controlling access during the window period 244
controlling program access by SYSID 310
country 372
creating
    certificate for the server 83
    IKE daemon configuration file 80
    IP security policy 169
    QoS policy rules 294
    RACF key ring 82
cryptographic key 77–78
    automatic management 78

© Copyright IBM Corp. 2007. All rights reserved.

currently unused (CU) field 282

D

Data Encryption Standard (DES) 338, 374
datagram 256
DCAS
  RACDCERT 372
DD SYSOUT 10, 79
default policy 8, 25
  group similar resources 26
  implicit rules 28
  major differences 26
default_realm statement 367
defining
  policies 5
  policy rules, considerations 17
DELUSER command 365
denial of service (DoS) 363
Differentiated Services 280
  DS field 282
  policies 283–284
  rule 285
  TOS octet 282
  traffic class octet 282
Differentiated Services Code Point (DSCP) 281–282
  setting using the Policy Agent 284
Diffie-Hellman 341
digital certificate 82, 224, 229, 312, 334, 343, 351, 369, 372
  Certificate Authority (CA) 344
  in RACF 312
  in z/OS 371
  management in OS/390 and z/OS 345, 371
  Public Key Infrastructure (PKI) 343
  security considerations 344
Digital Certificate Access Server (DCAS) 329, 352
  access control 329
digital certificates and key rings definition 243
digital signature 84, 334, 336, 348, 379
  tricky part 350
DIGTCERT and DIGTNMAP
  activate classes 82
disabling PAGENT policies 14
Distinguished Name
  only compulsory subfields 372
distinguished name (DN) 161, 343, 353, 370, 372
does not specifically (DNS) 367
download and installation 261, 287
DS field 282
dynamic tunnel 31, 77
  IPSec VPNs 31

E

elliptic curve 340
  cryptosystem 340
e-mail address 161
enabling CSFSERV resources 244
Encapsulating Security Payload (ESP) 76
encryption algorithms
  AES 338
  CDMF 338
  DES 338
  Diffie-Hellman 341
  elliptic curves 340
  IDEA 338
  performance issues 345
  RC2 338
  RC4 338
  triple DES 338
end user 316, 387
environment variable 10, 143
eServer IDS Configuration Manager 260
example setup 314
exit program 374
external CA-signed client certificate gskkyman procedure 413
EZB.NETACCESS 315
EZB.NETSTAT 325
EZB.PORT Access 306
  INSUFFICIENT ACCESS AUTHORITY 306
EZB.PORTACCESS 319
EZB.STACKACCESS 314

F

fair share algorithm 258
false positive scans 254
Fast Response Cache Accelerator (FRCA) 225, 330
  access control 330
fast scan 252
File Transfer Protocol (FTP) 375
  program 23, 313, 372
filter policy 23
Firewall Technologies configuration client, gskkyman 372
FTP and TN3270 filtering scenario 29
FTP client 329, 364, 382, 414
FTP server 50, 285, 316, 319, 352, 364, 382, 396
  access control 328
FTP SITE command control 328
FTP z/OS UNIX access control 328
further information 203, 312

G

generic profile 11, 306, 321, 324
  protecting all V TCPIP commands, defining 323
graphical user interface (GUI) 4, 259
GROUP TCPGRP 81, 320
gskkyman 354, 369, 371
  certificate request file 404
  exported certificate 391
  key pair file 404
  menu screen 403
  stash file 404
  strong crypto version 404
GSSAPI protocol 364

H

High Level Qualifier (HLQ) 308
HMAC 348
HTTP Server
  certificate management 402–403
  gskkyman 372
  RACDCERT 372

I

IBM Corporation 82, 243, 373
ICH408I User 317
IKE daemon (IKED) 77, 319
implement packet (IP) 21
implementation steps 30
implementing
  IPSec between two z/OS systems 88, 95
  IPSec between z/OS and Windows 143, 180
  PAGENT on z/OS 9
importing the z/OS certificates into Windows XP 163
inbound request 6
Information Technology (IT) 1
installing
  PAGENT 86, 260
  X509 digital certificate for IKED 82
Integrated Service 280
  narrow applicability 284
Integrated Services
  (RSVP) policies 284
  policies 283
International Data Encryption Algorithm (IDEA) 338
Internet Engineering Task Force (IETF) 352
Internet Key Exchange (IKE) 22, 76–77, 372
  RACDCERT 372
  SA 77
Intrusion Detection Services (IDS) 250
  attack categories 255
  attacks 255
  implementation 259
  policy 249
  Traffic Regulation (TR) policies 257
IP address 23, 88, 253, 316
  1.1.1.1 port 23 7
  10.40.1.230 29, 246
  10.40.1.241 29
  information 27, 163
IP Authentication Header (AH) protocol 77
IP checksum 206
IP datagram
  TOS octet 282
IP Encapsulating Security Payload (ESP) protocol 77
IP Filter
  list 171–172
IP filtering
  implementation 25
  importance 24
IP packet 22, 77, 282, 315
  IP datagram portion 77
IP precedence bits 282
IP Security 21
  configuration file 87
  event 85
  filter policy 23–24
  policy 29, 77, 79
  policy configuration file 28
  policy configuration statement 86, 260
  policy statement 28
IP traffic 26, 161
IP Version 6 (IPv6)
  traffic class octet 282
IPCONFIG 25
IPCONFIG6 25
IPSec
  implementation 78
IPSec command
  access control 328
ipsec command 22, 26, 86, 328
IPSec policies to PAGENT definition 87
IPSec policy
  file 28, 146
  statement 28, 86, 260, 286
IPSECURITY 25, 29
IPSECURITY IPCONFIG 7
IPSECURITY option on the IPCONFIG statement 24
IPv6 advanced socket API options 321
IPV6_NEXTHOP Chlorine 308
Issue pasearch (IP) 4, 280
ITSO Raleigh
  test environment 406
  TN3270 Server 381
  Webserver 402
  Webserver Cert 3 403

J

job name 228, 314

K

KERB segment 365
Kerberos 358
  assumptions 363
  authentication server (AS) 358
  authenticator 360
  denial-of-service (DoS) 363
  implementation in z/OS 364
  inter-realm operation 363
  Key Distribution Center (KDC) 358
  principal identifier 358
  realm 358
  ticket granting ticket (TGT) 358
  Version 5 358
    protocol 358
    server 364
Kerberos principal
  name 365
  name CS09 365
Kerberos server 358
  further information 368
KERBLINK class 365
  residual mapping profiles 365

key database 355, 370, 379
CA-signed server certificate 406
DEFAULT certificate 417
server certificate 407
Key Distribution Center (KDC) 351, 358, 364
statement 367
key IPSec components 76
key length 338, 347
key pair 374
key ring 80, 82, 229, 243, 312, 370, 375
key size 347, 356, 373, 383
kill command 14
kpasswd_server statement 367

L

LDAP server 4, 18, 250, 284, 372
policy information 266
policy objects 18
refreshing policies 18
Lightweight Directory Access Protocol (LDAP) 4, 282, 284
gskkyman 372
RACDCERT 372
Line Print daemon (LPD) 227
Link Pack Area (LPA) 309
List/Manage key 374
local realm
point 367
ZOS17.RAL.IBM.COM 364
locality 372
logging level definition 16
logging on to system A23 using TN3270 72
logging on to system A24 using TN3270 and using TSO FTP to system A23 74
logical partition (LPAR) 288
A23 29

M

MD2 347
MD5 347
message authentication 345
code 336, 346
process 336
message authentication code (MAC) 354
HMAC 348
message digest 336
algorithms, MD2 347
algorithms, MD5 347
algorithms, SHA-1 347
Microsoft Management Console (MMC) 162
MLS
Basic concepts 311
MODDVIPA utility program control 330
multiple realm 363
mutual authentication 362
MVS data 10, 28, 87, 353, 382
exported certificate 382
returned certificate 402
MVS.VARY.TCPI P 322
MVS.VARY.TCPIP.STRT STOP
Chlorine 325
profile 323
MVS.VARY.TCPIP.STRT Stop 322

N

name SAPSYS 306
TCP/IP ports 307
National Security Agency (NSA) 347
NETACCESS block 315
DEFAULT statement 318
network/mask entry 316
NETSTAT 325
generic profile definition 326
home 326
profile definition 327
security scenario 326
Netview and z/OS IDS 278
Network access control overview 315
Network Address Translation (NAT) 105
network administrator 251
network MIB variables and tools 302
network resource 23, 313
nonce 360
Number of available connections 257

O

OMVS segment 12, 81, 311, 365
onetstat 325
OPERCMDS 322
option number 374
Organization 372
organizational-unit 372
organizational-unit (OU) 372
organization-name 372

P

PAGENT 282
configuration
file 11, 14, 28, 260
policy 5, 25, 284
QoS policies 283
started task to RACF definition 11
Pascal API 225, 258
PASEARCH command 8, 74
passive attack 255
password-guessing attacks 364
PCOMM client 383
performance issue 279, 345
performance monitor 288, 290
per-hop behavior (PHB) 282
per-hop-behavior (PHB) 281
ping command 29
policy action 6, 250, 284–285
policy conditions 294
Policy Agent 1, 3, 22, 29, 77, 223, 250, 279, 282, 351, 372
basic operational characteristics 14

command security 327
following environment variables 11
gskkyman 372
IP security policy 78
log file 18
RACDCERT 372
started task name 127, 177
policy condition 285
policy decision point (PDP) 4
policy enforcement point (PEP) 4
policy model 6
policy rule 6, 27, 266, 284
and action statement 6
policy time period condition 285
policy_type 327
policy-based networking 1, 3, 260, 279
key part 279
PORT/PORTRANGE SAF keyword 319
PORTRANGE statement 318
principal identifier 358
private key 145, 336, 348, 354, 371, 375
problem determination 74, 247, 301
profile name 12, 314
last qualifier 322
profile to protect the V TCPIP, START command definition 324
profiles to control access to RACDCERT command definition 81
Program Access
to data sets 309
Program Access Control 309
facility 309
program protection by RACF resource class PROGRAM 309
protecting
FTP-related resources 328
miscellaneous resources 329
NETSTAT/onetstat at command level 326
NETSTAT/onetstat at command option level 326
network access 315
network management resources 329
network resources 308
programs 309
sensitive network commands 321
use of socket options 320
VARY TCPIP at command level 322
VARY TCPIP at command option level 322
your network ports 318
your TCP/IP stack 314
Public key
infrastructure 342
trustworthy distribution 350
public key 333, 341, 351, 379
cryptography standard 340
Public Key Data Set (PKDS) 374
Public Key Infrastructure (PKI) 342
digital certificate 343
public/private key encryption 339
pull-down menu 162

Q

QoS configuration using the zQoS Manager 286
QoS Device Manager (QDM) 302
QoS in z/OS Communication Server, configuring 284
Qos Policy Manager 302
QoS with z/OS Communications Server 282
QPM 302
Quality of Service (QOS) 279–280
condition set 294–295
importance 286
MIBs 302
policy 258, 280
application performance requirements 302
rules 293
tools 301
Quality of Service (QOS) actions 296

R

RACDCERT CERTAUTH
Export 398
GENCERT 373
LIST command 410
RACDCERT command 81, 345, 354, 371
RACDCERT command format 11, 85, 313, 369
RACDCERT Id 82, 243, 381
RACDCERT RACF command 371
RACF
certificate management 421
common key ring 374, 401
database 145, 306
CA certificate 376, 412
client certificate 387
digital certificate 312, 377
internal CA certificate 397
self-signed client certificate 387
self-signed server certificate 382
SERVAUTH class 307
delegation of cryptographic resources 329
error 317
key ring 82
multilevel security (MLS) for network resources 311
profile details 322, 325
profile name 322–323, 402
profile, MYSUBNET 315
RACDCERT 371
remote sharing facility (RRSF) 365
resource class 309
resource class, program 309
resource protection 311
user ID 365, 374
user ID and group with IKE daemon 81
user ID with the PAGENT started task 12
RACFDCERT Id 379
RC2 338
RC4 338
RDEFINE SERVAUTH
EZB.IPSE CCMD 86
realm 358
real-time SMF information service access control 330

recommendations 226
Redbooks Web site 433
  Contact us xv
refreshing policies 18
Request for Comments (RFC)
  RFC 1510 358
  RFC 2253 343
required field 373
requirements 261, 287
requirements and download instructions 261
reserving TCPIP ports for IKE daemon 81
Resource Access Control Facility (RACF) 303, 305
resource class
  OPERCMDS 12, 322
  realm 364
  SERVAUTH 306
restricting
  access to pasearch command to authorized users 12
  use of ipsec command 86
restrictions 227
RESTRICTLOWPORTS 319
Reusable Objects 261
Rexx socket application scenario 228
RFC
  2474 281–282
  2475 281
  2597 281
  2598 281
RFC 1631 206
RFC 2402 76
Rivest, Shamir, and Adleman (RSA) 338, 340
RSVPD 282

S

SAF check 318, 326
SAF profile
  MVS.VARY.TCPIP.DROP 326
  MYPC 315
  name 317
  world 315
scan event 254
  certain category 269
  current count total 254
  ICMP scan 254
  TCP port scan 254
  UDP port scan 254
scan global 269
scan policy 251
  parameters 253
secret key 333, 351
  encryption 337
  KAlice 360
  KBob 362
  system 351
Secure Hash Standard (SHS) 347
Secure Sockets Layer (SSL) 223, 351, 372
Security Access Facility (SAF) 303, 305
Security Association (SA) 76
security policy database (SPD) 22–23
security product authorization for TRMD definition 20
self-signed certificate 82, 345, 354, 357, 369–371, 380
self-signed client certificate 380
sensitivity level 254
SERVAUTH class 307, 313–314
  following RACF profile EZB.PAGENT.sysname.tcp-name.policy_type 327
  profile EZB.FTP.sysname.ftpdname.PORTxxxxx 328
  RACF profiles 328
  RACLIST in-storage profiles 308
  z/OS Communications Server profiles 308
SERVAUTH resource class 314
server certificate 243, 345, 354, 369
server considerations 316
server policy 229
  configuring 229
Service Level Agreement 301
  Performance Monitor (SLAPM) 302
session key
  KAlice 360
session key, KAlice 360
SETROPTS RACLIST 12, 81, 244, 307, 314
setting DSCP using the Policy Agent 284
setup
  certificates 144
  IKED 78, 143
    cataloged procedure 79
  policy 146
  policy using z/OS GUI 95, 126
  profile 245
  security for daemons in z/OS UNIX 311
  started task procedure 19
  syslogd to log IKED messages 85
  Traffic Regulation Manager daemon (TRMD) 19, 85
  TTLS Stack Initialization access control 13
  Windows XP 162
SHA-1 347
SIOCTTLSCTL IOCTL 225
slow scan 253
SNMP
  agent control 329
  SLA subagent 302
SO_BROADCAST Socket option access control 320
source IP address 27, 253
  only unique events 254
SSL
  Certificate Authority (CA) 370
  client authentication 353
  connection 355
  considerations 357
  exchange 382
  handshake
    operation 355
    processing 355
    protocol 353
  message authentication code (MAC) 354
  public-private key pair 403
  record protocol 354
  self-signed certificate 357, 404
  server authentication 407
  server certificate 407

symmetric encryption keys 354
SSL V2.0
Des 356
RC2 Export 356
RC4 Export 356
Triple DES US 356
SSL V3.0
DES SHA Export 356
NULL MD5 356
NULL SHA 356
RC2 MD5 Export 356
RC4 MD5 Export 356
RC4 MD5 US 356
RC4 SHA US 356
Triple DES SHA US 356
SSL-enabled applications 352
stack access overview 314
starting
IKED and verifying initialization 85
PAGENT as started task 9
PAGENT from UNIX 13
TRMD from z/OS UNIX 19
state-or-province 372
STDENV DD card 143
sticky bit in the z/OS UNIX environment 310
stopping PAGENT 14
subnet mask 254, 316
255.255.255.0 316
length 274
Sysplex Distributor 283
actual setup 115, 276, 298–299
policy 5, 283–284
system A23
IP filter rules 29
System Display and Search Facility (SDSF) 14
system SSL 223, 354, 371
APIs 356
call 248
Cryptographic Services Base element 356
DLLs 357
verifying 244

T

TCP checksum 206
TCP connection
activity 329
control block 6
information service 329
access control 329
TCP connection control block (TCB) 6
TCP layer 224
TCP port 318
990 227
scan 254
TR policies 257
TCP/IP 4, 22, 75, 254, 313, 357
packet trace service access control 330
profile statement 318
profile, data set 25
stack initialization access control 330
TCP-based application 225
TCPCONFIG 319
TcpImage statement 11, 227
Define 14
TCPIP 4, 79, 227, 306, 316, 323, 367
TCPIP command
option 322
security 322–323
TCPIP port 81, 306–307
telling IKED where to find key ring 83
ticket granting server (TGS) 359
ticket granting ticket (TGT) 358
time zone (TZ) 11
title 372
Tivoli Risk manager and z/OS IDS 278
TLS 357, 376
TLS exchange 414
FTP server 415
TN3270 375
gskkyman 372
RACDCERT 372
TN3270 client 357, 383, 385
Cert 396
Certificate 387
certificate PF 388
PCOMM key database 412
user 414–415
TN3270 server 23, 253, 321, 352, 372
total connections 257
TR TCP 257
policy information 257
TR UDP policies 258–259
information 258
LONG 259
SHORT 259
VERY_LONG 259
VERY_SHORT 259
traffic conditioner block (TCB) 281
Traffic Regulation (TR) policies 257
Transparent Transport Layer Security (TTLS) 13
Transport Layer Security (TLS) 223, 348, 351
proposed standard 357
Triple-DES 338, 374
TRMDSTAT 20
TSO NETSTAT
and UNIX onetstat command security 325
command 317
Home 326
HOME command 326–327

U

u/cs10 >
export RESOLVER_CONFIG 19
export TZ 19
UDP port 254
512 319
IDS policies 272
IDS TR policies 259
UDPQUEUELIMIT 258
updating TCP/IP stack to activate IPSec 86

user ID 20, 83, 240, 316, 354, 375
KERB segment 365
user ID for PAGENT started task, defining 12
user UTSM 306, 314
user1 414
User-to-Network Interface (UNI) 280
using
GUI 261, 287
NETSTAT for Network Access control 316
NETSTAT to display Port Access control 320
z/OS Network Security Configuration Assistant 87

V

VARY TCPIP command security scenario 323
verification 18, 72, 176
certificate creation 83
VeriSign 354
Virtual Private Networking (VPN) 76
VPN tunnel 95, 187, 327

W

Web site 87, 345, 405, 409
voice 280
Windows XP 161, 163
client 162
host 144
task bar 162
VPN tunnel 170
workstation 162
z/OS certificates 163
work with IDS objects/rules 264, 266
working example of Network Access control 317

X

XYZ Corporation 341–342
public key 342

Y

YNAMNBR 144, 373

Z

z/OS Communications Server
component 78
environment 282
IP 6, 21
PAGENT function 25
Policy Agent 1, 3
profile 308
security protection mechanism 250
SNMP SLA subagent 301
z/OS Communications Server SNMP SLA subagent 302
z/OS environment 4, 22, 284, 369
digital certificates 369
network interface 22
policy-based networking 4
traffic prioritization 4
z/OS host
information 288
IP address 161
z/OS image 22, 79, 314
inbound and outbound TCP/IP traffic 26
TCP/IP components 23
z/OS IP
filtering implementation 25
Security Configuration Assistant GUI 24
Security Service 162
z/OS Network Security Configuration Assistant 6, 87, 229
z/OS platform 306
inherent security 309
main strengths 309
z/OS security
access facility 303, 305
server 303, 305
z/OS system 6, 22, 89, 306–307, 314, 345, 353, 411
dynamic tunnel 95
name 307, 321
non-virtual interface 27
SC30 326
SMF system ID 310
telnet server 6
VPN traffic 95, 187
z/OS V1R8
system SSL FMIDs 356
z/OS V1R8.0 Communications Server 9, 76
z/OS VARY TCPIP command security 322
zIDS Manager
scan events 270
Work with IDS Objects/Rules
attacks 267
scan events
ICMP scans 270
TCP port scans 270
UDP port scans 270
Work with Reuseable Objects 267
zQoS Manager 287
Work with QoS Policy Rules 293
QoS Actions 296
QoS Condition Sets 295
Work with z/OS host 288
Performance Monitor 288, 290

Redbooks

Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-Based Network Security

Understand CS for z/OS TCP/IP security and policy capabilities

See CS for z/OS security and policy implementation examples

Protect your z/OS networking environment

This Communications Server (CS) for z/OS TCP/IP Implementation series provides easy-to-understand step-by-step how-to guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP.

With the advent of TCP/IP and the Internet, network security requirements have become more stringent and complex. Because many transactions come from untrusted networks such as the Internet, and from unknown users, careful attention must be given to host and user authentication, data privacy, data origin authentication, and data integrity. In addition, there are certain applications shipped with TCP/IP such as File Transfer Protocol (FTP) that, without proper configuration and access controls in place, could allow unauthorized users access to system resources and data.

This IBM Redbook explains how to set up security for your z/OS networking environment. For more specific information about CS for z/OS base functions, standard applications, and high availability, refer to the other volumes in the series. These are:

- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 2: Standard Applications*, SG24-7340
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341

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

SG24-7342-00 ISBN 0738489824