

Reduce Risk and Improve Security on IBM Mainframes: Volume 2 Mainframe Communication and Networking Security

Axel Buecker

Thomas Cosenza

Uma Kumaraguru

Christopher Meyer

Vinicius Oliveira

Vinodkumar Ramalingam

Jan Thielmann

Joe Welsh



 **Security**

z Systems



International Technical Support Organization

**Reduce Risk and Improve Security on IBM Mainframes:
Volume 2 Mainframe Communication and Networking
Security**

September 2015

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

First Edition (September 2015)

This edition applies to the IBM System z12 Enterprise Class server, the IBM System z12 Business Class server, and Version 2, Release 1 (2.1), of IBM z/OS operating system (product number 5694-A01). This edition also applies to the IBM z Systems platform.

© Copyright International Business Machines Corporation 2015. All rights reserved.

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

Contents

Notices	vii
Trademarks	viii
IBM Redbooks promotions	ix
Preface	xi
Authors	xi
Now you can become a published author, too	xiii
Comments welcome	xiii
Stay connected to IBM Redbooks	xiii
Chapter 1. Mainframe network concepts and functions	1
1.1 Introduction to mainframe networks	2
1.1.1 Technical overview	2
1.1.2 Communications Server features and benefits	6
1.1.3 Who supports the network	7
1.2 History of mainframe networks	8
1.3 Mainframe network architecture	10
1.4 Networking hardware	13
1.4.1 Network connections	14
1.5 Network protocols	15
1.5.1 TCP/IP	15
1.5.2 SMC-R	16
1.5.3 SNA	17
1.6 Additional network components	21
1.6.1 VTAM	21
1.6.2 TCP/IP stack and functions	23
1.6.3 Enterprise Extender	25
1.6.4 TN3270/E	27
1.6.5 Special features	27
1.7 Network tools and products	30
1.7.1 NetView Performance Monitor	30
1.7.2 OMEGAMON XE for Mainframe Networks	31
1.7.3 Session Manager for z/OS	31
1.7.4 Solve: Access Session Management	32
1.8 Operations and administration	32
1.8.1 Operational tasks	32
1.8.2 z/OS network administrator tasks	33
1.9 Securing mainframe networks	34
Chapter 2. Cryptography for network security	37
2.1 Security concepts and architecture for network cryptography on System z	38
2.1.1 Basics of cryptography for network security	38
2.1.2 Definition of a secure communication model for networks	39
2.1.3 Applications of cryptosystems for network security	40
2.1.4 Overview of the z/OS TCP/IP cryptographic infrastructure	44
2.1.5 Transport Layer Security on z/OS	46
2.1.6 AT-TLS	51
2.1.7 IPSec	54

2.1.8	OpenSSH on z/OS	60
2.1.9	PKI services	65
2.2	Guiding principles for cryptography for network security	68
2.2.1	Choosing appropriate cryptographic algorithms for network security	69
2.2.2	Defining a cryptography strategy within your organization	73
2.2.3	Choosing Transport Layer Security implementations	76
2.2.4	Things to keep in mind when defining certificates	79
2.2.5	Guiding principles for IPSec	84
2.2.6	OpenSSH on z/OS UNIX, z/OS dependant features implementation	86
Chapter 3.	TCP/IP security	89
3.1	Introduction	90
3.1.1	IP network design	90
3.1.2	System z in a DMZ	90
3.1.3	Mixing environments	90
3.1.4	HiperSockets	91
3.2	Sockets and APIs	91
3.3	Telnet Server	93
3.3.1	Security concepts and architecture	94
3.4	FTP	98
3.4.1	Security concepts and architecture	99
3.5	InetD, the Internet daemon	102
3.5.1	Security concepts and architecture	103
3.6	Virtual IP addressing	105
3.6.1	Security concepts and architecture	105
3.7	z/OS IP filtering	107
3.7.1	Security concepts and architecture	110
3.8	IPSec	111
3.8.1	Security concepts and architecture	112
3.9	z/OS Intrusion Detection Services	116
3.9.1	Security concepts and architecture	117
3.10	IP resource security	125
3.10.1	SAF controls	125
3.10.2	Multi-level security	127
3.10.3	OSA-Express connection isolation	128
3.10.4	IP Profile Controls	128
Chapter 4.	SNA security	131
4.1	Introduction	132
4.2	SNA encryption versus IP encryption	132
4.3	Security controls using VTAM start options	133
4.3.1	Crypto-based start options	133
4.3.2	Access control start options	133
4.4	Transport security	135
4.4.1	Enterprise Extender	135
4.4.2	UDP/IP considerations	137
4.4.3	Network Address Translation considerations	137
4.4.4	Enterprise Extender IP security	137
4.5	TN3270 Security	138
4.5.1	Background	138
4.5.2	Securing TN3270 IP flow	138
4.5.3	SSL/TLS support	139
4.5.4	TN3270 SSL support	139

4.5.5	Securing DLSw connections	141
4.6	Searching security	141
4.6.1	Basics of searching	141
4.6.2	Subarea searches	142
4.6.3	Searching an APPN network	142
4.6.4	Controlling searches of other APPN networks	143
4.6.5	ADJCLUST tables	144
4.6.6	Controlling searches entering a network	146
4.6.7	Session Management Exit	147
4.6.8	Directory Services Management Exit	147
4.6.9	Searches that are not network-qualified	148
4.6.10	Authorized Cross-Net searches	148
4.7	Application security	149
4.7.1	Session-level encryption for data confidentiality	150
4.7.2	Message authentication for data integrity	152
4.7.3	LU 6.2 session-level authentication	152
4.7.4	LU 6.2 conversation-level authentication	153
4.8	Recap of recommendations	154
Chapter 5.	Shared Memory Communications over RDMA	155
5.1	Overview	156
5.1.1	SMC-R: A hybrid protocol	156
5.1.2	SMC-R eligibility	157
5.1.3	Enabling SMC-R and connection setup	158
5.2	Security characteristics of SMC-R connections	159
5.2.1	Protecting application data	159
5.2.2	Protecting network protocol headers	159
5.2.3	Firewalls and Deep Packet Inspection (DPI) devices	161
5.3	z/OS network security features and SMC-R	162
5.3.1	Interface-based SMC-R enablement	162
5.3.2	Port-based SMC-R exclusion	162
5.3.3	SAF-based network access controls	162
5.3.4	IP filter rules	163
5.3.5	IPSec	163
5.3.6	SSL/TLS, including Application Transparent TLS (AT-TLS)	163
5.3.7	SSH	164
5.3.8	Application layer security protocols and features	164
5.3.9	Integrated Intrusion Detection Services (IDS)	164
5.3.10	Multilevel Security (MLS)	165
Related publications	167
IBM Redbooks	167
Other publications	167
Help from IBM	167

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 grant 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	IBM z Systems™	System z®
AnyNet®	IMS™	Tivoli®
CICS®	Language Environment®	VTAM®
DataPower®	MVS™	WebSphere®
DB2®	NetView®	z Systems™
developerWorks®	OMEGAMON®	z/OS®
FICON®	Parallel Sysplex®	z/VM®
First Failure Support Technology™	QRadar®	z/VSE®
Global Technology Services®	RACF®	z9®
HiperSockets™	Rational®	zEnterprise®
IBM®	Redbooks®	zSecure™
IBM z™	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Inc., and Inc. device are trademarks or registered trademarks of Kenexa, an IBM Company.

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

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Find and read thousands of IBM Redbooks publications

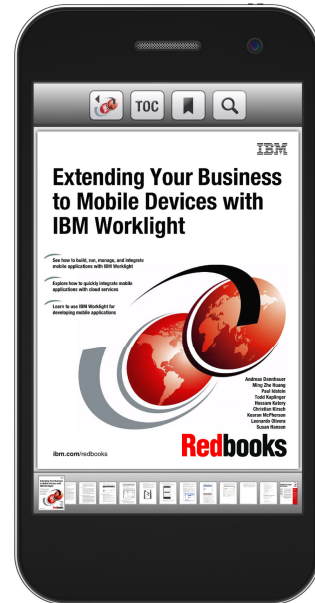
- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Download
Now

iOS



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication documents the strength and value of the IBM security strategy with IBM z™ Systems hardware and software (referred to in this book by the previous product name, IBM System z®). In an age of increasing security consciousness and more dangerous and advanced persistent threats, System z provides the capabilities to address today's business security challenges. This book explores how System z hardware is designed to provide integrity, process isolation, and cryptographic capability to help address security requirements.

We highlight the features of IBM z/OS® and other operating systems that offer a variety of customizable security elements. We also describe z/OS and other operating systems and additional software that use the building blocks of System z hardware to meet business security needs. We explore these from the perspective of an enterprise security architect and how a modern mainframe must fit into an enterprise security architecture.

This book is part of a three-volume series that focuses on guiding principles for optimized mainframe security configuration within a holistic enterprise security architecture. The intended audience includes enterprise security architects, planners, and managers who are interested in exploring how the security design and features of the System z platform, the z/OS operating system, and associated software address current issues, such as data encryption, authentication, authorization, network security, auditing, ease of security administration, and monitoring.

Authors

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

Axel Buecker is a Certified Consulting Software IT Specialist in the IBM Security business unit. He writes extensively and teaches IBM classes worldwide on areas of software security architecture and network computing technologies. He holds a master's degree in Computer Science from the University of Bremen, Germany and has 29 years of experience in various areas that are related to IT security architecture, workstation and systems management, network computing, and business solutions. Before he joined the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in software security architecture.

Thomas Cosenza is a Senior Certified IT Security Consultant and Specialist who joined IBM in 1998 after receiving his Computer Engineering degree from the University of Florida. Since 2004, Thomas has been working within the IBM Lab Services group creating IT security solutions on IBM System z for governments and corporations around the world. Among these solutions are IBM RACF®, IPsec VPN, Transport Layer Security (TLS), and IBM DataPower® XI50z. He is a regular speaker at the System z University, Share, and other vendor conferences, where he presents security solutions that customers can deploy within their enterprises. Thomas has also been certified by ISC2 as an Information Systems Security Professional (CISSP) since 2006 and has been a member in good standing since then.

Uma Kumaraguru is a Host Networking Specialist in the IBM Global Technology Services® Delivery organization. She is the Technical Lead for the team in India that provides Infrastructure Services support for Communications Server on z/OS. Uma has been with IBM since 2006 and has more than 11 years of experience in the IT industry. She specializes in z/OS Communications Server components, such as TCP/IP, SNA, and IBM VTAM®, and its related ISV and OEM products. She is based in Chennai, in South India, and has a bachelor's degree in Computer Science Engineering.

Christopher Meyer, CISSP, is a Senior Software Engineer in the IBM Software Group, on the IBM z Systems™ team, where he focuses on communications security. He has more than 30 years of experience in developing IBM operating systems and security-related software products.

Vinicius Oliveira is a Level 1 IBM IT Specialist located in Belo Horizonte, Brazil. He has 10 years experience in IT solutions. He joined IBM in 2009 as an IT Specialist for IBM Global Account, Canada, supporting internal IBM accounts as a member of the technical leadership team, with focus on infrastructure. His previous work experience includes network administration and analysis. Vinicius now works as a team leader for the IBM Rational® Support Team in Brazil. He specializes in infrastructure, middleware, and server support.

Vinodkumar Ramalingam is a Certified Senior IT Specialist for IBM System z. He works on architecture and technical solution development for the Mainframe Speciality Services Area in Global Technology Services, on the Delivery Technology and Engineering team. He has over 14 years of experience on System z, and has had various roles in IBM since 2004. He has a Master's in Philosophy degree in Computer Science, a master's in Information Technology, an MBA, and a post-graduate diploma in Cyber Law.

Jan Thielmann is a Certified IT Specialist for IBM z/OS in the IBM Software Group, in Germany. Jan joined IBM in 2006 and started working with System z and z/OS in 2008. Since 2009, he has been a member of a cross-brand System z Software Services Team. One of his main focus areas is Security on z/OS, mainly RACF with IBM Security zSecure™, but also other products in that area. He delivers service projects for clients accross Europe and presents regularly at IBM System z conferences. He is 11 years younger than RACF and holds a Bachelor of Science degree in Applied Computer Science.

Joe Welsh is a Senior Management IT Consultant and certified IT Network Specialist who joined IBM in 1988. He has held the roles of developer, designer, and tester for IBM z/OS Communications Server (VTAM, TCP/IP). Since 1998, he has performed IT consulting services engagements focused on SNA, Advanced Peer-to-Peer Networking (APPN) and high-performance routing (HPR), Enterprise Extender, VTAM, TCP/IP, and IP security for the Communications Server for (IBM AIX®, Linux, Microsoft Windows, and z/OS) at Fortune 500 companies around the world. This includes providing SNA, APPN and HPR, TCP/IP, and IP Security education and training, developing network designs and migrations, strategy and product direction, problem determination, implementation, and installation and migration assistance. Joe also provides network design, migration, and implementation services for 3745 NCP conversions to Communication Controller for Linux on zSeries for IBM customers worldwide. In 2012, he began assisting customers with IBM zEnterprise® BladeCenter Extensions, and Multi-site Workload Lifeline implementations.

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

Julie Bergh, Judith Broadhurst, Mike Fox, Gus Kassimis, Linwood Overby, Jerry Stevens
IBM

Now you can become a published author, too

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

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

- ▶ Use the online **Contact us** review Redbooks form:

ibm.com/redbooks

- ▶ Send your comments by email:

redbooks@us.ibm.com

- ▶ Mail your comments:

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

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

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

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

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

- ▶ Stay current on recent Redbooks publications with RSS feeds:

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



Mainframe network concepts and functions

In this chapter, we introduce mainframe network concepts and the latest networking technologies. We start with an overview of the network architecture and the various communication protocols, and then introduce the IBM z/OS Communication Server and its components. We also describe how the networking technology has evolved over the past years.

We take a look at the IBM Systems Network Architecture (SNA), including basic and Advanced Peer-to-Peer Networking (APPN). We then explain the TCP/IP implementation on z/OS and examine the SNA/IP implementation with a focus on the Enterprise Extender technology. We explain the TN3270 Enhanced protocol, its evolution, and provide a brief overview on the implementation. We also describe RDMA connectivity, which is a method of network connectivity added in version 2.1. Next, we cover the hardware connectivity on the mainframe and focus on the different types of connectivity. We include network software functions, such as FTP, SFTP, and SMTP. We conclude the chapter with information about the network management and network monitoring products that are available, with short descriptions of their functions.

Note: IBM z Systems platforms were previously called IBM System z platforms. Therefore, any text or labels in figures that say "System z" are valid for z Systems.

This chapter includes the following sections:

- ▶ 1.1, "Introduction to mainframe networks" on page 2
- ▶ 1.2, "History of mainframe networks" on page 8
- ▶ 1.3, "Mainframe network architecture" on page 10
- ▶ 1.4, "Networking hardware" on page 13
- ▶ 1.5, "Network protocols" on page 15
- ▶ 1.6, "Additional network components" on page 21
- ▶ 1.7, "Network tools and products" on page 30
- ▶ 1.8, "Operations and administration" on page 32
- ▶ 1.9, "Securing mainframe networks" on page 34

1.1 Introduction to mainframe networks

The z/OS network capability includes a full-featured Communications Server with integration of SNA and TCP/IP and RDMA protocols. This enables the mainframe to serve a large number of worldwide clients and applications simultaneously. The z/OS Communications Server provides a set of communications protocols, SNA, TCP/IP and RDMA, that allow these clients and applications to send and receive data using both local and wide-area networks (WANs).

This section covers the following aspects:

- ▶ Technical overview
- ▶ Communications Server features and benefits
- ▶ Who supports the network

1.1.1 Technical overview

The z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide-area networks, including the most popular wide-area network, the Internet. z/OS Communications Server is the IBM implementation of SNA and standard TCP/IP protocol suites on the IBM System z platform and also includes support for RDMA.

SNA is an IBM proprietary networking protocol, whereas TCP/IP is a component product of the z/OS Communications Server that provides a multitude of technologies that collectively provide an Open Systems environment for the development, establishment, and maintenance of applications and systems. RDMA is a high-speed, low latency, low CPU method of connecting servers that share a common LAN segment. RDMA is supported on z/OS using the Shared Memory Communications over RDMA (SMC-R) protocol that was first implemented in z/OS version 2.1 (V2R1).

z/OS Communications Server is a base element, meaning that it is part of the z/OS operating system. It provides both SNA and TCP/IP networking protocols for z/OS. The SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing (HPR) protocols. SMC-R is provided in the TCP/IP stack.

As Figure 1-1 on page 3 show, z/OS Communications Server includes three major components:

- ▶ The TCP/IP and RDMA (SMC-R) protocol stack.
- ▶ The SNA protocol stack contained in Virtual Telecommunications Access Method (VTAM).
- ▶ The Communications Storage Manager (CSM), which provides a shared I/O buffer area for both TCP/IP and VTAM data flow. The CSM function allows authorized host applications to share data without having to physically move the data.

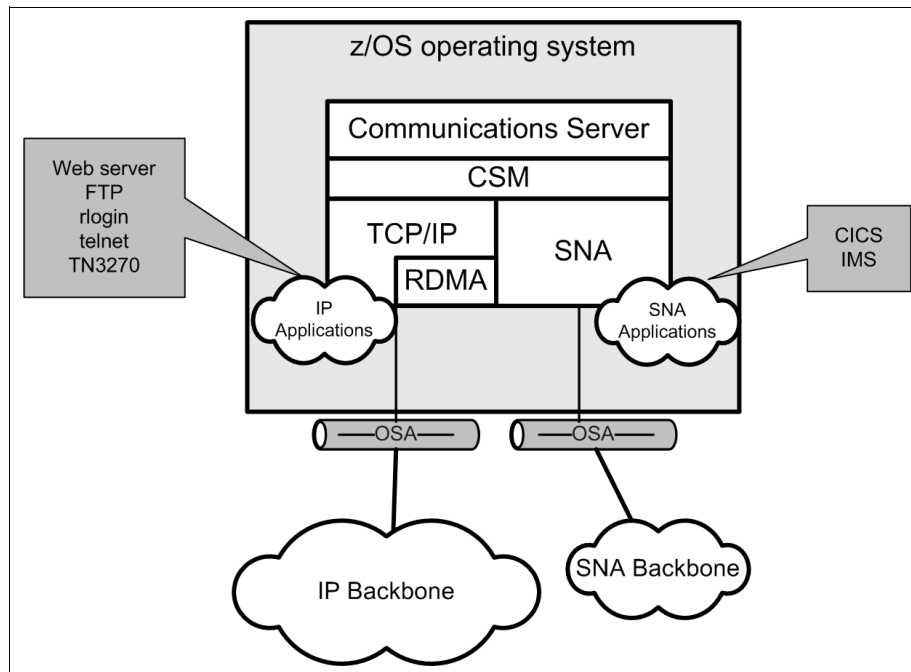


Figure 1-1 z/OS Communications Server

TCP, UDP, RAW protocols

The transport layer provides the support for the TCP, UDP, and RAW protocols. All three protocols use IPv4 or IPv6 as the network layer. The TCP protocol provides a connection-oriented, reliable transport layer, whereas UDP provides a simpler, connectionless and unreliable transport layer. The RAW transport layer provides for a more direct interface to the IP layer, which is primarily used by system management type applications.

TCP/IP protocol

TCP/IP is the set of communications protocols used both for the Internet and other similar networks. It is named for two of the most important protocols, which were the first two networking protocols defined in this standard:

- ▶ Transmission Control Protocol (TCP)

TCP is a transport layer protocol used by applications that require guaranteed delivery. TCP establishes a full duplex virtual connection between two endpoints. Each endpoint is defined by an IP address and a TCP port number. A byte stream is transferred in segments.

- ▶ Internet Protocol (IP)

IP routing is a set of protocols that determine the path that data follows to travel across multiple networks from its source to its destination. Data is routed from its source to its destination through a series of routers, and across multiple networks.

UDP protocol

As defined in RFC 768 the User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction-oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP).

ICMP protocol

As defined in RFC 792, the Internet Protocol (IP) is used for host-to-host datagram service in a system of interconnected networks. The network-connecting devices are called *gateways*. These gateways communicate for control purposes via a Gateway-to- Gateway Protocol (GGP). Occasionally, a gateway or destination host communicates with a source host, for example, to report an error in datagram processing. For such purposes, the Internet Control Message Protocol (ICMP) protocol is used. ICMP uses the basic support of IP as though it is a higher-level protocol; however, ICMP is actually an integral part of IP and must be implemented by every IP module.

ICMP messages can be sent in several situations:

- ▶ When a datagram cannot reach its destination
- ▶ When the gateway does not have the buffering capacity to forward a datagram
- ▶ When the gateway can direct the host to send traffic on a shorter route

IP is not designed to be absolutely reliable. The purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. There are still no guarantees that a datagram will be delivered or a control message will be returned. Some datagrams might still be undelivered without any report of their loss. The higher-level protocols that use IP must implement their own reliability procedures if reliable communication is required.

SMC-R protocol

The SMC-R protocol is a low latency, low CPU, high-speed method of connecting servers across a common network. SMC-R is implemented within the TCP/IP stack so that it can be transparently used by TCP/IP sockets applications, however it bypasses most of the TCP and IP processing to move data directly between hosts using Remote Direct Memory Access (RDMA).

SMC-R uses the RDMA over Converged Ethernet (RoCE) standard and requires the IBM 10GbE RoCE Express feature to perform RDMA operations over your existing Ethernet fabric. The SMC-R protocol stack is under the sockets layer alongside the TCP/IP stack to facilitate transparent exploitation by TCP sockets-based applications.

SMC-R operates by registering memory regions called Remote Memory Buffer Elements (RMBEs) in each partner host with the RoCE Express feature. Data is transferred by moving it into an RMBE on the sending side and then signaling the RoCE Express feature to communicate with the peer RoCE Express feature to move the data into the receiver's RMBE. In this manner, the data transfer is performed by the RoCE Express features without using the general-purpose CPU for networking protocol management.

SMC-R is a hybrid protocol. Peers first create a TCP connection by using existing logic, and then they negotiate use of SMC-R over that connection. If both peers support SMC-R and the configuration is suitable, the connection data is moved out of band by the RoCE Express features by using RDMA while the TCP connection remains up and idle. The TCP connection controls the SMC-R connection. For example, if the TCP connection is reset, the corresponding SMC-R connection is terminated.

SNA protocols and VTAM

Systems Network Architecture (SNA) is a data communication architecture established by IBM to specify common conventions for communication among the wide array of IBM hardware and software data communication products and other platforms. Virtual Telecommunications Access Method (VTAM) controls the network and maintains a table of all the machines and phone links in the network. It selects the routes and the alternate paths that messages can take between different NCP nodes. A subarea is the collection of terminals, workstations, and phone lines managed by an NCP. Generally, the NCP is responsible for managing ordinary traffic flow within the subarea, and VTAM manages the connections and links between subareas. Any subarea network must have a mainframe.

Communications Storage Manager

The Communications Storage Manager (CSM) is a VTAM component that allows authorized host applications to share data with VTAM and other CSM users without the need to physically copy the data. CSM includes an API that provides a way to obtain and return CSM buffers, change ownership of buffers, copy buffers, and manage CSM buffers.

High-Performance Routing

The HPR protocol is based on two key components: Rapid-Transport Protocol (RTP) and Automatic Network Routing (ANR). RTP is a reliable, connection-oriented protocol that ensures delivery and manages end-to-end network error and flow control. RTP creates new routes following a network failure. ANR is a connectionless service that is responsible for node-to-node source-routed service.

The RTP layer is invoked only at the edges of an APPN network. In intermediate nodes, only the ANR layer is invoked. RTP nodes establish RTP connections to carry session data. All traffic for a single session flows over the same RTP-to-RTP connection and is multiplexed with traffic from other sessions using the same connection.

For a more detailed overview, see the “High Performance Routing” section on the IBM Systems Network Architecture Routing page on Cisco.com:

<http://bit.ly/1QAGHqb>

Peer-to-peer networking

Earlier releases of VTAM support type 2.1 casual connections, a way of connecting two VTAMs in a peer-to-peer relationship using low-entry networking (LEN). Although type 2.1 casual connection is still supported, IBM recommends that you use APPN to connect two VTAMs in a peer-to-peer relationship.

z/OS UNIX System Services

z/OS UNIX System Services is the z/OS Communications Server implementation of UNIX as defined by X/Open in XPG 4.2. z/OS UNIX System Services coexists with traditional IBM MVS™ functions and traditional MVS file types (partitioned data sets, sequential files, and so on). It concurrently allows access to z/OS UNIX file system files and to z/OS UNIX utilities and commands by means of application programming interfaces and the interactive shell environment. z/OS Communications Server components run on UNIX System Services as processes that use other system services through z/OS.

Enterprise Extender

The Enterprise Extender (EE) is used in conjunction with the APPN extended border node function can replace SNA Network Interconnection (SNI) functions in a way that does not require SNA mature hardware, such as the IBM 3745 communication controller. High performance routing is an addition to APPN that improves reliability, increases network performance, and was designed to use higher link speed technologies.

1.1.2 Communications Server features and benefits

z/OS Communications Server provides application programming interfaces (APIs) and networking protocol support to enable SNA and TCP/IP applications running on z/OS to communicate with partner applications or users on the same system, other systems within a single data center, or in remote locations.

High-speed connectivity

Communications Server provides support for high bandwidth and high-speed networking technologies through the following features:

- ▶ Gigabit Ethernet with OSA Express QDIO for TCP/IP
- ▶ High-speed communication between TCP/IP stacks running in a z/OS logical partition (LPAR) using IBM HiperSockets™
- ▶ IBM FICON® connectivity, a third high-speed connectivity option
- ▶ The Shared Memory Communications over RDMA (SMC-R) protocol that uses the RDMA (Remote Direct Memory Access) over Converged Ethernet (RoCE) standard, which is implemented by the RoCE Express feature

High availability

Communications Server provides high availability to z/OS applications over both SNA and TCP/IP networks. IBM Parallel Sysplex® technology is used to enable high availability application support through the following functions:

- ▶ APPN High Performance Routing (HPR) provides end-to-end session continuity by supporting nondisruptive path switch around failed network components.
- ▶ The SNA Generic Resources function provides workload balancing for multi-instance applications within a parallel sysplex to maximize availability and efficiency of application data sharing.
- ▶ Multi-node Persistent Sessions provides session recovery for SNA applications.
- ▶ Dynamic VIPA (Virtual IP address) provides TCP/IP application availability across z/OS systems in a sysplex and allows participating TCP/IP stacks on z/OS to provide backup and recovery for each other for both planned and unplanned TCP/IP outages.
- ▶ Sysplex Distributor provides intelligent load balancing for TCP/IP application servers in a Sysplex and, along with Dynamic VIPA, provides a single system image for client applications connecting to these servers.

Enterprise connectivity

Communications Server has many features that support enterprise connectivity, including:

- ▶ TN3270 Server provides workstation connectivity over TCP/IP networks to access z/OS and enterprise SNA applications.
- ▶ Enterprise Extender allows SNA enterprise applications to communicate reliably over an IP network using SNA HPR and UDP transport layer protocols.

- ▶ APPN Extended Border Node (EBN) enables two or more SNA NetIDs to communicate via APPN protocols using VTAM. It can be used as a replacement for 3745/NCP-based SNA Network Interconnect (SNI).

System and data security

No network infrastructure these days can operate without proper security measures in place. Network security has to protect sensitive data and the operation of the TCP/IP stack on z/OS, as follows:

- ▶ z/OS provides robust TLS/SSL implementations and z/OS Communications Server provides a policy-based mechanism to apply that protection to z/OS applications transparently. Many key middleware products like IBM DB2®, IBM IMS™ Connect, IBM CICS®, IBM MQ, and IBM WebSphere® Application Server support TLS/SSL through one or more of these mechanisms as do numerous other z/OS components, such as these:
 - FTP client and server
 - RACF
 - JES
 - CSSMTP
 - TN3270 server
- ▶ Kerberos 5 and GSSAPI support is provided for the following applications:
 - FTP client and server
 - UNIX rshd server
 - UNIX System Services telnet server (supports only Kerberos 5)
- ▶ IPSec VPN functions enable the secure transfer of data over an IP network using standards for encryption, authentication, and data integrity.
- ▶ Intrusion Detection Services (IDS) provide mechanisms that detect a wide variety of potential attacks. The specific events to detect attacks and the actions to take (for example, logging, dropping packets, terminating connections, and such) are defined in IDS policy.
- ▶ z/OS Communications Server also provides SAF-based access controls to protect a wide variety of networking-related resources.

Network management

Network management support collects network topology, status, and performance information, and makes it available to network management tools.

- ▶ VTAM supports the traditional Common Management Information Protocol (CMIP) network management protocol for managing SNA network topology and providing other management interfaces for performance monitoring.
- ▶ Managing TCP/IP, the SNMPv3 protocol is supported.
- ▶ Communications Server provides support for standard SNMP applications and IETF standard TCP/IP protocol management information bases (MIB).
- ▶ Additional MIB support is provided by the IBM MVS TCP/IP enterprise-specific MIB, which supports management data for Communications Server TCP/IP stack-specific functions.

1.1.3 Who supports the network

Network communication functions have both a software and a hardware aspect, and a separation of software and hardware administrative duties is common in large organizations. However, the network administrator, who is a skilled software data communication expert, needs to understand both aspects.

The network administrator must bring a thorough understanding of the operating system's communications interfaces to any project that involves working with the organization's network. Although network hardware technicians have specific skills and tools for supporting the physical network, their expertise often does not extend to the operating system's communications interfaces. For example, when a nationwide retail chain opens a new store, the network administrators and network hardware technicians must coordinate their efforts to open the new store.

The following tasks are among the responsibilities of a z/OS network administrator:

- ▶ Defining, maintaining, and modifying an existing mainframe network
- ▶ Providing technical guidance to application development and business unit projects
- ▶ Determining, isolating, and correcting problems
- ▶ Tuning performance
- ▶ Making planning recommendations for capacity
- ▶ Developing operational procedures
- ▶ Training network operators
- ▶ Maintaining an awareness of emerging network technologies
- ▶ Recommending and implementing new network technologies

1.2 History of mainframe networks

Established in 1969, the Transmission Control Protocol/Internet Protocol (TCP/IP) is actually five years older than the System Network Architecture (SNA). However, SNA was immediately made available to the public, but TCP/IP was limited at first to military and research institutions, for use in the interconnected networks that formed the precursors to the Internet.

In 1974, IBM introduced its Systems Network Architecture (SNA), which is a set of protocols and services enabling communication between host computers (IBM mainframes) and peripheral nodes, such as IBM dedicated hardware offerings, like the IBM 3174 controller for IBM 3270 type displays and printers, controllers for the retail and finance industry, and more. The mainframe subsystem that implements SNA was named Virtual Telecommunication Access Method (VTAM). The robustness of the SNA protocol, the IBM hardware, and the transaction management infrastructure software supplied by IBM (CICS and IMS) made SNA the dominant protocol in the Fortune 1000 companies. In addition, SNA was designed to include network management controls not originally in TCP/IP through the Synchronous Data Link Control (SDLC) protocol.

In the 1980s, SNA was widely implemented by large corporations because it allowed their IT organizations to extend central computing capability worldwide with reasonable response times and reliability. For example, widespread use of SNA allowed the retail industry to offer new company credit card accounts to customers at the point of sale.

In 1983, TCP/IP entered the public domain in Berkeley BSD UNIX. TCP/IP maturity, applications, and acceptance advanced through an open standards committee, the Internet Engineering Task Force (IETF), using the Request For Comment (RFC) mechanism.

Note: The term *internet* is used as a generic term for a TCP/IP network and should not be confused with the *Internet*, which consists of the large international backbone networks connecting all TCP/IP hosts that have links to the Internet backbone.

TCP/IP was designed for interconnected networks (an *internet*) and seemed to be easier to set up, but SNA design was hierarchical with the centralized mainframe being at the top of the hierarchy. The SNA design included network management, data flow control, and the ability to assign *class of service* priorities to specific data workloads. Communication between autonomous SNA networks became available in 1983. Before that, SNA networks could not talk to each other easily. The ability of independent SNA networks to share business application and network resources is called SNA Network Interconnect (SNI). During the 20-year period when SNA was the primary networking method, many CICS and IMS application programs were developed and put in place. The application programming interfaces (API) of these application programs are heavily dependent on the underlying protocol, SNA.

It is apparent that TCP/IP is the dominant networking protocol now and for the foreseeable future. Today, new applications use state-of-the-art programming techniques, like Java and HTTP, but it will take many years until all SNA applications disappear. Why is that so?

A networking application is dependent on the communication protocol it uses. Every protocol provides an application programming interface (API). TCP/IP's API is called *socket programming* and SNA has its own API. Migrating a networking application from one protocol to another (that is, from SNA to TCP/IP) requires replacing the calls to the API. Business managers are reluctant to invest in protocol conversion only for the sake of changing the underlying protocol without introducing new functions and improvements. More importantly, organizations have invested a tremendous amount of labor and money in developing SNA applications. Considering the investments in SNA applications, these programs will be used for many years to come. To recode these applications as TCP socket applications is often impractical and cost-prohibitive. Besides, alternatives exist.

IBM introduced new technologies to help organizations preserve the investment in SNA and use IP as the protocol for connecting SNA computers. The technology is known as SNA/IP (*SNA over IP*) integration. The two endpoints, the SNA application in the mainframe and the SNA application in the remote location (branch, store), remain unchanged, thereby preserving the investment in SNA. SNA is stable, trusted, and relied upon for mission-critical business applications worldwide. A significant amount of the world's corporate data is handled by z/OS-resident SNA applications. A distinctive strength of SNA is that it is connection-oriented with many timers and control mechanisms to ensure reliable and secure delivery of data.

Mainframe IT organizations are often reluctant and skeptical about moving away from SNA, despite the allure of TCP/IP and web-based commerce. This reluctance is often justified. Rewriting stable, well-tuned business applications to change from SNA program interfaces to TCP/IP sockets can be costly and time consuming. Many organizations choose to use web-enabling technologies to make the vast amount of centralized data available to the TCP/IP-based web environment, while maintaining the SNA APIs. This *best of both worlds* approach ensures that SNA and VTAM will be around well into the future. Because SNA applications will exist for years to come, someone has to care for SNA definitions, problem determination, recovery, business continuity procedures, and many other tasks. These tasks are the responsibility of the mainframe networking systems programmer who needs to know in depth the architecture and how to implement SNA on various hardware and software platforms.

1.3 Mainframe network architecture

Basic elements of a computer network include hardware, software, and communication protocols. The inter-relationship of these basic elements constitutes the infrastructure of the network. If we think of a network as roads, highways, rails, and other means of transport, the network protocols are the traffic rules.

Connectivity is the pipeline through which data is exchanged between clients and servers via physical and logical communication interfaces and the network. The IBM System z provides a wide range of interface options for connecting your z/OS system to an IP network or to another IP host. Some interfaces offer point-to-point or point-to-multipoint connectivity, but others support local area network connectivity. The physical network interface is enabled through z/OS Communications Server (TCP/IP) definitions.

There are several ways to establish network connections with the mainframe. The IBM Open Systems Adapter-Express 5S (OSA-Express5S) and Open Systems Adapter-Express4S (OSA-Express4S) features that are available for an IBM System z mainframe computer connect to other servers and clients in 1000BASE-T Ethernet (10, 100, and 1000 Mbps), Gigabit Ethernet (GbE), and 10 Gigabit Ethernet environments. They provide redundancy capability and throughput improvements when running in Queued Direct I/O (QDIO) mode. QDIO mode allows direct access to central memory. QDIO mode can be emulated within a central processor complex (CPC) by allowing memory-to-memory data transfer among LPARs running IBM z/VM®, Linux, or z/OS.

The 10Gb IBM RoCE Express feature provides RDMA connectivity between hosts over a shared network segment using the SMC-R protocol.

z/OS Communications Server provides the data transportation corridor between the external network and the business applications running on z/OS. z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide-area networks, including the most popular wide area network, the Internet. z/OS Communications Server also provides performance enhancements that can benefit a variety of TCP/IP applications and includes several commonly used applications.

TCP/IP and SNA are both layered network models. Each can indirectly map to the international Open Systems Interconnect (OSI) network model, as shown in Figure 1-2.

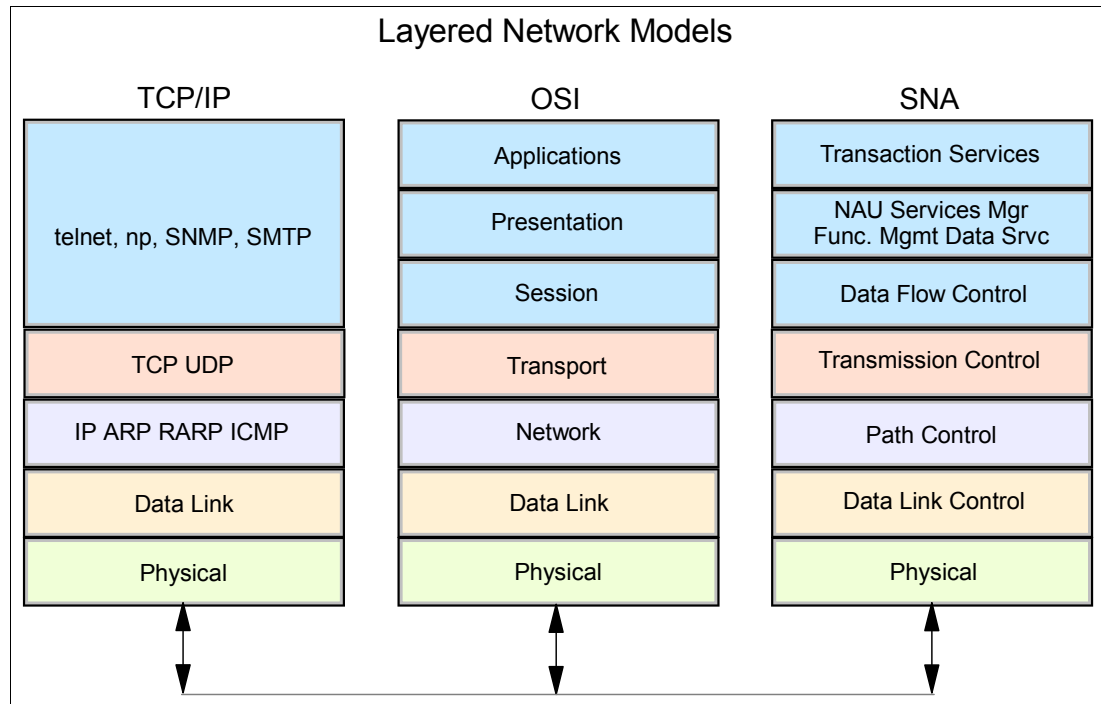


Figure 1-2 Open Systems Interconnect (OSI) network model

The OSI network model depicts the organization of the individual elements of technology involved with end-to-end data communication. As shown in Figure 1-2, the OSI network model provides some common ground for both SNA and TCP/IP. Although neither technology maps directly into the OSI network model (TCP/IP and SNA existed before the OSI network model was formalized), common ground still exists due to the defined model layers. The OSI network model is divided into seven layers. OSI layer 7 (Application) indirectly maps into the top layers of the SNA and TCP/IP stacks. OSI layer 1 (Physical) and layer 2 (Data Link) map into the bottom layers of SNA and TCP/IP stacks.

The z/OS Communications Server includes several sophisticated products and functions, including these major services:

- ▶ IP, using Transmission Control Protocol/Internet Protocol (TCP/IP).
- ▶ RDMA, using the SMC-R protocol over RoCE Express features, which is integrated with TCP/IP to provide seamless exploitation for TCP sockets applications.
- ▶ Systems Network Architecture (SNA), using Virtual Telecommunication Access Method (VTAM).

The Communications Storage Manager (CSM) component provides a shared I/O buffer for data flow. The CSM function allows authorized host applications to share data without having to physically move the data.

Figure 1-3 depicts the overall architecture of the z/OS Communications Server.

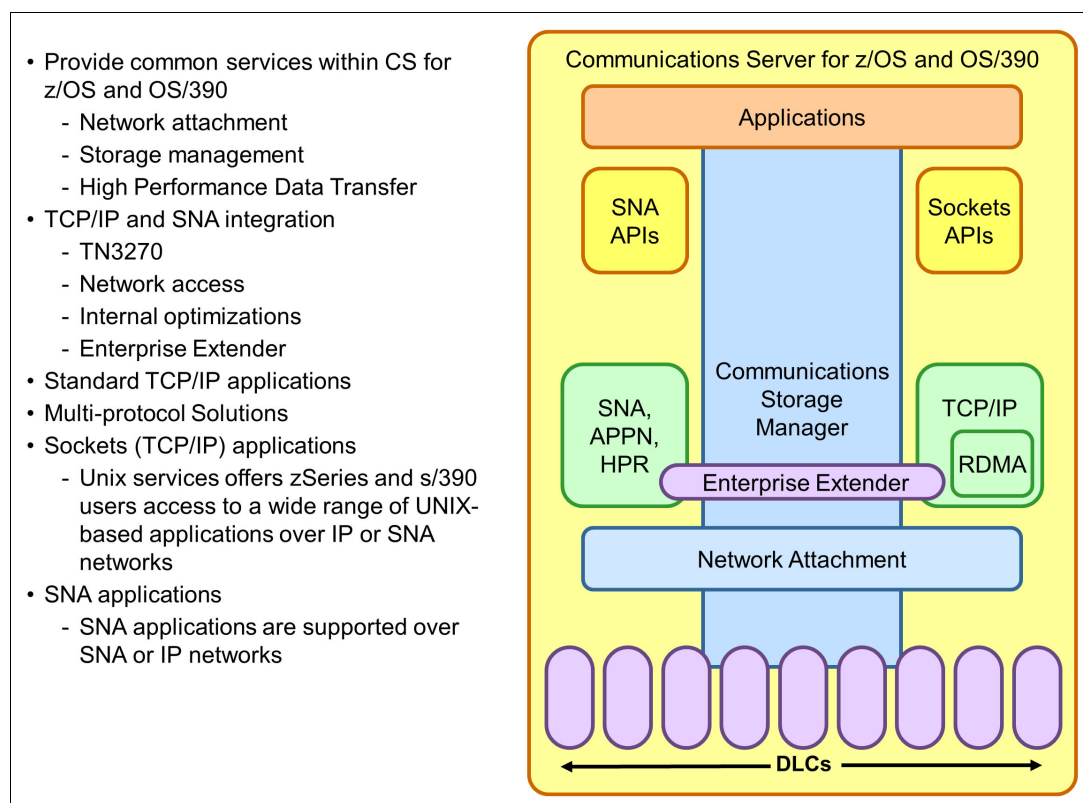


Figure 1-3 The z/OS Communications Server architecture

A resource in an SNA network is a *network accessible unit* (NAU), which is either an origin or a destination of information transmitted by the transport network (the data link control and path control layers). SNA consists of network accessible units such as system services control points, physical and logical units.

Physical units are components that manage and monitor resources, such as attached links and adjacent link stations, that are associated with a node. System services control points (SSCPs) indirectly manage these resources through physical units. A physical unit receives and acts upon requests from the SSCP, such as activating and deactivating links to adjacent nodes. It also manages links and link stations and accounts for the unique aspects of different link types.

Users and applications access SNA networks through *logical units* (LUs), which are the entry point through which users and applications access the SNA network. Logical units manage the exchange of data between end users and applications and application-to-application, acting as intermediaries between the two session partners on the two endpoint LUs.

Because SNA is a connection-oriented protocol, before transferring data the respective logical units must be connected in a session. In SNA hierarchical networks, logical units require assistance from SSCP to activate a session with another logical unit. A session between a logical unit (LU) and an SSCP is called SSCP-LU session. Control information flows from the SSCP to LU session. A session between two logical units either in the same node or in two different nodes is called an LU-LU session. The session between two LUs is used for application data flows. All node types can contain logical units. The control point assists in establishing the session between the two LUs and does not take part in the data transfer between the two LUs.

1.4 Networking hardware

Connectivity is the pipeline through which data is exchanged between clients and servers via physical and logical communication interfaces and the network. The IBM System z servers provide a wide range of interface options for connecting your z/OS system to an IP network or to another IP host, depicted in Figure 1-4. Some interfaces offer point-to-point or point-to-multipoint connectivity, and others support local area network (LAN) connectivity. This flow diagram depicts the physical interfaces (and device types) provided by System z servers. The physical network interfaces are enabled through z/OS Communications Server (TCP/IP) definitions.

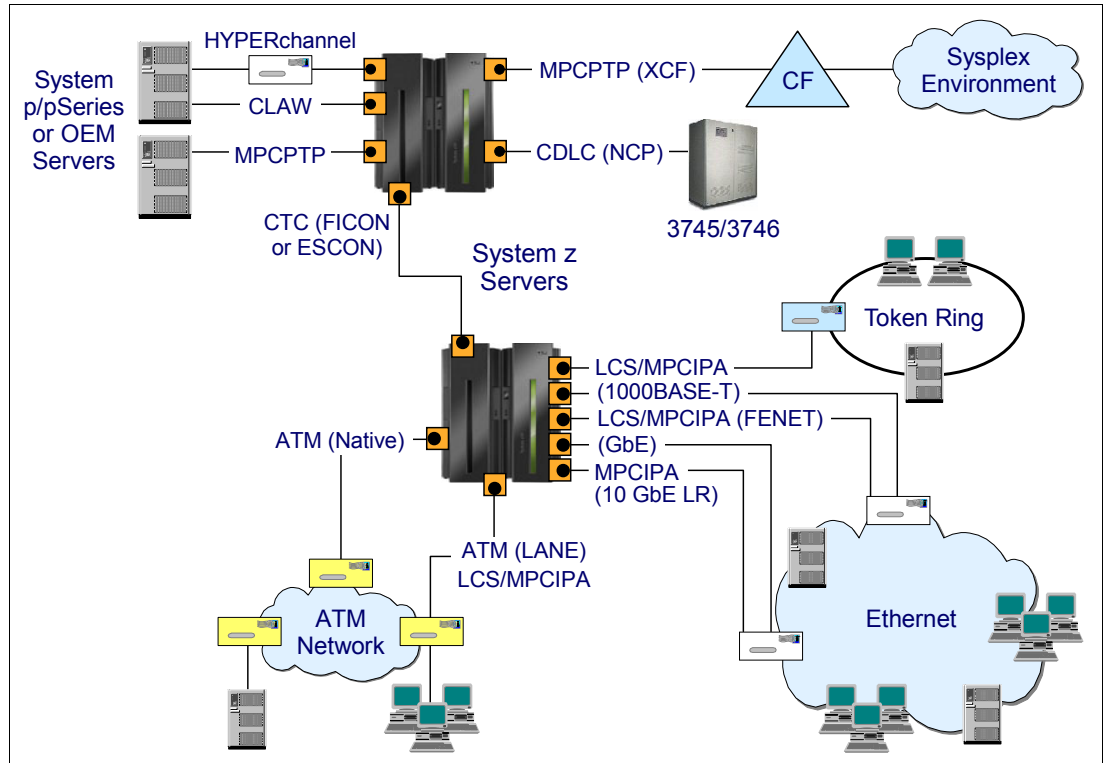


Figure 1-4 System z physical interfaces with Communications Server

Ethernet is a family of frame-based computer networking technologies for local area networks (LANs). It defines several wiring and signaling standards for the physical layer of the OSI networking model. Ethernet is standardized as IEEE 802.3. The combination of the twisted-pair versions of Ethernet for connecting end systems to the network, along with the fiber optic versions for site backbones, is the most widespread wired LAN technology. It has been in use since 1980, largely replacing competing LAN standards such as token ring, FDDI, and ARCNET.

Ethernet can support multiple networking protocols, including TCP/IP and RDMA over Converged Ethernet (RoCE).

ATM is a switching technology that provides fast, reliable, simultaneous transfer of data, voice, and video. VTAM supports ATM technology by enabling communication across ATM networks – both public (wide area networks, or WANs) and private (campus networks) – that are accessed through LAN emulation and native connections

Token ring local area network technology is a local area network protocol that resides at the data link layer (DLL) of the OSI model. It uses a special three-byte frame called a *token* that travels around the ring. Token ring frames travel completely around the loop.

IBM FICON provides all of the strengths of ESCON, plus more. The link data rate is 4 Gbps, with an expected effective data transfer rate of up to 350 MBps (full-duplex data transfer and large sequential read/write mix). The System z servers build on this I/O architecture by offering high-speed FICON connectivity.

The Enterprise Systems Connection (ESCON) channel is a high-bandwidth host attachment facility that can be used to connect SNA and non-SNA 3174 controllers, 3172 Nways interconnect controllers, 3746-900 controllers, and channel-attached hosts to VTAM running on MVS/ESA. ESCON provides bidirectional serial bit transmission, in which the communication protocol is implemented through sequences of special control characters and through formatted frames of characters. ESCON uses fiber optic cables for data transmission. The ESCON link data rate is 200 megabits per second (Mbps), which results in a maximum aggregate data rate of up to 17 megabytes per second (MBps). The maximum unrepeatable distance of an ESCON link using multimode fiber optic cables is 3 km (1.86 miles) when using 62.5 micron fiber.

1.4.1 Network connections

Network connections can be made in several different fashions. The mainframe originally relied upon the channel subsystem to offload I/O processing to channel programs. Disk storage is still accessed using ESCON channels in some places as most of them are migrating to faster FICON channels. But, for networking connectivity, OSA-Express cards offer better performance and availability. Connectivity is the pipeline through which data is exchanged between clients and servers via physical and logical communication interfaces and the network. The IBM System z servers provide a wide range of interface options for connecting your z/OS system to an IP network or to another IP host. Some interfaces offer point-to-point or point-to-multipoint connectivity, but others support local area network (LAN) connectivity.

The Open Systems Adapter is actually a network controller that you can install in a mainframe I/O cage. The adapter integrates several hardware features and supports many networking transport protocols. The OSA card is the strategic communications device for the mainframe architecture. It has several key features that distinguish it from CCW-based communications. The IBM Open Systems Adapter-Express 5S (OSA-Express5S) and Open Systems Adapter-Express4S (OSA-Express4S) features that are available on an IBM System z mainframe computer connect to other servers and clients in 1000BASE-T Ethernet (10, 100, and 1000 Mbps), Gigabit Ethernet (GbE), and 10 Gigabit Ethernet environments. They provide redundancy capability and throughput improvements when running in QDIO mode. QDIO mode allows direct access to central memory. QDIO mode can be emulated within a CPC by allowing memory-to-memory data transfer among LPARs running z/VM, Linux, or z/OS.

A new type of OSA, the OSN channel type, is only available with OSA-Express2 and later and requires an IBM z9® mainframe or later model. The primary intention of this type is to free organizations from the constraints of obsolete hardware: device types 3745 and 3746. The 374x device types, as they are called, are no longer manufactured or sold by IBM. A 374x host is required to run the Network Control Program (NCP). NCP is a significant functional component of subarea type SNA networks. The OSN channel type allows an Open Systems Adapter to communicate with an NCP using Channel Data Link Control protocol (CDLC). CDLC cannot be used over an OSD or OSE channel type, and even with channel type OSN it can only communicate to other LPARs within the CPC. Historically, 374x devices were often

connected to parallel or ESCON channels, which support CDLC. In this case, NCP will run on a software program called Communications Controller for Linux (CCL). And, as mentioned, both LPARs must be within the same CPC, because the data flows do not enter the network. In many cases, CCL provides the easiest way to migrate from older SNA-based network controllers to modern network devices.

The RoCE Express feature is supported on zEC12/zBC12 and newer hardware. It is a PCIe adapter that implements RDMA over converged Ethernet (RoCE), supporting the SMC-R protocol. Because SMC-R was designed for compatibility, you generally do not need to update or change your network switches to use SMC-R. The only advanced switch feature that SMC-R requires is the Global Pause frame described in the IEEE 802.3x standard.

1.5 Network protocols

In this section we take a closer look at both TCP/IP and SNA. More detailed information about security-related topics can be found in the individual chapters Chapter 3, “TCP/IP security” on page 89 and Chapter 4, “SNA security” on page 131.

1.5.1 TCP/IP

The single entity that handles, and is required for, all IP-based communications in a z/OS environment is the TCP/IP daemon itself. The TCP/IP daemon implements the IP protocol stack and runs a large number of IP applications to the same specifications as any other operating system might do. That's the beauty of TCP/IP. TCP/IP connectivity and gateway functions handle the physical interfaces and the routing of IP data packets called datagrams.

The network protocol layer provides the support for the IP protocol. All TCP and User Datagram Protocol (UDP) data goes through the IP layer when entering and leaving the host. TCP and UDP use the IPv4 routing layer or the IPv6 routing layer. The network layer also provides support for the Internet Control Message Protocol (ICMP) and ICMPv6. This is used by the IP layer to exchange information and error messages with IP layers on other hosts and routers. ICMP is used for the IPv4 protocol and ICMPv6 is used for the IPv6 protocol. The transport layer provides the support for the TCP, UDP, and RAW protocols. All three protocols use IPv4 or IPv6 as the network layer. The TCP protocol provides a connection-oriented, reliable transport layer, whereas UDP provides a simpler, connectionless and unreliable transport layer. The RAW transport layer provides for a more direct interface to the IP layer, which is primarily used by system-management type applications.

The file system layer provides the main interface between the application programming interfaces (APIs) and the transport layers. The first component of the file system layer is the z/OS UNIX logical file system (LFS). The LFS provides the API layer with a common interface to access files and sockets.

Each of the application programming interfaces (APIs) can be used to interface with the TCP/IP Services protocol stack provided by z/OS Communications Server. Like most networking software, TCP/IP is modeled in layers. This layered representation leads to the term *protocol stack*, which refers to the stack of layers in the protocol suite. It can be used for positioning (but not for functionally comparing) the TCP/IP protocol suite against others, such as Systems Network Architecture (SNA) and the Open System Interconnection (OSI) model. Functional comparisons cannot easily be extracted from this, because there are basic differences in the layered models used by the different protocol suites. By dividing the communication software into layers, the protocol stack allows for division of labor, ease of implementation and code testing, and the ability to develop alternative layer implementations.

Layers communicate with those above and below via concise interfaces. In this regard, a layer provides a service for the layer directly above it and uses services provided by the layer directly below it. For example, the IP layer provides the ability to transfer data from one host to another without any guarantee of reliable delivery or duplicate suppression. Transport protocols such as TCP make use of this service to provide applications with reliable, in-order data stream delivery.

TCP/IP stack

The TCP/IP address space is where the TCP/IP protocol suite is implemented for the z/OS Communications Server. The TCP/IP address space is commonly referred to as a stack. In earlier versions Open Edition (OE) TCP/IP could run either as a stand-alone or in parallel with the TCP/IP for MVS. This was often necessary because the OE stack did not support all the functions and network connections available with TCP/IP for MVS.

z/OS Communications Server IP now has a highly efficient direct communication between the UNIX System Services address space (OMVS) and a TCP/IP stack that was integrated in UNIX System Services. This communication path includes the UNIX System Services Physical File System (PFS) component for AF_INET and AF_INET6 (Addressing Family-Internet) sockets communication. Two configuration files are used by the TCP/IP stack, PROFILE.TCPIP and TCPIP.DATA. PROFILE.TCPIP is used only for the configuration of the TCP/IP stack. TCPIP.DATA is used during configuration of both the TCP/IP stack and applications.

The z/OS Communications Server allows a single TCP/IP address space to drive multiple instances of any supported device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration data set. The LINK statements show how to define a network interface link associated with the device and are included with the DEVICE statement for that device type. The new alternative to coding a DEVICE and a LINK statement is the INTERFACE statement to define network connectivity for the TCP/IP stack.

The TCP/IP address space operates as a transport provider for the INET physical file system. For this to occur, the TCP/IP system address space must connect to z/OS UNIX and become a z/OS UNIX process. Therefore, the started task UID that is assigned to the TCP/IP system address space must have a valid OMVS segment. As a transport provider, the TCP/IP address space requires superuser privileges in z/OS UNIX.

1.5.2 SMC-R

Remote Direct Memory Access (RDMA) is a communications technology that enables a host to make a subset of its memory directly available to a remote host. By doing so, data can be transferred between hosts efficiently and without any help from the CPU on the source or target host. Historically, RDMA has been confined to high-performance computing environments where the cost of maintaining RDMA-capable network fabrics such as InfiniBand was justified given the emphasis of performance over cost. However, RDMA is now available on standard Ethernet-based networks by using the industry (InfiniBand Trade Association) standard referred to as RDMA over Converged Ethernet (RoCE). With RoCE, the cost of adopting RDMA is lower because it can flow over the Ethernet fabrics that are already in place to carry IP network communications. Both standard TCP/IP and RDMA traffic can flow over the same physical LAN fabric at the same time, but RDMA network interface cards (RNICs, also referred to as RoCE host channel adapters (HCAs)), are required to do so. On System z, the 10Gb RoCE Express adapter serves as the RNIC.

z/OS Communications Server 2.1 introduced a new capability that combines the performance benefits of RDMA with the widely-used TCP/IP sockets programming interface. This function, called Shared Memory Communication - RDMA (SMC-R), allows your TCP sockets applications to benefit from direct, high-speed, low-latency, memory-to-memory (peer-to-peer) communications over RDMA transparently - no changes are required in your application programs.

SMC-R provides an enterprise class of services for RDMA that are designed for enterprise class data center networks. Communicating peers (the z/OS TCP/IP stacks) dynamically learn about the shared memory capability by using traditional TCP/IP connection establishment flows. With this awareness, the TCP/IP stacks can switch from TCP network flows to more efficient direct memory access flows that use RDMA. The application programs are unaware of the switch to shared memory communications.

1.5.3 SNA

Systems Network Architecture (SNA) is a data communication architecture established by IBM to specify common conventions for communication among the wide array of IBM hardware and software data communication products and other platforms. Among the platforms that implement SNA in addition to mainframes are the IBM Communications Server on Windows, AIX, and Linux, Microsoft Host Integration Server (HIS) for Windows, and many more. The way in which products internally implement these common conventions can differ from one product to another, but because the external interface of each implementation is compatible, different products can communicate without the need to distinguish among the many possible product implementations.

SNA products recognize and recover from loss of data during transmission, use flow control procedures to prevent data overrun and avoid network congestion, identify failures quickly, and recover from many errors with minimal involvement of network users. SNA products also increase network availability through options such as the extended recovery facility, backup host, alternative routing capability, and maintenance and recovery procedures integrated into workstations, modems, and controllers.

A networking application is dependent on the communication protocol it uses. Every protocol provides an application programming interface (API). TCP/IP's API is called *socket programming* and SNA has its own API. Migrating a networking application from one protocol to another (that is, from SNA to TCP/IP) requires replacing the calls to the API. Business managers are reluctant to invest in protocol conversion only for the sake of changing the underlying protocol without introducing new functions and improvements.

IBM introduced new technologies to help businesses preserve the investment in SNA and use IP as the protocol for connecting SNA computers. The technology is known as SNA/IP ("SNA over IP") integration. The two endpoints, the SNA application in the mainframe and the SNA application in the remote location (branch, store), remain unchanged, thereby preserving the investment in SNA. There are two implementations of SNA - Subarea networking and Advanced Peer-to-Peer Networking (APPN).

Subarea networking

Subarea networking was the initial implementation of SNA that defined mainframe-based hierarchical networks in which every resource and SNA route had to be predefined. In the initial implementation of SNA, adding resources or changing SNA routes necessitated the shutdown of parts of the network.

The initial implementation by IBM was the SNA subarea network. This network is a hierarchical network implemented by the Virtual Telecommunications Access Method (VTAM) in the mainframe. VTAM is the software that controls communication and data flow in an IBM mainframe SNA network. VTAM resides in the mainframe and supports a wide variety of network protocols, like SDLC and LAN.

VTAM controls data transfer between channels and OSA LAN-attached devices and performs SNA routing functions. VTAM provides an application programming interface (API) that enables the development of application programs that communicate using SNA with remote application programs or devices. Currently, VTAM is part of the z/OS Communications Server and is called SNA services. You'll find more information about VTAM in 1.6.1, "VTAM" on page 21.

SNA nodes

A data communication network can be described as a configuration of nodes and links. Nodes are the network components that send data over, and receive data from the network. Node implementations include processors, controllers, and workstations. Links are the network components that connect adjacent nodes. Nodes and links work together in transferring data through a network. An SNA node is a set of hardware and associated software components that implement network functions. Nodes differ based on the architectural components and the set of functional capabilities they implement. Nodes with different architectural components represent different node *types*.

- ▶ A T5 node is located only in the mainframe. The software that implements the T5 node is the SNA component of the Communications Server. The SNA component in z/OS is also referred to as VTAM (Virtual Telecommunications Access Method).
- ▶ A T4 node is a communication controller attached to peripheral nodes through communication lines or a LAN either to another communication controller through communication lines or to a mainframe through an ESCON or a parallel channel. IBM uses special hardware and software to implement the T4 Node. The software is the IBM network control program (NCP), and the hardware is the IBM 3745 or 3746 device. The Communication Controller of Linux (CCL) is a software package that replaces the 3745 or 3746.
- ▶ A T2.0 is a peripheral node that attaches to the communication controller or the mainframe. T2.0 is an alias for the IBM 3174 display controller, which attaches 3270 displays and printers and is connected through a communication line to the T4 node or through a channel to the T5 node. Additional devices that implement T2.0 node are banking branch controllers and retail store controllers.
- ▶ A T2.1 is a peer-oriented peripheral node that attaches to a mainframe, a communication controller, or another peripheral node. A T2.1 node is called a low entry networking (LEN) node.

Many of the SNA node types have been replaced by up-to-date hardware and software. The T2 node was replaced by a workstation (Windows or UNIX) that implements software called *3270 emulation* and the banking and retail controller by Windows, UNIX, or Linux-based servers. The 3745 and 3746 hardware is nearing its end of life. The migration to TCP/IP in the backbone reduces the number of lines in the 3745 and 3746. The OSA and the routers can implement most of the functions of the 3745 and 3746 at much lower cost. One of the alternatives to the 3745/3746 hardware is the IBM Communication Controller of Linux (CCL) software package implemented on the mainframe. CCL uses OSA for NCP (OSN) or an OSA port operating in LCS mode and routers.

System services control point (SSCP)

A type 5 subarea node contains a *system services control point* (SSCP). An SSCP activates, controls, and deactivates network resources in a subarea network. To control and provide services for its subordinate nodes, an SSCP establishes *sessions* with components in the network.

Advanced Peer-to-Peer Networking network topology

To address the deficiency of the static nature of subarea SNA, IBM introduced an SNA-based peer network, with no hierarchical relations, and with dynamic definition of resources. At a later stage, APPN was enhanced with the introduction of High Performance Routing (HPR) and Enterprise Extender (SNA/IP), which, as its name implies, is a high performance routing protocol that can be optionally used by APPN.

Hierarchical systems are organized in the shape of pyramid, with each row of objects linked directly to objects beneath it. SNA subarea, besides implementing the model of a hierarchical system, is centrally managed from the top of the pyramid. Network resources in SNA are managed (that is, known and operated) from a central point of control that is aware of all the activity in the network, whether a resource is operational, and the connectivity status of the resource. The resources can send reports on their status to the control point. Based on networking and organizational requirements, a hierarchical network can be divided into subnetworks, where every subnetwork has a control point with its controlled resources.

Advanced Peer-to-Peer Networking (APPN) is a type of data communications support that routes data in a network between two or more systems that do not need to be directly connected. APPN topology does not have a subarea number or have exclusive ownership of the SNA resources. Each APPN-participating VTAM is included in a geographically dispersed collection of shared SNA resources, eliminating the need for a cross-domain resource manager to establish sessions. APPN includes a high-performance routing (HPR) method of sending SNA application data through existing TCP/IP network equipment. APPN includes a function called Enterprise Extender (EE), sometimes referred to as HPR/IP. EE ensures that SNA applications can be served by state-of-the-art IP networking technology.

HPR is not limited exclusively to SNA/APPN over TCP/IP networks. Rather, HPR is the APPN function that provides high-performance delivery of data through an APPN network, combined with the high-availability feature of dynamic rerouting of sessions around failures in the network. But HPR is supported over most types of APPN connections (not just APPN over TCP/IP). Enterprise Extender (EE) is the APPN/HPR function that allows SNA sessions and other APPN functions (like HPR) to work over a TCP/IP network (rather than a native SNA network).

Depending on the software that implements APPN in T2.1 nodes, the node can be configured in the APPN networks with varying complexity, from the simplest case of an isolated pair of low-entry networking nodes to a large APPN network. Using low-entry networking or APPN protocols, any node can control the establishment and termination of sessions.

The following node types can be implemented by T2.1 nodes:

- ▶ Low-entry networking (LEN)
- ▶ APPN end node (EN)
- ▶ APPN network node (NN)

To ease the migration from subarea networking to APPN in the mainframe, the following nodes types can be implemented in a mainframe:

- ▶ Interchange node (ICN)
- ▶ Migration node

T5 and T4 nodes also support low-entry networking and APPN protocols, and are fully compatible with T2.1 nodes in these contexts. They also introduce product features of their own, related to enhanced subarea-APPN interchanges. When a subarea node implements either APPN or low-entry networking protocols, it acts as a T2.1 node and can still implement, depending on VTAM's definitions, the subarea T5 and T4 functions.

Architectural components of the SNA network

A resource in an SNA network is a *network accessible unit*, which is either an origin or a destination of information transmitted by the transport network (the data link control and path control layers). You already read about control points and system services control points, which are network accessible units. Other network accessible units are:

Physical units

Physical units are components that manage and monitor resources such as attached links and adjacent link stations associated with a node. SSCPs indirectly manage these resources through physical units. Physical units (PUs) exist in subarea and type 2.0 nodes. (In type 2.1 peripheral nodes, the control point performs the functions of a PU.) The PU supports sessions with control points in type 5 nodes and also interacts with the control point in its own node.

A physical unit provides the following functions:

- Receives and acts upon requests from the system services control point (SSCP), such as activating and deactivating links to adjacent nodes
- Manages links and link stations and accounts for the unique aspects of different link types

Logical units

Users and applications access SNA networks through *logical units* (LUs), which are the entry point through which users and applications access the SNA network. Logical units manage the exchange of data between users to applications and application to application, acting as intermediaries between the two session partners on the two endpoint LUs. Because SNA is a connection-oriented protocol, before transferring data the respective logical units must be connected in a session. In SNA hierarchical networks, logical units require assistance from system services control points (SSCPs), which exist in type 5 nodes, to activate a session with another logical unit.

A session between a logical unit (LU) and an SSCP is called an *SSCP-LU* session. Control information flows from the SSCP to LU session. A session between two logical units either in the same node or in two different nodes is called an *LU-LU* session. The session between two LUs is used for application data flows. All node types can contain logical units. In SNA hierarchical networks, the logical unit has sessions with only one control point in type 5 nodes and with logical units in other nodes. A control point assists in establishing a session between its managed LU and an LU. It does not manage in a different node.

SNA defines different kinds of logical units called *LU types*. LU types identify sets of SNA functions that support user communication. LU-LU sessions can exist only between logical units of the same LU type. For example, an LU type 2 can communicate only with another LU type 2; it cannot communicate with an LU type 3. The following list explains the LU types that SNA defines, the kind of configuration or application that each type represents, and the hardware or software products that typically use each type of logical unit:

► LU type 1

This is for application programs and single-device or multiple-device data processing workstations communicating in an interactive or batch data transfer. An example of the use of LU type 1 is an application program running under IMS/VS and communicating with a 3270 printer.

► LU type 2

This is for application programs and display workstations communicating in an interactive environment using the SNA 3270 data stream. An example of the use of LU type 2 is an application program running under IMS/VS and communicating with an IBM 3270 display station at which an user is creating and sending data to the application program.

► LU type 3

This is for application programs and printers using the SNA 3270 data stream. An example of the use of LU type 3 is an application program that runs under IBM Customer Information Control System/Virtual Storage (CICS/VS) and sends data to a 3270 printer.

► LU type 6.2

This is for transaction programs communicating in a client/server data processing environment. The type 6.2 LU supports multiple concurrent sessions. LU 6.2 can be used for communication between two type 5 nodes, a type 5 node and a type 2.1 node, or two type 2.1 nodes. Examples of the use of LU type 6.2 are:

- An application program running under CICS in a z/OS system communicating with another application program running under CICS in another z/OS system.
- An application program in a Microsoft Host Integration Server (HIS) or IBM AIX Communications Server communicating with a CICS in a z/OS system.

One of the main reasons for networks to migrate to APPN is the implementation of a sysplex. A sysplex environment provides some unique advantages to the VTAM installation, namely generic resources (GR) and multinode persistent sessions (MNPS). With the latest emphasis on IP application load balancing within the sysplex through the use of Sysplex Distributor, and SNA application load balancing within the sysplex through the use of generic resources, migrating to an HPR/IP (EE) infrastructure has become a natural step toward supporting SNA applications through native IP connectivity to the mainframe. HPR/IP accommodates the presence of SNA data within the IP packet, and uses the dynamic routing capabilities of both HPR and IP.

APPN/HPR improves the availability of sysplex resources. Generic resources enable multiple application copies within a sysplex to present a single image to the user, resulting in better performance through load balancing and improved availability. The multinode persistent sessions function enables sessions to survive the failure of a VTAM, or a z/OS system, or even a processor, without disruption. You can use an IP network for SNA sessions with Enterprise Extender, which provides enablement of IP applications and convergence on a single network transport and preserves SNA application and endpoint investment. An EE connection is a logical connection that represents IP connectivity from a host to a specified IP address or host name. Conceptually, an IP network looks like an APPN/HPR transmission group (TG) in a session route.

1.6 Additional network components

This section describes additional components within the z/OS Communications Server.

1.6.1 VTAM

In z/OS, VTAM provides the SNA layer network communication stack to transport data between applications and the user. VTAM manages the SNA-defined resources, establishes sessions between these resources, and tracks session activity.

VTAM performs several tasks in a network, for example:

- ▶ It monitors and controls the activation and connection of resources.
- ▶ It establishes connections and manages the flow and pacing of sessions.
- ▶ It provides application programming interfaces (for example, an APPC API for LU 6.2 programming) that allow access to the network by user-written application programs and IBM-provided subsystems.
- ▶ It provides interactive terminal support for the Time Sharing Option (TSO).
- ▶ It provides support for both locally and remotely attached resources.

z/OS runs only one VTAM address space. Each application that uses VTAM, such as CICS/TS, requires a VTAM definition. The application and VTAM use this definition to establish connections to other applications or users.

What HTML is to a web application and browser, the 3270 data stream is to an SNA application and device in an LU-LU session. Specialized commands are embedded in the data of panel devices and printers. The 3270 data stream is data with these embedded instructions and data field descriptors. The 3270 data stream commands are created and read by SNA applications, such as Physical Unit (PU) controllers managing the displays, printers, and TN3270 emulators available in AIX and PC operating systems.

Each endpoint of an SNA session is known as a logical unit (LU). An LU is a device or program by which a user (application program, a terminal operator, or an input/output mechanism) gains access to the SNA network. VTAM-established sessions are known as LU-to-LU sessions. In an SNA network, CICS/TS, for example, is considered an LU and typically has many sessions with other LUs, such as displays, printers, POS devices, and other remote CICS/TS regions. Each LU is assigned a unique network addressable unit (NAU) to facilitate communication.

A physical unit (PU) controls one or more LUs. A PU is not literally a physical device in the network. Rather, it is a portion of a device (usually programming or circuitry, or both) that performs control functions for the device in which it is located and, in some cases, for other devices that are attached to the PU-containing device.

A PU exists in each node of an SNA network to manage and monitor the resources (such as attached links and adjacent link stations) of the node. The PU exists either within the device or within an attached controlling device. VTAM must activate the PU before it can activate and own each LU attached to the PU.

Although TCP/IP is by far the most common way to communicate over a network with a z/OS host, some environments still use native SNA, and many environments now carry (encapsulate) SNA traffic over UDP/IP. The hierarchical design of SNA serves the centralized data processing needs of large enterprises.

At the top of this hierarchy is VTAM. VTAM serves the following types of network topologies:

- ▶ Subarea
- ▶ Advanced Peer-to-Peer Networking (APPN)
- ▶ Subarea/APPN mixed

The part of VTAM that manages a subarea topology is called the System Services Control Point (SSCP). The part of VTAM that manages APPN topology is called the Control Point (CP).

Virtual Telecommunications Access Method (VTAM) controls the network and maintains a table of all the machines and phone links in the network. It selects the routes and the alternate paths that messages can take between different NCP nodes. A subarea is the collection of terminals, workstations, and phone lines managed by an NCP. Generally, the NCP is responsible for managing ordinary traffic flow within the subarea, and VTAM manages the connections and links between subareas. Any subarea network must have a mainframe. The SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing protocols.

The communications storage manager (CSM) is a VTAM component that allows authorized host applications to share data with VTAM and other CSM users without the need to physically copy the data. CSM includes an API that provides a way to obtain and return CSM buffers, change ownership of buffers, copy buffers, and manage CSM buffers. VTAM supports the attachment of a sysplex to a network. Sessions into the sysplex can be established from either subarea nodes or APPN nodes. Several VTAM functions are available only in a sysplex environment. If a coupling facility and an APPN or mixed APPN and subarea environment exists in the sysplex, the user can take advantage of the VTAM functions. VTAM also provides support for TCP/IP functions that need access to a coupling facility.

VTAM supports the following types of LANs through the IBM Open Systems Adapter (configured in SNA mode):

- ▶ Ethernet or Ethernet-type LAN
- ▶ Token-ring network
- ▶ Fiber distributed data interface (FDDI)

Resources in a VTAM network are classified under major and minor nodes. Each major node can contain one or more minor nodes. To define the attachment of any device, set of devices, various tables, or search order of resources to VTAM, you need to code major node members and store them in the VTAMLST partitioned data set.

The start options are stored in the SYS1.VTAMLST partitioned data set (PDS) or any other PDS concatenated to the VTAMLST DD statement. The member name is ATCSTRxx, where xx specifies the identifier of the start option file. Start options provide information about the conditions under which VTAM runs. They also enable you to tailor VTAM to meet your needs each time VTAM is started. Many operands can have defaults specified as start options, thus reducing the amount of coding required. Many start options can be dynamically modified and also displayed. The ATCSTRxx member contains VTAM start parameters that define the VTAM's identity in the network, plus tuning options and a pointer to the major node startup list, ATCCONxx. A configuration list specifies the resources that are to be activated when VTAM is started. The member names of the resources you want to have activated when VTAM starts are stored in the ATCCONxx member in the VTAM definition library, where xx is any two alphanumeric characters.

1.6.2 TCP/IP stack and functions

The highest level protocols within the TCP/IP protocol stack are application protocols. They communicate with applications on other internet hosts and are the user-visible interface to the TCP/IP protocol suite. All application protocols have some characteristics in common. They can be user-written applications or applications standardized and shipped with the TCP/IP product. The TCP/IP protocol suite includes application protocols such as:

- ▶ Telnet for interactive terminal access to remote internet hosts
- ▶ File Transfer Protocol (FTP) for high-speed disk-to-disk file transfers
- ▶ Simple Mail Transfer Protocol (SMTP) as an internet mailing system

These are some of the most widely implemented application protocols, but many others exist. Each particular TCP/IP implementation will include a lesser or greater set of application protocols.

Telnet

Telnet is a terminal emulation protocol. With Telnet, users can log on to remote host applications as though they were directly attached to that host. Telnet protocol requires that the user have a Telnet client that emulates a type of terminal that the host application can understand. The client connects to a Telnet server, which communicates with the host application. The Telnet server acts as an interface between the client and host application. A PC can support several clients simultaneously, each with its own connection to any Telnet server.

The original basic Telnet protocol was defined in RFC 854. This RFC effectively defined all that was needed to support the 3270 data stream, because the 3270 data stream is just part of the Telnet data payload. In other words, the 3270 portion was implemented outside of (above) the Telnet protocol. Specific options could be negotiated (beyond basic Telnet) using the Telnet option standard of RFC 855. Option negotiation in turn allowed for device type negotiations (later formally defined in RFC 1091) to be completed as part of the Telnet session setup.

There are two types of Telnet servers:

- ▶ **TN3270E Telnet server**

This type provides access to z/OS VTAM SNA applications on the MVS host by using Telnet TN3270E, TN3270, or Linemode protocol. A TN3270E client uses the TN3270E protocol to access the resources on a TN3270E server. However, the TN3270E client cannot complete the connection all the way to the target application because the TN3270E client communicates according to the TN3270E protocol but the target application expects communication to be SNA protocol.

- ▶ **z/OS UNIX Telnet server**

This type provides access to z/OS UNIX shell applications on the MVShost by using Telnet Linemode protocol.

File Transfer Protocol (FTP)

The File Transfer Protocol (FTP) allows a user to copy files from one machine to another. The protocol allows for data transfer between the client (the user) and the server in either direction. In addition to copying files, the client can issue FTP commands to the server to manipulate the underlying file system of the server (for example, to create or delete directories, delete files, rename existing files, and so on). FTP is the most frequently used TCP/IP application for moving files between computers. FTP is built on a client/server architecture and uses separate control and data connections between the client and server.

From an FTP user's point of view, the link is connection-oriented. FTP uses TCP as a transport protocol to provide reliable end-to-end connections. Both hosts must run TCP/IP to establish file transfer. FTP is used to transfer files between TCP/IP hosts. The FTP client is the TCP/IP host that initiates the FTP session; the FTP server is the TCP/IP host to which the client connects.

The z/OS model for the FTP server includes a daemon process and a server process. The daemon process starts when you start your cataloged procedure (for example, START FTPD), and it listens for connection requests on a specific port. When the daemon accepts an incoming connection, it creates a new process (server's address space) for the FTP server, which handles the connection for the rest of the FTP login session. Each login session has its own server process.

z/OS UNIX sockets

Sockets are used for local (intrahost) and remote (interhost or network) communication between UNIX processes. Simply consider the concept of an API as being the set of C library service calls used to work with a particular resource. Although sockets and HFS files are both streams, the complexity of socket handling requires more powerful syntax and options for handling socket data. This is relevant in a later discussion where you'll see that z/OS has perhaps eight different socket APIs. A powerful design principle was used in UNIX data processing: Any type of data manipulated by a process can be represented by a standard file/stream structure. Also, a single set of program calls is used to open streams, read and update data, and close streams to insulate the program from dealing with the physical origin of stream data.

The Resolver

The resolver acts on behalf of programs as a client that accesses name servers for name-to-address or address-to-name resolution. The resolver can also be used to provide protocol and services information. To resolve the query for the requesting program, the resolver can use information obtained from any of the following sources:

- ▶ Available name servers
- ▶ Its system-wide resolver cache of name server data
- ▶ Local definitions, such as /etc/resolv.conf, /etc/hosts, /etc/ipnodes, HOSTS.SITEINFO, HOSTS.ADDRINFO, or ETC.IPNODES

Whether the resolver uses name servers, and how, is controlled by TCPIP.DATA statements (resolver directives). When system-wide caching is enabled, the resolver can also use DNS response information that has been cached locally.

Resolver flows show the information request and response flows as the resolver gets a request, and based on its own configuration file either looks at a local hosts file or sends a request to a DNS server. After the relationship between the host name and IP address is established, the Resolver returns the response to the application. In a z/OS system, this task is more complex, because the applications can be built using one of three main groups of sockets API environments:

- ▶ Native TCP/IP sockets
- ▶ UNIX System Services callable sockets (BPX1xxxx calls)
- ▶ IBM Language Environment® C/C++ sockets

To initiate a request to the resolver in z/OS, an application executes a set of commands based on the Sockets API Library the application used to generate the socket to connect to the TCP/IP stack.

1.6.3 Enterprise Extender

Enterprise Extender (EE) has provided a useful solution to the dilemma of running SNA applications over IP networks. The Enterprise Extender (EE) technology was implemented on most IBM networking platforms during 1998. Its main objective is to provide SNA-over-IP integration that is significantly superior to its predecessors, such as data link switching

(DLSw) and IBM AnyNet®. AnyNet is no longer supported in the z/OS Communications Server V1R8 and later. Enterprise Extender has provided a useful solution to the dilemma of running SNA applications over IP networks. “Extending the enterprise” is an appropriate description. Enterprise Extender is a standard created by the Internet Engineering Task Force (IETF) and APPN Implementers’ Workshop (AIW). It is documented in RFC 2353.

Because of its design, the EE architecture is extremely flexible. It can be used in all networks from the smallest to the largest, and provides the network architect with a wide choice of locations within the network to place the SNA/IP boundary. EE uses all the latest routing performance enhancements provided by APPN/HPR.

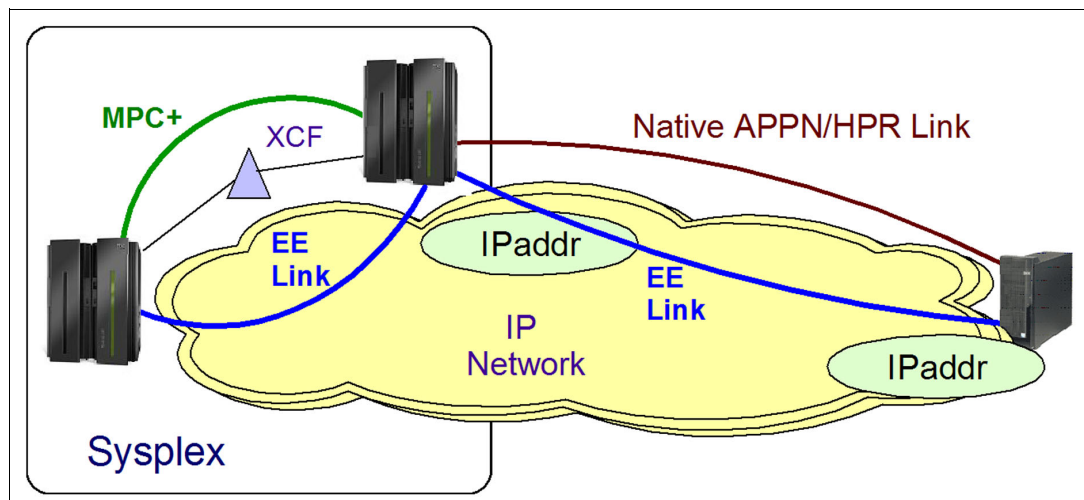


Figure 1-5 Enterprise Extender

EE uses APPN's High-Performance Routing (HPR) technology to provide encapsulation of SNA application traffic within UDP frames by HPR-capable devices at the *edges* of an IP network. These edge devices are called *endpoints*. This is illustrated in Figure 1-5. To the IP network, the SNA traffic is UDP datagrams that get routed without hardware or software changes to the IP network. To the SNA application, the session is normal SNA with predictable performance and high availability. By wrapping the SNA application in this way, EE enables SNA data to be carried over an IP network without changing either the SNA applications or the IP hardware.

EE architecture

The Enterprise Extender architecture carries the SNA High-Performance Routing (HPR) traffic of any logical unit type over an IP infrastructure without requiring changes to that infrastructure. It treats the IP network as a particular type of SNA logical connection. In this manner, the SNA protocols act as transport protocols on top of IP, as does any other transport protocol such as Transmission Control Protocol. An important aspect of Enterprise Extender is the ability to view the IP network as an APPN connection network. In this case, the benefit comes from the ability to establish dynamically a single one-hop HPR link to any host to which IP connectivity is enabled, provided that the host implements Enterprise Extender. In general, this allows the routing function to be handled entirely within IP. IP routers serve as the only routing nodes (hosts) in the network.

The implementation of EE in z/OS involves data transfer between the VTAM and the TCP/IP address spaces. A special connection type called IUTSAMEH is used to move data from VTAM to TCP/IP and vice versa. This connection type is used to connect two or more Communications Server for z/OS IP stacks running on the same MVS image. In addition, it is used to connect the z/OS Communications Server IP stacks to z/OS VTAM for use by

Enterprise Extender. For Enterprise Extender, z/OS Communications Server implements a separate UDP layer called “Fast UDP” that is optimized for Enterprise Extender communication. Fast UDP, communicates with Enterprise Extender (the APPN over UDP component in VTAM through the IUTSAMEH device.

1.6.4 TN3270/E

During the last several decades, before the Internet Protocol’s rise in popularity, large organizations established their own SNA networks. These SNA networks were used to communicate between remote users and the centralized mainframe. The display management protocol used to facilitate this communication within an SNA environment was called the 3270 data stream. At the user’s location in an SNA network was a device referred to as a 3270 terminal.

A 3270 terminal was a non-programmable (sometimes called “dumb”) workstation. Stated more simply, it was a monitor with a keyboard attached. The 3270 terminal had only rudimentary communications capabilities and was text-based. One of the earliest model 3270 terminal displays (3278 model 1) consisted of 12 rows and 80 columns of text characters, no more and no less. Eventually, a 24 x 80 screen size became the standard, with some alternate sizes available.

Today, a single instance of the TN3270E server can support up to 128 000 emulated 3270 display terminals. Display terminals are emulated in software called *TN3270E clients*, which can run on a standard personal computer or workstation. In the world of genuine 3270 display terminals, 128,000 users would require an enormous amount of dedicated, limited function keyboard and display devices, not to mention 2000 dedicated 3174 control units. As network use exploded, the SNA 3270 method of communicating with a mainframe became untenable. The solution came in the form of the Internet Protocol (IP).

A TN3270E client uses the TN3270E protocol to access the resources on a TN3270E server. However, the TN3270E client cannot complete the connection all the way to the target application because the TN3270E client communicates according to the TN3270E protocol, but the target application expects communication to be SNA protocol. In effect, the TN3270E server is nothing more than a protocol converter: on one side it maintains an RFC 2355-compliant TN3270E session; on the other side, it emulates a 3270 data stream terminal (including the 3174 control unit) to VTAM. The target application cannot tell the difference between a genuine 3270-attached terminal and a TN3270E server-emulated terminal. Among the many capabilities of TN3270E, one is the ability of the TN3270E client specifically to request an LU to be used as the terminal LU. This level of control allows more control, from an SNA point of view.

1.6.5 Special features

The IBM z/OS Communications Server provides some special functions to enable high availability, load balancing and performance. This section describes some of those functions.

Virtual IP Address (VIPA)

A VIPA is a generic term that refers to an internet address on a z/OS host that is not associated with a physical adapter. There are two types of VIPAs:

- ▶ A *static* VIPA cannot be changed except through an operator command.
- ▶ A *dynamic* VIPA (DVIPA) can move to other TCP/IP stack members in a sysplex or it can be activated by an application program or by a supplied utility. Dynamic VIPAs are used to implement sysplex distributor.

The virtual IP address (VIPA) removes the adapter as a single point of failure by providing an IP address that is associated with a stack without associating it with a specific physical network attachment. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it. Also, the TCP/IP stack does not maintain interface counters for VIPA interfaces (virtual links).

Dynamic Virtual IP Address (DVIPA)

DVIPA is part of the evolution of the VIPA feature that we described. The VIPA mentioned in the previous section was static. It is defined through a DEVICE and LINK statement pair and remains unchanged unless explicitly removed by changing the active configuration statements. By contrast, a dynamic VIPA would normally be activated in one of two ways:

- ▶ An application explicitly issuing a bind() function call to the IP address. This is called *unique application-instance DVIPA*.
- ▶ A TCP/IP stack dynamically activating the address. This is called *multiple application-instance DVIPA*.

In order for TCP/IP to communicate DVIPA status among LPARs, TCP/IP uses an XCF group called EZBTCPCS. DVIPA is beneficial, from a network perspective due to following reasons:

- ▶ DVIPAs allow servers to be made available independently of hardware or software failures. This can be done dynamically by TCP/IP or even by a system automation product.
- ▶ DVIPA allows multiple LPARs to appear to be a single, highly available network host.
- ▶ Because DVIPA movement is automatic, users and clients might never know a DVIPA address movement has occurred.
- ▶ With DVIPA, applications can be seamlessly moved from one LPAR to another, allowing a virtualization of the application itself.

A distributed DVIPA, which is a special type of DVIPA, can distribute connections within a sysplex. Dynamic VIPAs are designed to interoperate with a dynamic routing daemon. Therefore, it is highly recommended that a dynamic routing daemon (such as OSPF) be used on a z/OS host that uses VIPAs.

Sysplex Distributor

The Sysplex Distributor can be viewed as an evolution of connectivity improvements. It is a combination of the high availability features of DVIPA and the workload optimization capabilities of WLM. The implementation has one significant difference: Rather than all participating hosts being effectively equal, LPARs can be given specific roles to play. When combined with WLM, the overall effect on availability is exceptional. Another change with sysplex distributor is that the distribution is possible only with TCP connections. Other Layer 4 protocols are not supported.

VMAC

Before the introduction of the *virtual* MAC function, an OSA interface only had one MAC address. This restriction caused problems when using load balancing technologies in conjunction with TCP/IP stacks that share OSA interfaces. The single MAC address of the OSA also causes a problem when using TCP/IP stacks as a forwarding router for packets destined to unregistered IP addresses.

VMAC support enables an OSA interface to have not only a physical MAC address, but also many distinct virtual MAC addresses for each device or interface in a stack. That is, each stack can define up to eight VMACs per protocol (IPv4 or IPv6) for each OSA interface. With the use of VMACs, forwarding decisions in the OSA can be made without having to involve

OSI Layer 3 level (network layer/IP layer). From a LAN perspective, the OSA interface with a VMAC appears as a dedicated device or interface to a TCP/IP stack. Packets destined for a TCP/IP stack are identified by an assigned VMAC address and packets sent to the LAN from the stack use the VMAC address as the source MAC address. This means that all IP addresses associated with a TCP/IP stack are accessible using their own VMAC address, rather than sharing a single physical MAC address of an OSA interface.

QDIO

QDIO is a highly efficient data transfer mechanism that is designed to dramatically reduce system overhead and improve throughput by using system memory queues and a signaling protocol to directly exchange data between the OSA microprocessor and network software. QDIO is the interface between the operating system and the OSA hardware. The components that make up QDIO are DMA, data router, priorityqueuing, dynamic OSA Address Table building, LPAR-to-LPAR communication, and Internet Protocol (IP) Assist functions. QDIO supports both IP and non-IP traffic.

HiperSockets

HiperSockets implementation is based on the OSA-Express Queued Direct Input/Output(QDIO) protocol, hence HiperSockets is called internal QDIO (iQDIO). The Licensed Internal Code (LIC) emulates the link control layer of an OSA-Express QDIO interface. Typically, before you can transport a packet on an external LAN, you have to build a LAN frame, and insert the MAC address of the destination host or router on that LAN into the frame. HiperSockets do not use LAN frames, destination hosts, or routers. TCP/IP stacks are addressed by inbound data queue addresses rather than MAC addresses.

The System z LIC maintains a lookup table of IP addresses for each HiperSocket. This table represents an *internal LAN*. When a TCP/IP stack starts a HiperSockets device, the device is registered in the IP address lookup table with its IP address, and its input and output data queue pointers. If a TCP/IP device is stopped, the entry for this device is deleted from the IP address lookup table. HiperSockets copy data synchronously from the output queue of the sending TCP/IP device to the input queue of the receiving TCP/IP device by using the memory bus to copy the data through an I/O instruction.

The controlling operating system that performs I/O processing is identical to OSA-Express in QDIO mode. The data transfer time is similar to a cross-address space memory move, with latency close to zero. To get a data move total elapsed time, you have to add the operating system I/O processing time to the LIC data move time.

HiperSockets operations are executed on the central processor (CP) where the I/O request is initiated. HiperSockets starts read or write operations. The completion of a data move is indicated by the sending side to the receiving side with a Signal Adapter (SIGA) instruction. Optionally, the receiving side can use dispatcher polling rather than handling SIGA interrupts. The I/O processing is performed reducing demand on the System Assist Processor (SAP). This new implementation is also called *thin interrupt*.

Open Systems Adapter Support Facility (OSA/SF)

The TCP/IP subagent can retrieve SNMP management data from the Open Systems Adapter Support Facility (OSA/SF) for several OSA adapters. The OSA product also provides an SNMP subagent, the OSA-Express Direct subagent, which supports management data for OSA-Express adapters. The OSA-Express Direct subagent can be used with Communications Server SNMP support to retrieve the management data. Use the OSA-Express Direct subagent for OSA management data, rather than the TCP/IP subagent, because the OSA-Express Direct subagent communicates directly with the OSA-Express adapters and does not require the OSA/SF and IOASNP applications.

OSA/SF includes a Java-based graphical user interface (GUI) in support of the client application. The Java GUI is independent of any operating system or server (transparent to the operating system), and is expected to operate wherever the current Java runtimes are available. Use of the GUI is optional. A REXX command interface is also included with OSA/SF. OSA/SF is not required to set up the OSA features in QDIO mode (CHPID type OSD), but it can be used for monitoring and controlling ports. OSA/SF has been, and continues to be, integrated in z/OS, z/VM, and IBM z/VSE®, and runs as a host application. For OSA/SF, Java GUI communication is supported via TCP/IP only. In the past, communication was supported via EHLLAPI (3270), APPC, and TCP/IP. This integrated version of OSA/SF is a complete replacement for the currently integrated versions in z/OS, z/VM, and z/VSE. This version of OSA/SF is not being offered as a separately orderable program product.

The OSA/SF is used primarily for the following functions:

- ▶ Manage all OSA ports.
- ▶ Configure all OSA non-QDIO ports.
- ▶ Configure local MAC.
- ▶ Display registered IPv4 addresses (in use and not in use). It is supported on System z servers for QDIO ports.
- ▶ Display registered IPv4 or IPv6 Virtual MAC and VLAN ID associated with all OSA Ethernet features configured as QDIO Layer 2.
- ▶ Provide status information about an OSA port (shared or exclusive use state). This support is applicable to all OSA features on System z servers. With z/OS, a second interface using a set of REXX EXECs through the Time Sharing Option Extensions (TSO/E) can be used to control the OSA features defined to System z servers on which the TSO/E is running.

1.7 Network tools and products

With such a complex architecture that provides a variety of functions to communicate with the System z and z/OS, the mainframe networking has a wide range of licensed programs to simplify tasks and provide ease of management and administration. These products provide session management functions, network monitoring, performance management and diagnosis tools. There are products from IBM and third-party vendors that provide similar or exclusive functions.

1.7.1 NetView Performance Monitor

The IBM Tivoli® NetView® Performance Monitor (NPM) aids network support personnel in managing VTAM-based communications networks. It collects and reports on data on the host and NCP. NPM data can be used for the following purposes:

- ▶ Identify network traffic bottlenecks
- ▶ Display screens showing volume and response times for various resources
- ▶ Generate color graphs of real-time and historical data
- ▶ Alert users to response time threshold exceptions

NPM performance data can also help in several ways:

- ▶ Determine the performance characteristics of a network and its components
- ▶ Identify network performance problems
- ▶ Tune communications networks for better performance and verify the effects of problem resolutions
- ▶ Gauge unused capacity when planning for current network changes
- ▶ Produce timely and meaningful reports on network status for multiple levels of management.

1.7.2 OMEGAMON XE for Mainframe Networks

IBM Tivoli OMEGAMON® XE for Mainframe Networks collects network performance data across z/OS systems. The following list includes several other functions and benefits of Omegamon XE for Mainframe Networks:

- ▶ Immediately sense poor or unstable network connections that hamper application performance, and allows you to quickly pinpoint root causes to ensure high availability of services and improve user productivity
- ▶ Enables the management of multiple z/OS and network stacks from a single interface to improve user productivity and operational scalability
- ▶ Improves your ability to standardize event, incident, problem, and availability management across domains and platforms through extensive integration with Tivoli NetView for z/OS, Tivoli Enterprise Portal and the entire OMEGAMON suite, Tivoli Netcool/OMNIbus, and Tivoli System Automation for z/OS to achieve operational excellence
- ▶ Real-time reporting and alert monitoring on Enterprise Extender (EE) and High Performance Router (HPR) connections
- ▶ The first monitor for IP Security (IPSec) and capability to detect intrusions and identify the culprit
- ▶ Monitors OSA Adapter performance and detect IP congestion which might slow down transaction response time
- ▶ Monitors FTP job status and performance throughput with alert notification and automated actions to correct problems

1.7.3 Session Manager for z/OS

IBM Session Manager for z/OS allows access to multiple z/OS applications from a single 3270 terminal, helping users manage their workload more efficiently and productively. Users can switch between applications with a single key stroke or command. This secure single menu gives users access to applications running on any z/OS machine in the network. Session manager provides following features and functions:

- ▶ Provides secure, intuitive access to multiple z/OS systems from a single 3270 terminal.
- ▶ Session recovery provides greater resilience and flexibility after a planned or unplanned outage.
- ▶ Cost and effort of network administration is reduced with further benefits to help desk and operations personnel.
- ▶ Centralized user ID administration allows the ability to broadcast messages to users.

- ▶ Batch administration eases the potential administration overhead for mass updates for large sites.
- ▶ Dynamic menus allow users and applications to be administered using External Security Manager definitions.

1.7.4 Solve: Access Session Management

CA SOLVE:Access Session Management improves user productivity, reduces training costs and simplifies the process of logging client programs into centralized applications. It reduces the time and cost of network administration and enhances the support the Service Desk provides to users through centralized user ID administration, enhanced visibility of user problems and more. CA SOLVE:Access simplifies user access and network administration, and enhances security. In addition, the product integrates with CA NetMaster Network Management for TCP/IP, which helps you improve correlation between users and connections, and provides CA SOLVE:Access with IP addresses for display. It includes the following key features:

- ▶ Simplified mainframe application access
- ▶ Scripted logon to applications
- ▶ Reduced network administration time and cost
- ▶ Enhanced network security
- ▶ Improved customization and configuration
- ▶ Modern network infrastructure support

1.8 Operations and administration

In this section, we describe the tasks, roles, and responsibilities involved in managing and administering a mainframe network environment.

1.8.1 Operational tasks

The role of a z/OS network administrator can span a wide area. In some organizations you might be a generalist, looking after all z/OS networking components, printer subsystems, and even some of the hardware. However, it is more likely that you will specialize in one or two particular areas, such as VTAM and TCP/IP.

This chapter remains at a high level, and attempts to give you a general understanding of the role. Within z/OS networking components on the mainframe, these are some of the common tasks that you are expected to perform:

- ▶ Correct z/OS network-related faults.
- ▶ Change and configure network components.
- ▶ Monitor and control network components.
- ▶ Provide network performance and usage statistics.
- ▶ Work with other groups on projects, tasks and problems. These groups might include operations, the WAN network team, z/OS systems programmers, change control, business users, and testers.

“Monitoring” means watching and observing, normally to see something change. This does not imply that you sit there all day watching a screen. There are network management tools installed on z/OS to capture alert and monitor messages and events. There are also network operations staff members who normally are the first-level filters for problems.

1.8.2 z/OS network administrator tasks

Network administrator tasks can be varied and are, one way or another, derived from the needs of the organization within which the network administrator functions. In an environment other than z/OS, a network administrator might have a role that encompasses many platforms and hardware areas. However, z/OS network administrators tend to have a more narrow focus. One reason for this is complexity: network administration on z/OS requires a good working knowledge of z/OS itself. It also requires a good knowledge of networking hardware. When you combine this with the actual VTAM, TCP/IP, LAN, and WAN knowledge required, the z/OS network administrator is unlikely to have the opportunity to include other platforms.

So what are some of the tasks a z/OS network administrator might undertake? A sampling is described in Table 1-1.

Table 1-1 Network administrator tasks

Task	Description
Problem source identification	When something does not function as expected, the network administrator is one of the first persons to work towards resolution.
Network control	VTAM, TCP/IP, and their associated applications must be started, stopped, monitored, and maintained as required.
Planning	Planning includes network architecture decisions, such as what role z/OS plays in the network and how z/OS should be situated in the network.
Change control	In a z/OS network environment, all changes are part of a planned and controlled process. The z/OS network administrator would work with other network administrators.
Hardware evaluation	If a new feature is to be added to the z/OS host or to the network used by the z/OS host, then an evaluation of the impact of the feature must be assessed.
Software evaluation	In the TCP/IP world, new networking applications and updated existing ones are literally a daily phenomenon.
Installation of hardware and software	After the decision is made, the actual software or hardware must be implemented.
Capacity planning	Network capacity requirements are always changing.
Paperwork	Documentation of the environment, changes, and procedures are all part of the network administrator's role.

For problem determination, the network administrator should first determine the general cause of the problem by reading error messages, checking for system memory dumps, checking to see whether software or hardware has changed, and reading the system log. After determining the general cause of the problem, the network administrator should use the tools and diagnostic aids available to determine the specific cause of the problem. Lastly, tuning tasks should be carried out to ensure good network performance.

z/OS has diagnostic aids that the network administrator can use: abend dumps, stand-alone dumps, and supervisor call (SVC) dumps, which the Interactive Problem Control System can format for easier reading. VTAM also has specific aids, such as IBM First Failure Support Technology™, CSDUMPs, network traces, sense codes, VTAM traces, and commands that display the state of VTAM components and resources. TCP/IP has component traces and diagnostic commands (such as the `NETSTAT` command) that help determine problems in the IP network.

Communications Storage Manager (CSM) problems generally manifest as central storage problems. The network administrator can display CSM's use of storage, activate CSM VTAM traces, and dump CSM storage for analysis.

1.9 Securing mainframe networks

The security objectives in networking aim to achieve data integrity and privacy despite all the threats that networks are exposed to in today's computing environment. This would not be a complex challenge if communication between two applications were given a physically isolated and dedicated wire link for each conversation. However, physical networks share resources among numerous users. Furthermore, technologies such as TCP/IP and the Internet have been designed to make this sharing flexible and easy. Therefore, network security is an area that needs to be addressed.

With respect to the mainframe, the complexity of the hardware components, various protocols and advanced features pose more challenges in ensuring security and isolation, even within a single central processing complex that hosts multiple logical partitions.

From the hardware perspective, System z does have integrated hardware network adapters, and the counter-measures to the security threats are left to the software running in the system. However, there are other needs for security at the hardware level, because these adapters are also resources that must be sharable between logical partitions and even, in the case of VLANs, between networks. Another consideration regarding network security with System z is that, in some System z configurations, the TCP/IP protocol can be transported by IBM proprietary technologies and is less exposed to the classical networking threats.

Thus, as mentioned, data security and isolation demanded is much higher and complex on System z due to the virtualization it provides at various levels. Technologies such as HiperSockets or RoCE makes it evident to prove the customers that the data belonging to a particular transaction is secure, and which is traceable throughout its lifecycle. Also, with SNA not being popular as TCP/IP due to its restricted use with only mainframe applications, any incorrect configuration or practice might expose it to a serious security threat. As the technology now allows SNA over IP communication, such threats might propagate to a wider scope endangering a larger part of the business.

Therefore, there is a need to protect mainframe network resources appropriately to secure the applications that use the functions of System z and IBM Communications Server components and protocols.

In the remainder of this book, we focus on network security for the following integrated System z components:

- ▶ Chapter 2, "Cryptography for network security" on page 37
- ▶ Chapter 3, "TCP/IP security" on page 89
- ▶ Chapter 4, "SNA security" on page 131
- ▶ Chapter 5, "Shared Memory Communications over RDMA" on page 155

The mainframe, like any other IT component in a complex deployment, does not stand-alone. It is crucial for every organization to define and apply an enterprise-wide security architecture that regards the resources on a mainframe as part of the overall solution. Policies, access control, log file collections, and more must be consolidated at a centralized security office. To further investigate this integration, read the related IBM Redbooks publication titled *Security on the IBM Mainframe: Volume 1 A Holistic Approach to Reduce Risk and Improve Security*, SG24-7803.



Cryptography for network security

In this chapter, we provide an overview of security concepts and architecture for network cryptography on IBM z Systems. Then, we describe guiding principles for configuring network cryptography.

This chapter includes the following sections:

- ▶ 2.1, “Security concepts and architecture for network cryptography on System z” on page 38
- ▶ 2.2, “Guiding principles for cryptography for network security” on page 68

2.1 Security concepts and architecture for network cryptography on System z

This section describes security concepts and architecture for network cryptography on System z for the following topics:

- ▶ Basics of cryptography for network security
- ▶ Definition of a secure communication model for networks
- ▶ Applications of cryptosystems for network security
- ▶ Overview of the z/OS TCP/IP cryptographic infrastructure
- ▶ Transport Layer Security on z/OS
- ▶ AT-TLS
- ▶ IPSec
- ▶ OpenSSH on z/OS
- ▶ PKI services

2.1.1 Basics of cryptography for network security

Cryptography is a widely used and often referenced term. It applies to many areas in information technology. For example, cryptography is the only manageable way to protect valuable data on modern disk storage systems from unauthorized use. Or think about credit cards and the way the personal information is stored on them. To protect this information, cryptography is used.

The same principles apply to network security. You have to use systems that use cryptographic algorithms to provide proper protection of data that is transmitted over the network. Review the four basic principles of cryptography with a focus on network security:

- ▶ Confidentiality

Confidentiality is the most prominent idea associated with cryptography; that is, data is scrambled using a known algorithm and secret keys such that the intended party can descramble the data but an attacker cannot.

- ▶ Authentication

Authentication is the process of proving your identity to your communication partner.

- ▶ Integrity

Integrity checking ensures that what we receive was what you sent, and vice versa (for example, that no one has altered a transmission, or that the decimal point in a transferred number *is* exactly where it is supposed to be).

- ▶ Non-repudiation

Non-repudiation ensures that I know *you* agreed to what was exchanged, and not someone masquerading as you. Non-repudiation implies a legal liability. I know you and only you agreed to the matter at hand and, therefore, you are legally and contractually obligated. This is the same as a signature on a contract.

Therefore, you could say that network security using appropriate cryptography can ensure the following principles of communication in a network. If two or more partners are communicating over the network they can be certain of the identity of the other parties. The participants in the conversation can be sure that data sent over the network is not forged or modified in any way. They can also be sure that no one else but their partners are able to read what has been sent. And finally, no participant can deny having sent something that has been received by the other participants.

Of course, no cryptosystem is unbreakable. No cryptosystem offers a 100% safety to deliberate attacks from third parties. The more effort you make to implement cryptography the right way, the safer you can be. But no effort guarantees complete safety.

2.1.2 Definition of a secure communication model for networks

All secure communications in a network follow a set of common concepts, regardless of what cryptosystem is used to establish a secure communication. Imagine two partners that want to start a secure communication over a generally unsecure network. Figure 2-1 illustrates the components and concepts that are involved in such a communication.

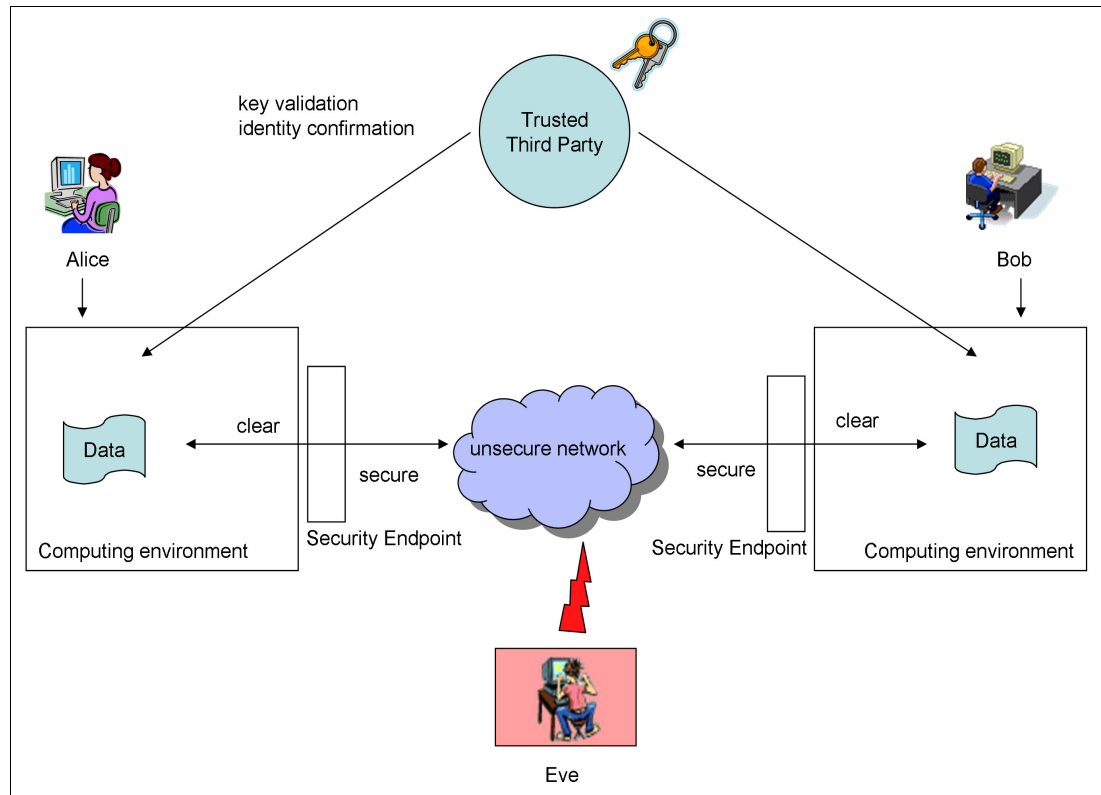


Figure 2-1 A secure communication model for networks

When two participants want to communicate securely over the network by using cryptographic protection, a *trusted third party* is often used to establish trust between the participants and to assist in key distribution. This trusted third party is a necessary element for secure communication to receive and exchange encryption keys and for authentication purposes. Consider, for example, the concept of Public Key Infrastructure (PKI). The certificate authority (CA) as a trusted third party is essential in a PKI. The CA is a globally trusted third party within the PKI. Authentication works only on the basis of trustworthiness of the CA. For a detailed description of the concept of PKI, see “Public Key Infrastructure” on page 41. When both parties share that they trust the third party, they may also trust each other, because the third party offers services to confirm the identity of the respective partners.

To cryptographically protect a message as it traverses a network, the clear text message has to be encrypted into ciphertext before sent over the network. This is accomplished using a secure session based on well-known cryptographic algorithms that are negotiated between the communication partners.

These algorithms assure the authenticity, integrity, and privacy of the message. The partners also have to agree on the key values to use. This process is called key exchange and usually involves help from the trusted third party.

In 2.2.2, “Defining a cryptography strategy within your organization” on page 73 we describe in more detail where security endpoints are placed within an enterprise network and how to connect to external networks.

Another component of this communication model is the attacker (Eve). In a world with no attackers trying to gather important information and misusing it, no cryptography and network security is needed. But this is no longer the case. The *imaginary enemy with unlimited time, skills, and resources* must be considered by everyone every time communication security is evaluated.

2.1.3 Applications of cryptosystems for network security

The secure transmission of data over the network demands cryptographic systems. We now provide a summary of common cryptographic systems to achieve secure communication by the standards of the four basic principles of security.

Key management

An important basis for the proper operation of a cryptosystem is key management. Secret keys must never be compromised or made available to anyone outside the cryptosystem. This is true for all keys used in cryptosystems making use of symmetric algorithms. For cryptosystems that make use of asymmetric algorithms, no one but the owner of a private key is ever allowed to possess it. If these conditions fail, the cryptosystem is compromised and is no longer capable of assuring data protection. The protection of keys is vital to the security of a cryptosystem.

Key management in a closed environment

In high-security environments like the System z mainframe, key management can be implemented using cryptographic hardware that is installed and managed by a centralized security facility. Master Keys and key-exchange keys can be installed centrally, and the hardware facility can be delivered to the users with the necessary keys installed. In z/OS, these tasks are accomplished by the Integrated Cryptographic Service Facility (ICSF) for software key management, CryptoExpress cards for key storage, and the Trusted Key Entry for secure key entry and retrieval. For a more detailed introduction to these key management capabilities of z/OS see section “9.12 Integrated Cryptographic Service Facility” in the IBM Redbooks publication titled *Security on the IBM Mainframe: Volume 1 A Holistic Approach to Reduce Risk and Improve Security*, SG24-7803.

Cryptosystems for data privacy and authentication

Encrypting and decrypting large amounts of data with cryptosystems making use of asymmetric algorithms is expensive (in reference to time and resources). Therefore, symmetric algorithms, such as AES and Triple DES, are used for bulk data encryption. The disadvantage of symmetric algorithms, however, is that both partners (the party that encrypts the data and the party that decrypts the data) must be in possession of the same key. Key management or safe distribution of keys in insecure networks is a problem with symmetric cryptosystems, even more so because data encryption keys need to be changed frequently to make an adversary's task more difficult and limit the potential damage if a key is compromised. Therefore, today mostly hybrid cryptosystems are in place to overcome the shortcomings of both approaches. One example for such a hybrid system is the *public key infrastructure*, described in the following section.

Public Key Infrastructure

Public key cryptosystems can be used to transmit keys used to encrypt data to a recipient over the network. Data that has been encrypted with the public key of the recipient can only be decrypted using the recipient's private key. If someone (the *recipient*) makes the public key publicly known, anyone (a *sender*) can encrypt a message that only that person can decrypt, for example:

1. A sender can generate some random secret from which a symmetric key can be derived.
2. The sender encrypts that secret using the recipient's public key.
3. The sender sends the encrypted secret to the recipient.
4. The recipient decrypts the secret using the private key.
5. The recipient derives the symmetric key from the secret using an agreed-to method.
6. After both sides have the symmetric key, they can use it to encrypt or decrypt larger messages.

This method works well and has reasonable performance because asymmetric algorithms are used to encrypt and decrypt only small amounts of data.

The problem with this method arises from the question: How can someone publish a public key in a secure manner? If a malicious user sends a public key to Bob claiming it's Alice's key, how can Bob be sure that the key really belongs to Alice? In this situation, Digital Certificates and a Public Key Infrastructure (PKI) can help. Recall the secure communication model described in 2.1.2, "Definition of a secure communication model for networks" on page 39. There, we stated that secure communication over the network usually requires a trusted third party to confirm the identities of the participants in the conversation. A PKI will do exactly this job in a secure manner. A PKI is a hierarchy of authorities that issue certificates and attest to their authenticity,

Public key infrastructures provide a way to ensure the authenticity of a certain identity which possesses a private key. Authentication is the second cryptographic principle that PKIs can be used for.

To achieve authentication, PKI introduces the concept of a trust chain or trusted authority. These trusted parties are called certificate authorities (CA). A CA is a trusted third party within the PKI. The private key of a CA is generally trusted within the PKI. A CA is used to sign certificates issued to entities (users, servers, and so on) using its private key. The entity receives a signed private and public key pair from the CA. The corresponding public key is published within the PKI. Because the CA is trusted and because everybody can verify a message signed with a private key using the corresponding public key, anybody who trusts the CA can be sure that the private key which has signed the message belongs to the entity which has requested the signing from the CA. The CA itself will make sure by any means that the receiver of a private/public key pair is not forging its identity. The overall concept of this method is trust. If the CA is compromised, then the PKI that uses this CA is also compromised. The trust chain is broken. Therefore, CAs have to be safeguarded at all times. There have been cases in the media, where globally trusted CAs have been compromised which had an enormous impact on secure communication over the internet.

With PKI Services, z/OS offers a powerful PKI infrastructure. PKI Services is part of the Security Server for z/OS and shipped as a base component. It uses z/OS concepts for availability and outstanding cryptographic abilities. We describe PKI Services further later in this chapter.

Kerberos

Kerberos is a network authentication protocol that was developed in the 1980s by Massachusetts Institute of Technology. Kerberos can use a variety of symmetric encryption algorithms, including AES and Triple DES, to provide data privacy, especially for the sensitive data such as password to log in to a server. Kerberos Version 5 is the latest release. Kerberos is implemented as a component of the Network Authentication Service in z/OS, and chosen by Microsoft Corporation as their preferred authentication technology in Windows 2000 and after, and by SUN for Solaris 8. Kerberos is an encryption-based security system that provides mutual authentication between the users and the servers in a network environment.

The Kerberos authentication is heavily based on shared secrets, which are passwords stored on the Kerberos server. Those passwords are encrypted with a symmetrical 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 uses public key cryptography. Therefore, the servers must be placed in locked rooms with physical security to prevent an attacker from stealing a password.

For a complete description about the Kerberos Version 5 protocol, see *RFC 1510 - The Kerberos Network Authentication Service (V5)*.

Though it is a formally accepted secure protocol, its support in software is only limited in contrast to others. We do not cover Kerberos in further detail in this book.

Cryptosystems for data integrity

Data integrity checking is the ability to assert that the data that is received over a communication link is identical to the data that is sent. Data can be compromised not only by an attacker, but it can also be damaged by transmission errors (although these are normally handled by the transmission protocols). Although ensuring data integrity over a network requires the use of cryptographic algorithms, these algorithms do not prevent others from viewing the data as it traverses the network as is the case with data privacy.

Message authentication codes

Data integrity in transmission of a message can be ensured using message authentication codes (MAC). A MAC can either be based on a symmetric algorithm or on a cryptographic hash function.

For symmetric algorithms, using a variation of the AES algorithm and a secret key, a MAC is created from the data. The MAC is sent with the message. The receiver performs the same operation using the same key and compares the resulting MAC with the MAC that was sent with the data. If both match, the integrity of the data is assured. MACs rely on the same secret key that is used by both the sender (to create the MAC) and the receiver (to verify the MAC). Because the MAC is derived from a secret key known only to the sender and receiver the MAC can be sent in the clear. An adversary sitting between the sender and the receiver (a so-called “man-in-the-middle” attack) can alter the message but cannot forge the MAC because the key to create the MAC is unknown. The mathematical principle behind using the MAC is that finding a message that fits a certain MAC is as difficult as breaking DES encryption.

A disadvantage to this method is that, as in symmetric cryptosystems, secret keys must be shared by sender and receiver. Furthermore, because the receiver has the key that is used in MAC creation, it is difficult to make it impossible for the receiver to forge a message and claim it was sent by the sender.

Constructing MACs from hash algorithms is a different approach to data integrity. Hash algorithms digest (condense) a block of data into a shorter string (usually 128 or 160 bits). A MAC constructed from a hashing algorithm is called Hash-bases MAC (HMAC).

The principles behind hash algorithms are that the message cannot be recovered from the message digest and that it is hard to construct a block of data that has the same message digest as another given block.

The following list includes some of the common message-digesting algorithms:

SHA-2	SHA-2 is an improved algorithm and generates a 256, 384, or 512-byte hash value. SHA-2 is considered to generate message digest values that are less likely to yield collisions.
MDC-4	The MDC-4 algorithm calculation is a one-way cryptographic function that is used to compute the hash pattern of a key part. MDC uses encryption only, and the default key is 5252 5252 5252 5252 2525 2525 2525. It is used by the TKE.
MD2	This algorithm was developed by Ron Rivest of RSA Data Security, Inc.®. The algorithm is used mostly for PEM certificates. MD2 is fully described in RFC 1319. Because weaknesses have been discovered in MD2, its use is discouraged.
MD5	This algorithm was developed in 1991 by Ron Rivest. The algorithm takes a message of arbitrary length as input and produces as output a 128-bit message digest of the input. The MD5 message digest algorithm is specified by RFC 1321, The MD5 Message-Digest Algorithm. Because weaknesses have been discovered in MD5, its use is discouraged
SHA-1 SHA-1	This algorithm was developed by the US Government. The algorithm takes a message of arbitrary length as input and produces as output a 160-bit hash of the input. SHA-1 is fully described in standard FIPS 180-1.

To construct an HMAC sender of a message (block of data) uses an algorithm (for example SHA-1) to create a message digest from the message. Some key, to which sender and receiver must agree before the creation of the HMAC is appended to the data. The message digest is sent together with the message. The receiver runs the same algorithm over the message using the secret key and compares the resulting message digest to the one sent with the message. If both match, the message is unchanged. Considering attacking schemes and the problems of key management, the same principles hold for HMACs as for MACs.

Cryptosystems for data non-repudiation

Data non-repudiation is crucial to modern ways of doing business over networks and also signing and verifying contracts closed electronically. Any piece of data sent via networks that is sensitive to contract or legal issues or just basically seen as important for you or your customers is in need of a system that denies the refusal of ownership to that data. These systems protect the sender, because he can prove his ownership and they protect the receiver because he can rely on the non-deniability of the authorship the sender.

Digital Signatures

Digital signatures are an extension of data integrity. Although data integrity only ensures that the data received is identical to the data that is data sent, digital signatures provide data non-repudiation. For the purpose of authentication in secure communications, digital signatures provide identification of the communicating parties.

The creator of a message or electronic document that is to be signed uses a hashing algorithm, such as SHA-1 or SHA-2, to create a message digest from the data. The message digest and some information that identifies the sender are then encrypted with the sender's private key. This encrypted information is sent together with the data. The receiver uses the sender's public key to decrypt the message digest and sender's identification. The receiver then uses the message digesting algorithm to compute the message digest from the data. If this message digest is identical to the one recovered after decrypting the digital signature, the message is authentic, and the signature is recognized as valid.

With digital signatures, only public-key encryption can be used. If symmetric cryptosystems are used to encrypt the signature, it is difficult to make sure that the receiver (having the key to decrypt the signature) could not misuse this key to forge a signature of the sender. If the private key of the sender is well-protected (kept secret), no one else can forge the sender's signature.

There are significant differences 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 a private key. Therefore, everybody can send encrypted data to the receiver that only the receiver can decrypt.
- ▶ With digital signatures, the sender uses the private key to encrypt the signature, and the receiver decrypts the signature with the sender's public key. Therefore, only the sender can encrypt the signature, but anyone who receives the signature can decrypt and verify it. The tricky thing with digital signatures is the trustworthy distribution of public keys.

Public Key Infrastructure can provide functions for the proper distribution and management of keys used for digital signatures in an enterprise. See "Public Key Infrastructure" on page 41 for more information about PKI.

2.1.4 Overview of the z/OS TCP/IP cryptographic infrastructure

Cryptography is, as stated before, a widely used concept throughout information technology. Let us take a closer look on the overall architecture for cryptography for network security on the mainframe. z/OS offers a wide range of products and protocols to secure the communication within a network. This architecture is shown in Figure 2-2 on page 45.

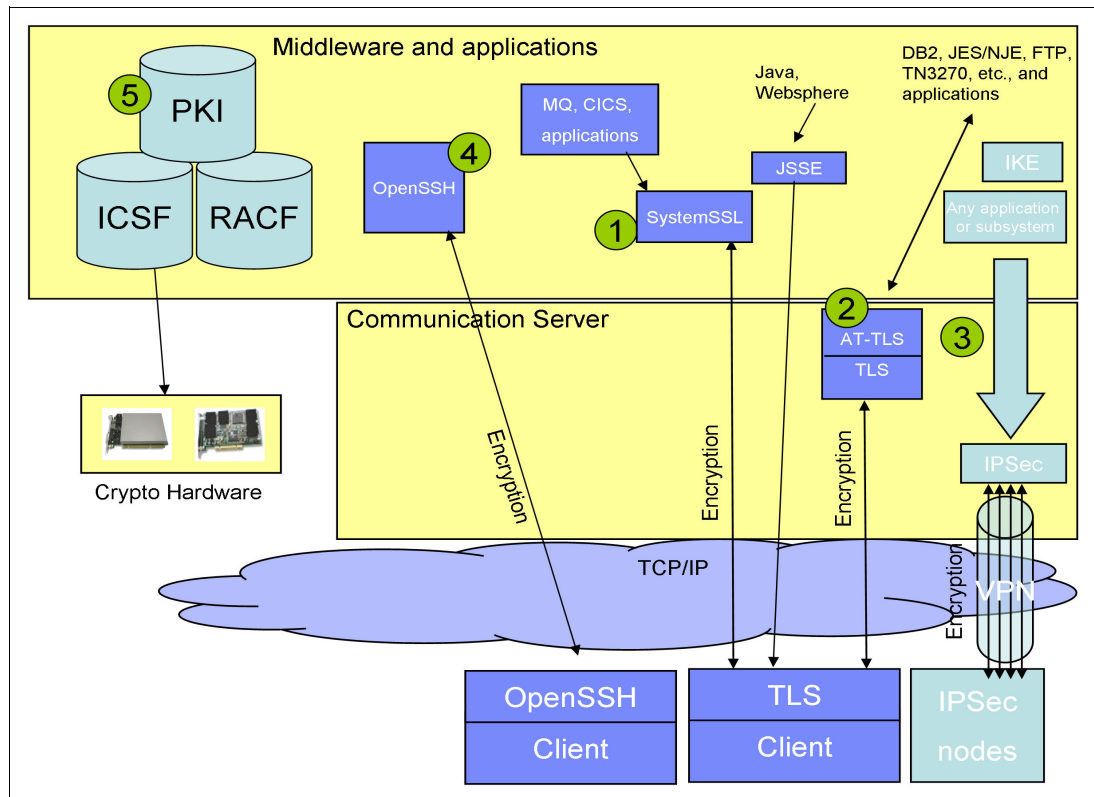


Figure 2-2 z/OS TCP/IP cryptographic architecture for applications

In the next sections of this chapter, we describe the following components of z/OS Communications Server, which enable an organization to secure network communication using cryptography:

- ▶ **Transport Layer Security (TLS)**
Securing TCP data sent over the network, providing data privacy, data integrity and authentication (to some extent), explained in 2.1.5, “Transport Layer Security on z/OS” on page 46.
- ▶ **Application Transparent TLS (AT-TLS)**
A z/OS Communications Server feature that protects application traffic using System SSL based on policy rather than requiring source code changes in the application. AT-TLS policy is provided through the Policy Agent as explained in 2.1.6, “AT-TLS” on page 51.
- ▶ **Virtual private networks using IP Security (IPSec) and Internet Key Exchange (IKE)**
Securing any kind of IP traffic (TCP, UDP, ICMP, and so on) using virtual private networks as explained in 2.1.7, “IPSec” on page 54.
- ▶ **OpenSSH for z/OS**
Shipped as a part of the Ported Tools for z/OS. Secure access to UNIX System Services providing data privacy and integrity, explained in 2.1.8, “OpenSSH on z/OS” on page 60.
- ▶ **PKI Services z/OS**
The z/OS implementation of a Public Key Infrastructure. This component provides a reliable, scalable and secure PKI, including a full-function certificate authority, leveraging the System z cryptographic infrastructure, explained in 2.1.9, “PKI services” on page 65.

2.1.5 Transport Layer Security on z/OS

z/OS uses its own implementation of the Transport Layer Security and Secure Sockets Layer family of protocols. Let us provide a short introduction to TLS and the features that z/OS System SSL offers.

TLS operation in general

Transport Layer Security (TLS) is an IETF-standardized client/server communication protocol based on the Secure Sockets Layer protocol which was developed by Netscape for securing communications that use TCP/IP sockets. It uses asymmetric and symmetric key cryptography for these purposes:

- ▶ For server authentication using digital certificates
- ▶ To provide data privacy and integrity using symmetric encryption and MACs (mostly SHA-1 or MD5 today)
- ▶ For optional client authentication using digital certificates

The TLS protocol is based on Netscape's SSL version 3 protocol. Given the success and popularity of SSL for securing HTTP traffic, the Internet Engineering Task Force (IETF) adopted SSLv3 with some minor modifications and standardized it in RFC 2246 as TLS version 1.0. As such, SSLv3 was the last version of SSL proper -- all new development of the protocol family is done under the TLS protocol (the latest version as of this writing is TLSv1.2). When a client and server negotiate a TLS/SSL session, one of the things they must agree on is the protocol version that they will use, which might be anywhere between SSLv2 (not recommended) up through TLSv1.2. For our purposes, we will use the terms TLS and SSL interchangeably, but we will prefer the term *TLS* to refer to the general protocol family.

TLS services are typically invoked by the application program and therefore an application has to be designed to support TLS to benefit from the TLS protection. It is also interesting to notice that, although TLS is well-known as a way to secure HTTP communications, any kind of TCP socket communication is a candidate for TLS protection, with proper code support. For instance, the z/OS TN3270 server and the FTP and LDAP servers and clients are TLS-enabled. This increases the TLS protocol popularity.

There are two main peer authentication options under TLS. With server authentication, the server proves its identity to the client by passing its X.509 certificate to the client during the TLS handshake. With client authentication or mutual authentication, the TLS client also proves its identity by passing its certificate to the server after it has authenticated the server's identity.

Note that the TLS client and server must have the appropriate certificate authority certificates on their key rings to verify the peer certificate. In addition to verifying the validity of a certificate, TLS can also ensure that the peer certificate has not been revoked by checking a Certificate Revocation List (CRL) retrieved from an LDAP directory entry managed by the client's peer's certificate authority.

TLS consists of two major phases. The first phase is known as the handshake phase, under which the two communication partners establish a secure session with each other, and the second one is known as the data transmission phase, under which application traffic is cryptographically protected using the secure session.

The following list gives an overview of the steps necessary to perform a TLS handshake:

1. A TCP connection is established between with the client and the server.
2. The client indicates that it wants to perform TLS operation with the server by sending a *client hello* message to the server. This message also advertises the protocol and ciphers that the client is willing to use.
3. The server agrees upon TLS conversation and sends back an acknowledgment to the client called a *server hello* message. The server hello message indicates which protocol and cipher suite the server has chosen and it also contains the server's certificate (and, optionally a request for the client's certificate).
4. The client verifies the digital certificate sent by the server and optionally sends its own certificate to the server for client authentication.
5. The client and server then exchange messages to establish a set of secret keys that will be used to protect the application data according to the agreed-upon cipher suite. These messages are encrypted, at first with the server's public key, and later with the agreed-upon session keys. After this exchange is complete, the secure session is ready to protect application data.
6. Data transmission phase begins. In this phase, the application data is packaged within *TLS records* which are cryptographically protected using the negotiated ciphers and keys.

TLS is known to be a demanding protocol in terms of computing resources in the handshake phase. These resources are needed to support the heavy calculation required by the asymmetric cryptography that the handshake uses during this phase. The data transmission phase, on the other hand, does not use asymmetric cryptography. Instead, it uses more efficient symmetric algorithms to protect application data (which can be quite large in some cases). Even though these algorithms are less expensive than those used during the handshake phase, they are still computationally intensive. For this reason, the data transmission phase makes heavy use of the System z CP Assist for Cryptographic Function (CPACF) hardware facility, which implements most of the symmetric algorithms within the System z instruction set.

TLS implementation on z/OS

There are two implementations of SSL/TLS family of products available on z/OS. System SSL providing SSL/TLS functions as part of z/OS Cryptographic Services for C/C++ applications and Java Secure Sockets Extension (JSSE), a pure Java implementation of SSL/TLS.

System SSL

System SSL provides a complete TLS/SSL implementation for C/C++ applications on z/OS. System SSL provides a library with a full set of APIs that TCP sockets applications can use to protect their network traffic. Hence these applications only have to properly call the System SSL API, as opposed to having been designed with complete SSL support embedded. These applications can act either as an SSL client or an SSL server, and should call the API functions accordingly. System SSL supports the SSL V2.0, SSL V3.0 and TLS (Transport Layer Security) V1.0, V1.1 and TLS V1.2 protocols. TLS V1.2 is the latest version of the secure sockets layer protocol, it is only supported with z/OS 2.1.

To negotiate TLS sessions, the TLS implementation (System SSL or JSSE) must have access to the necessary certificates and private keys. System SSL supports three different key and certificate stores:

- ▶ RACF key ring
- ▶ ICSF PKCS#11 token
- ▶ An HFS file built and managed by the gskkyman utility

Figure 2-3 is a schematic view of the z/OS System SSL implementation. The SSL-enabled server, on the right side, invokes the System SSL API in the center of the picture. After it has established the TCP connection with the client; that is, after the TCP/IP accept socket function is unblocked by the client's response.

Note that SSL-enabled applications usually have a set of directives related to the SSL environment, which are set by the user and are transferred to System SSL via the API. For instance, when setting up the z/OS HTTP server to use SSL, a directive in the server configuration file specifies the location of the keys and certificates to be used by the server:

```
KeyFile racf_keyring_name SAF
```

When protecting communication with TLS, the z/OS HTTP server invokes a variety of System SSL APIs. One of those APIs is the `gsk_environment_open()` function, which takes the key ring value as an input parameter.

Upon invocation via the API, the System SSL code encapsulates the application data in cryptographically protected TLS records, which are then sent by the application via regular TCP/IP sockets functions. In the same way the application receives inbound messages, which are used by the System SSL code. System SSL reads TLS records built by the client from the socket, decapsulates them (including decryption) and then passes the cleartext data to the application.

The diagram in Figure 2-3 is not showing the sessionID cache that is maintained in the application address space (unless the sysplex sessionID caching is in use, as explained later).

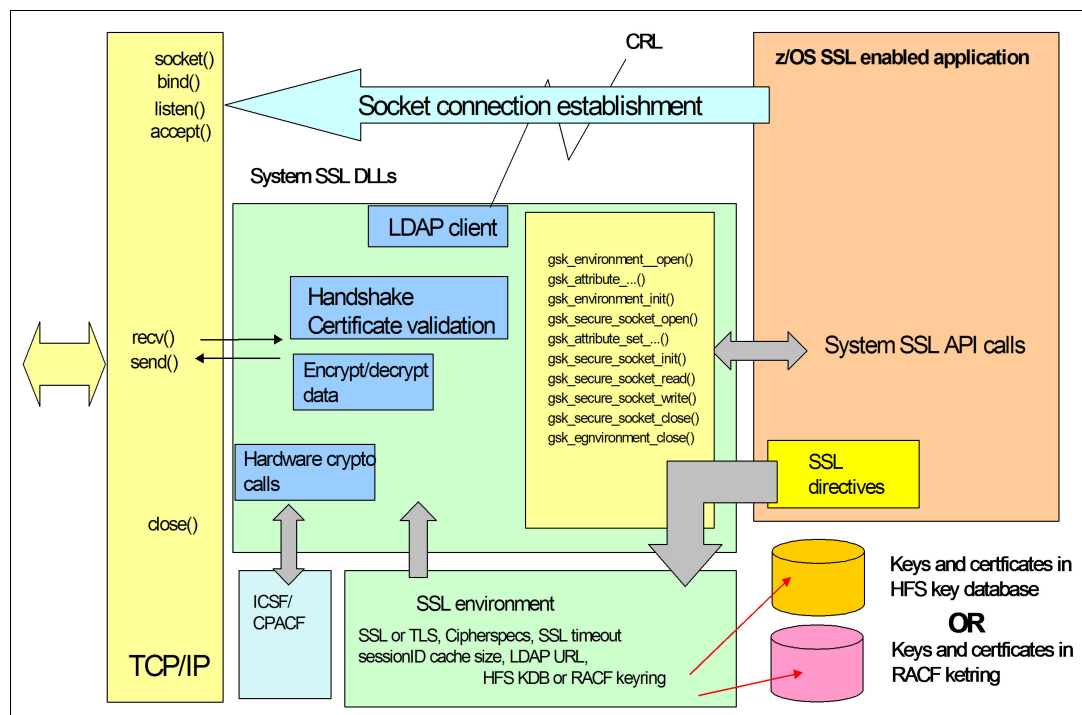


Figure 2-3 System SSL implementation

System SSL includes software implementation of commonly used cryptographic algorithms. As mentioned previously, some of the TLS operations can be fairly computing intensive and therefore System SSL will make use of the cryptographic hardware in System z whenever possible. If the hardware cryptography is not enabled, or if the selected algorithms are not supported by hardware, System SSL performs the cryptographic algorithms with its own software encryption routines.

Java Secure Socket Extension

The Java Secure Socket Extension (JSSE) enables secure Internet communications using SSL/TLS for Java applications. It provides a framework and an implementation for a Java version of the SSL and TLS protocols and includes functions for data encryption, server authentication, message integrity, and optional client authentication. Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol, such as Hypertext Transfer Protocol (HTTP), Telnet, or FTP, over TCP/IP. JSSE was previously an optional package (standard extension) to the Java 2 SDK, Standard Edition (SDK) versions 1.2 and 1.3. JSSE was integrated into the SDK, v 1.4.

The JSSE API is capable of supporting SSL versions 2.0 and 3.0, Transport Layer Security (TLS) 1.0, and from service refresh 10, TLS 1.1 and 1.2. The IBMJSSE2 implementation in the SDK implements SSL 3.0, TLS 1.0, and from service refresh 10, TLS 1.1 and 1.2. It does not implement SSL 2.0.

JSSE includes the following important features:

- ▶ Included as a standard component of JRE 1.4 and later
- ▶ Extensible, provider-based architecture
- ▶ Implemented in 100% Pure Java, so workload can be offloaded to zAAP coprocessors.
- ▶ Provides API support for SSL versions 2.0 and 3.0 and an implementation of SSL version 3.0
- ▶ Provides API support and an implementation for TLS versions 1.0, 1.1 and 1.2
- ▶ Includes classes that can be instantiated to create secure channels (SSLSocket, SSLServerSocket, and SSLEngine)
- ▶ Provides support for several cryptographic algorithms commonly used in cipher suites

The TLS/SSL information in this book focuses on System SSL implementation of SSL/TLS, rather than JSSE.

FIPS 140-2 considerations

National Institute of Standards and Technology (NIST) is the US federal technology agency that works with industry to develop and apply technology, measurements, and standards. One of the standards published by NIST is the Federal Information Processing Standard Security Requirements for Cryptographic Modules referred to as 'FIPS 140-2'. FIPS 140-2 specifies a set of requirements that a cryptographic module must meet to be considered secure and reliable in the way it preserves the integrity of its cryptographic algorithms and the way it protects the cryptographic keys it is using.

System SSL provides an execution mode that has been designed to meet the NIST FIPS 140-2 Level 1 criteria. To this end, System SSL can run in either FIPS mode or non-FIPS mode. System SSL by default runs in non-FIPS mode. To meet the FIPS 140-2 Level 1 criteria, System SSL, when executing in FIPS mode, is more restrictive with respect to cryptographic algorithms, protocols and key sizes that can be supported.

Table 2-1 shows the cryptographic algorithms available with System SSL when executing in FIPS mode.

Table 2-1 System SSL FIPS and non-FIPS mode key lengths in bits

	non-FIPS		FIPS	
Algorithm	Size	Hardware	Size	Hardware
RC2	40 and 128			
RC4	40 and 128			
DES	56	X		
TDES	168	X	168	X
AES	128 and 256	X	128 and 256	X
MD5	40			
SHA-1	160	X	160	X
SHA-2	224, 256, 384 and 512	X	224, 256, 384 and 512	X
RSA	512-4096	X	1024-4096	X
DSA	512-1024		1024	
Diffie-Hellman	512-2048		2048	

Use of cryptographic hardware with System SSL

System SSL uses the cryptographic infrastructure in System z through the z/OS ICSF component. ICSF provides a complete set of cryptographic primitives and access to the System z hardware cryptographic. If ICSF is available, then System SSL calls it to perform many of its cryptographic operations, many of which directly use available hardware features. If the relevant hardware features are not available, then ICSF will perform the functions in software. There is no need for additional manual configuration. This is an automated process performed by System SSL itself.

Figure 2-4 on page 51 illustrates when and how cryptographic coprocessors on System z are used for System SSL processing. This picture is also valid for AT-TLS, which is discussed in the next section 2.1.6, “AT-TLS” on page 51.

Crypto Type	Algorithm	CPACF only	CPACF + Crypto Express card
Asymmetric Encrypt/Decrypt	RSA signature generation	In software	In coprocessor (non-FIPS) or CEX4P (FIPS or non-FIPS), else in software.
	RSA signature verification	In software	In coprocessor or accelerator, else in software
	RSA encrypt for handshake	In software	In coprocessor or accelerator, else in software
	RSA decrypt for handshake	In software	In coprocessor, accelerator or CEX4P
	ECDSA signature generation	In software	In coprocessor on z10, z196/z114, zEC12 or CEX4P, else in software
	ECDSA signature verification	In software	In software
Symmetric Encrypt/Decrypt	DES	CPACF (non-FIPS mode only: DES not allowed in FIPS mode)	
	3DES	CPACF	
	AES-CBC-128	CPACF	
	AES-CBC-256	CPACF on z10, z196/z114, zEC12, in software on z9	
	AES-GCM-128, AES-GCM-256	CPACF on z196/z114, zEC12, in software on z9, z10	
Symm Auth	MD5	In software (non-FIPS mode only: MD5 not allowed in FIPS mode)	
	SHA-1	CPACF	
	SHA-224, SHA-256	CPACF	
	SHA-384, SHA-512	CPACF on z10, z196/z114, zEC12, in software on z9	

Figure 2-4 Exploitation of the cryptographic coprocessor in System z in System SSL

For more information about appropriate cryptographic ciphers and what to keep in mind when choosing a cipher, see 2.2.1, “Choosing appropriate cryptographic algorithms for network security” on page 69.

Recommended reading: For more information about System SSL, see *z/OS Cryptographic Services System SSL Programming*, SC14-7495-00

2.1.6 AT-TLS

As we discussed earlier, TLS-enablement of an application program typically requires some fairly extensive programming changes. To make TLS-enablement less costly and more available to TCP applications, z/OS offers a unique feature called Application Transparent Transport Layer Security (AT-TLS). With AT-TLS you can now deploy TLS encryption without the time and expense of re-coding your applications.

AT-TLS invokes SystemSSL on behalf of existing clear-text socket applications without requiring any application changes. Therefore, the term *application transparent* applies.

AT-TLS uses policy-based networking in z/OS Communication Server using the Policy Agent. We have discussed principles of policy-based networking in z/OS throughout Chapter 3, “TCP/IP security” on page 89.

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.

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.

Figure 2-5 on page 52 illustrates the operation of AT-TLS if z/OS operates as the server and some remote client is initiating the connection. If a matching AT-TLS rule is found, then a TLS or SSL session will be set up to protect the connection as specified by the action associated with that rule. Figure 2-5 illustrates the flow of AT-TLS operation with z/OS acting as the server of a TCP socket application.

Be aware: AT-TLS only supports TCP-based applications. It cannot be used to provide security for other transport layer protocol (like UDP) based applications. To provide security independent of the transport layer protocol, consider taking advantage of IP Security (IPSec) support.

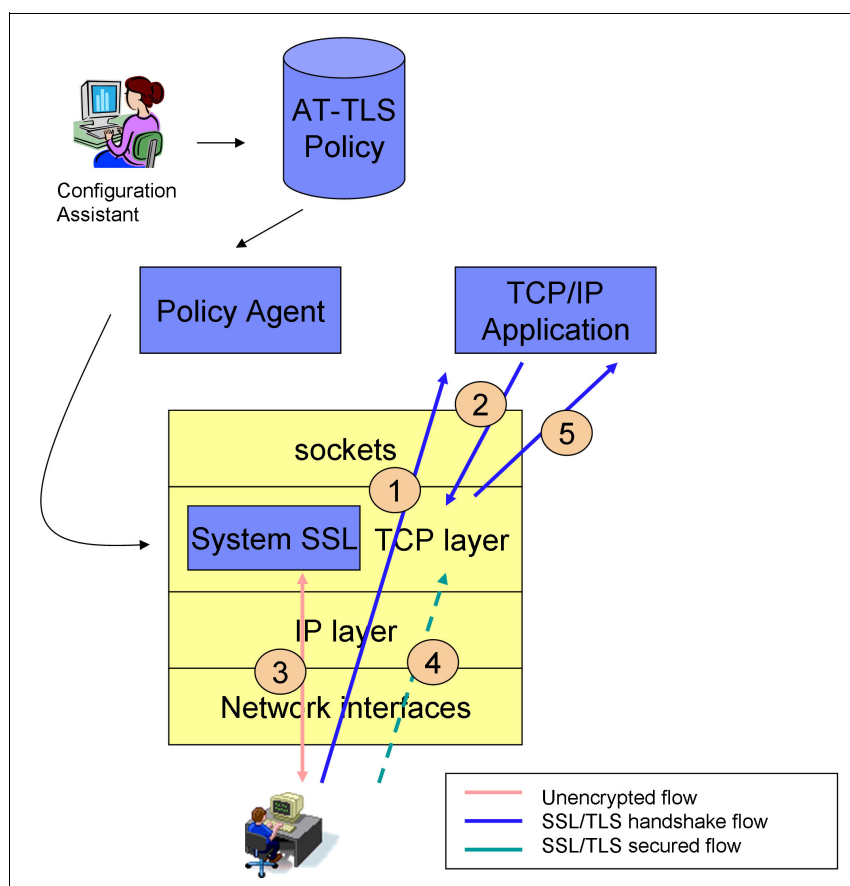


Figure 2-5 AT-TLS operation with z/OS as a server

This diagram illustrates the following flow:

1. The client connection to the server gets established in the clear (no security, TCP handshake only).
2. After accepting the new connection, the server issues a read request on the socket. The TCP layer checks AT-TLS policy and sees that AT-TLS protection is configured for this connection. As such, it prepares for the client initiated SSL handshake.
3. The client initiates the SSL handshake and the TCP layer invokes System SSL to perform the handshake under the authority of the server.
4. Client sends data traffic under protection of the new SSL session.
5. TCP layer invokes System SSL to decrypt the data and then delivers the cleartext inbound data to the server.

All of this is performed transparently to the remote application, because it has no way of determining that the handshake and encryption are being done through AT-TLS rather than through direct System SSL calls. Because AT-TLS simply uses System SSL, almost all of the System SSL features and capabilities are available, including the latest TLS versions and cipher suites

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 more control over the negotiation of TLS or need to participate in client authentication. However, these applications must be aware of AT-TLS.

Though we have been focusing on the transparent nature of AT-TLS so far, there are some applications that want to be aware of the TLS/SSL sessions being used to protect their network traffic and others that actually want to actively participate in the decision to use TLS/SSL. As such, IOCTL support is provided for applications that need to be aware of AT-TLS for status or to control the negotiation of security.

AT-TLS application types

Applications have different requirements concerning security. Certain applications need to be aware of when a secure connection is being used. Others might need to assume control if and when a TLS handshake occurs. For this reason, there are different application types supported by AT-TLS. These include the following application types:

- ▶ Not-enabled applications:
 - Pascal API and web Fast Response Cache Accelerator (FRCA) applications are not supported at all by AT-TLS.
 - By default, when there are no AT-TLS policies in place and an application is considered to be not-AT-TLS enabled. This includes applications that start during the InitStack window and if the policy explicitly says `enabled off`.
 - A third category applies to FTP and Telnet. They can be not-enabled for AT-TLS in the Policy Agent, but function with their native TLS/SSL capabilities independent of AT-TLS.
- ▶ Basic applications:
 - The AT-TLS policy says `enabled on`.
 - The application is unchanged and unaware of AT-TLS (no AT-TLS IOCTL calls).
- ▶ Aware applications
 - The application is changed to use the `SIOCTTLSCTL` IOCTL to extract TLS session information such as the peer's X.509 certificate or the cipher suite being used. These applications typically want to access the TLS information for additional authentication or reporting purposes.
 - 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 logic decides when to initiate a secure session.
 - The AT-TLS policy says enabled On and ApplicationControlled On.
 - Based on that logic, the application uses the SIOCTTLSCTL IOCTL to tell AT-TLS when to start the TLS/SSL handshake and when to terminate it.

z/OS Communication Server supports AT-TLS 1.2 currency with z/OS System SSL. Support is added for the following functions:

- Renegotiation (RFC 5746) in z/OS 1.12
- Elliptic Curve Cryptography (RFC 4492 and RFC 5480) in z/OS V1.13
- TLSv1.2 (RFC 5246) in z/OS 2.1
- AES GCM cipher suites (RFC 5288) in z/OS 2.1
- Suite B Profile (RFC 5430) in z/OS 2.1
- ECC and AES GCM with SHA-256/384 (RFC 5289) in z/OS 2.1
- Twenty-one new cipher suites
 - 11 new HMAC-SHA256 cipher suites
 - 10 new AES-GCM cipher suites
- Requires new System SSL support
- Support for Suite B cipher suites

Implementing TLS protocols directly into applications (without using AT-TLS) requires modification to incorporate a TLS capable API toolkit. Only the z/OS System SSL toolkit supports RACF key rings and the associated advantages (user ID mapping, SITE certificates, and more).

Recommended reading: For more information about AT-TLS, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

2.1.7 IPSec

Internet Protocol Security (IPSec) family of protocols enables the secure communication over unsecure networks. Much like TLS, it encrypts the traffic and provides data integrity and data origin authentication. In difference to TLS, IPSec operates at the IP Layer, not the upper protocol layer like TCP. As such, it can be used to secure all traffic on the IP layer, regardless of upper layer protocols. IPSec connections are commonly referred to as *virtual private networks*.

IPSec defines a unidirectional connection between two endpoints. These connections are mostly referred to as *tunnels*. There are two types of IPSec tunnels:

- ▶ Manual IPSec tunnels

The security parameters and encryption keys are configured statically and are managed by a security administrator manually. Manual tunnels are not commonly implemented.
- ▶ Dynamic IPSec tunnels

The security parameters are negotiated, and the encryption keys are generated dynamically using Internet Key Exchange (IKE).

The characteristics of security in an IPSec communication are defined by Security Associations (SA). The concept of the SA is crucial to IPSec. An SA defines how a particular type of unidirectional traffic is protected between two endpoints. Because IPSec SAs are unidirectionally, they are typically established in pairs, one for inbound traffic and one for

outbound traffic. SAs may be defined in varying *widths*. For example, you may define an SA to protect all traffic between two different networks (a *wide* SA), only the traffic between two specific IP addresses and ports (a *narrow* SA), or somewhere in between. Let us now take a look at basic concepts of IPSec.

IPSec uses the z/OS Communication Server policy-based networking infrastructure. IPSECURITY policies are defined using the Configuration Assistant for both IP filtering and IPSec protection. The IPSECURITY policy is then read by the Policy Agent, which installs the IP filtering and IPSec rules into the TCP/IP stack and the Internet Key Exchange (IKE) daemon on z/OS.

IPSec is defined and maintained by the Internet Engineering Task Force (IETF) through a variety of RFCs, include these primary ones:

- ▶ RFC 4301: Security Architecture for the Internet Protocol¹
This RFC and its associated RFCs define the means of transporting data securely over an IP network
- ▶ RFC 2409: The Internet Key Exchange (IKE)²
This RFC and its associated RFCs define the initial version of IKE, now called IKEv1
- ▶ RFC 5996: Internet Key Exchange (IKEv2) Protocol³
This RFC and its associated RFCs define version 2 of IKE

Modes of transportation

There are two different modes of encapsulating IP packets under IPSec. The following list illustrates those modes:

- ▶ Transport mode
Protection of the IP payload information (including transport header, if present), but not the IP header. The IP header remains as-is and is used as usual when the packet travels through the network
- ▶ Tunnel Mode
Protection of the complete IP packet and inclusion of that packet into an outer IP packet. This concept is called encapsulation. The encapsulating IP packet is provided with a new IP address of the IPSec security endpoint, which does not necessarily have to be the data endpoint. This mode of operation is a true VPN.

Transport mode is only used when the security endpoints of the communication are also the data endpoints. Therefore, transport mode may be used only for host-to-host communication. With tunnel mode, the security endpoint could be different from the data endpoint. For example, the security endpoint is a network device like a router or a firewall, but the data endpoint is a host behind the router. Tunnel mode may be used to secure communication between two networks, a host and a network or two hosts. Figure 2-6 on page 56 illustrates the modes of operation.

¹ Available at <http://tools.ietf.org/html/rfc4301>

² Available at <http://tools.ietf.org/html/rfc2409>

³ Available at <http://tools.ietf.org/html/rfc4306>

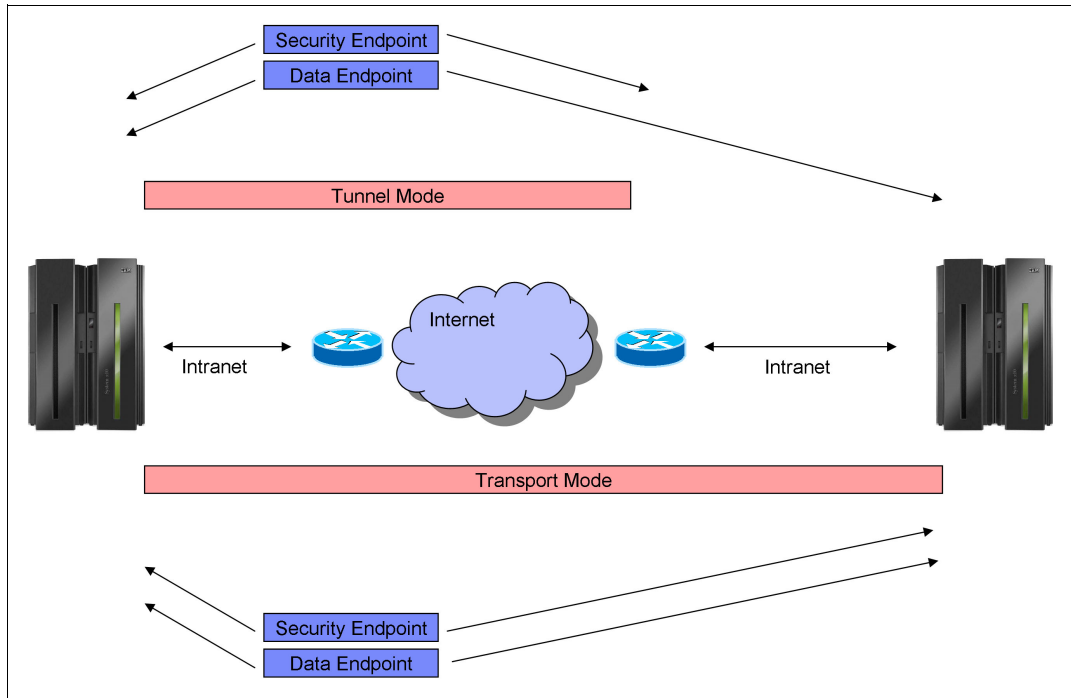


Figure 2-6 IPSec modes of operation

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 the following features:

- ▶ 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.

Authentication Header does not provide encryption of the data sent over the network. If you need encryption, you have to use Encapsulating Security Payload.

Encapsulating Security Payload protocol

Encapsulating Security Payload (ESP) provides additional protection beyond AH:

- ▶ It provides privacy protection by encrypting the IP packet.
- ▶ It applies authentication, integrity and privacy protection to the entire IP packet, including the IP header. This is possible because ESP completely encapsulates the original IP packet within an outer IP packet. For most users, the authentication protection provided by ESP should be sufficient, and AH should not be necessary if ESP is already being used for encryption.

It is best to use ESP over AH because of the reasons stated previously.

Internet Key Exchange protocol

Recall what we have said before in this section about manual and dynamic IPsec tunnels. Using a manual IPsec tunnel with pre-shared keys managed by a security administrator is not a scalable solution and can weaken the security of the symmetric keys, because these manually configured keys can be compromised easily and cannot be changed during an IPsec session. To refresh the symmetric keys for a given SA, the session has to be closed, then the key can be changed and a new session can be established. In most production environments, this is not an acceptable solution.

The IKE protocol, which is implemented in z/OS Communications Server by the IKE daemon, manages the transfer and periodic changing of security keys between senders and receivers and is required when implementing IPsec dynamic tunnels. Key exchange, defined in IKE, is normally a multistep process:

1. First, the partners authenticate each other and negotiate a secure logical connection, over which IPsec (AH and ESP) SAs can be negotiated between the two partners. called and IKE Security Association (IKE SA) and is sometimes referred to as the phase 1 tunnel or IKE tunnel.
2. After the logical connection is in place, the partners negotiate AH and ESP SAs to be used to protect specific types of data traffic (*IPsec or child SA*). These SAs are sometimes referred to as the *phase 2 tunnel* or *IPsec tunnel*. The IKE messages exchanged to negotiate the phase 2 tunnels are protected by the IKE SA.
3. Thereafter, all of these SAs and their associated session keys are renegotiated periodically. The IKE daemon uses the IP security policies that you define in the Policy Agent and to manages the keys dynamically.

The **IPsec** command can display, activate, refresh, and deactivate both IKEv1 and IKEv2 tunnels. Figure 2-7 illustrates the concepts of IKE for IPsec processing.

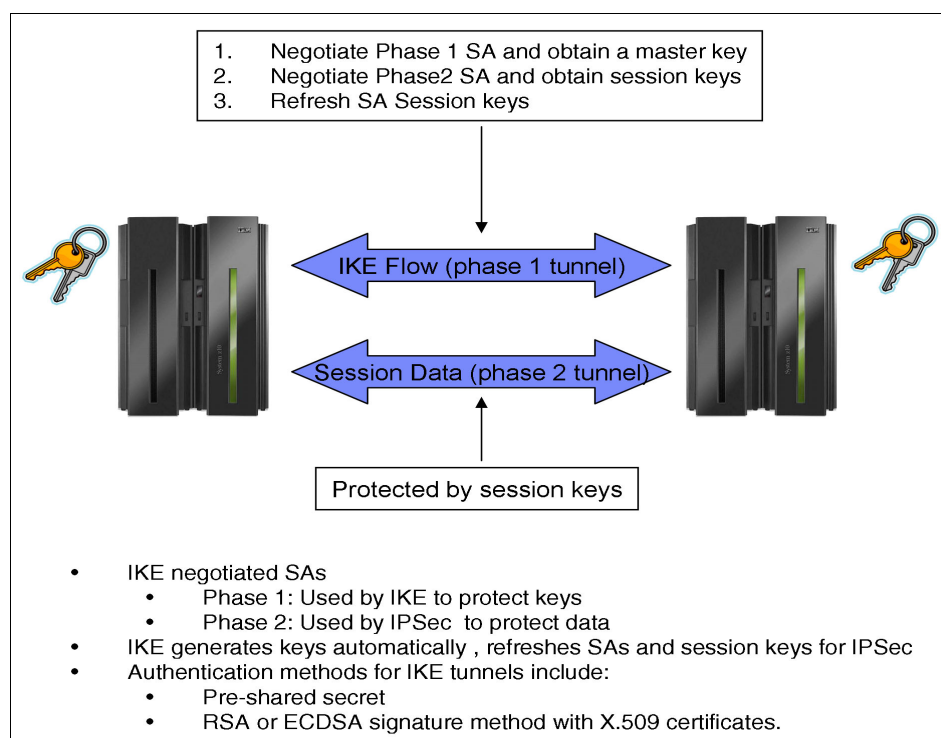


Figure 2-7 IKE flow for IPsec processing

There are two methods of authenticating the IPSec peers when using dynamic tunnels:

- ▶ Authenticate the IKE partner with a pre-shared key
- ▶ Authenticate the partner using digital signature mode

A *pre-shared key* defines secret values that are shared between the IKE peers to authenticate the partner during the Phase 1 IKE exchange and to provide a value to the Diffie-Hellman exchange that produces a cryptographic key to protect and authenticate the Phase 1 IKE negotiations.

Digital signature mode authentication relies on X.509 certificate exchanges to provide verification of the trusted partner. The z/OS IKE daemon (IKED) supports two digital algorithms: RSA and ECDSA. The certificates must be stored in an SAF key ring accessible to the IKE daemon. The local identity of an IKE peer must be configured in the IPSec policy, and it must represent an identity established in x.509 certificate on the local IKE key ring. The remote identity of an IKE peer must also be configured in the IPSec policy, and it must represent an identity established in the x.509 certificate presented by the remote IKE peer during Phase I negotiations.

When implementing digital signature mode on z/OS, certificate management operations can be performed by the local z/OS IKE daemon (IKED) or by a Network Security Services daemon (NSSD) running locally or in another, more secure network zone on z/OS. IKED supports local digital signature mode for IKE version 1 (IKEv1) negotiations only, but NSSD supports both IKEv1 and IKE version 2 (IKEv2).

When IKE is used, SAs for the secure communication are derived using exchange protocols defined in the relevant IKE RFCs. The mode of operation to perform the exchange might vary based on definitions in the IPSec policy. The following list provides an overview of the negotiation modes:

- ▶ IKEv1 Main Mode

This mode is used to establish a phase 1 tunnel and is also referenced as *Identity Protection Mode*. When using Main Mode, the identities of the communicating parties will be secured using a previously negotiated secret. Authentication, key exchange and SA information will be transported over the network separately. It takes longer, but it provides more complete protection than aggressive mode.

- ▶ IKEv1 Aggressive Mode

This mode is an alternative to Main Mode, also used to establish a phase 1 tunnel. It allows for the authentication, key-exchange and SA-related information to be sent together over the network. This is a faster approach than Main Mode using fewer messages, but identity information is transported in clear over the network.

- ▶ IKEv1 Quick Mode

This mode is used to negotiate phase 2 tunnels after a phase 1 tunnel is in place.

- ▶ IKEv2

IKEv2 only offers a single exchange mode that is more efficient than those offered by IKEv1. It combines the identity protection of IKEv1 Main Mode with a reduced number of messages (four) similar to IKEv1 Aggressive Mode. In addition, one complete IKEv2 initial exchange negotiates a phase 1 and a phase 2 tunnel. If additional phase 2 tunnels are required, then they can be negotiated using subsequent two-message exchanges (one less than IKEv1 Quick Mode).

FIPS 140-2 mode considerations

As we have described in “FIPS 140-2 considerations” on page 49, System SSL can be configured to operate in FIPS 140 mode. The z/OS IPsec implementation also provides a complete FIPS 140 mode of operation. These restrictions also apply to IPsec operation when FIPS mode is enabled. The IKE daemon, the Network Security Services daemon, and the TCP/IP stack’s IPSECURITY support can be configured for FIPS mode. As with System SSL, the IPsec FIPS 140 mode restricts the set of cryptographic algorithms and key lengths to ensure sufficient strength of the cryptographic operations.

Network Security Services daemon

The Network Security Services daemon (NSS or NSSD) provides several security services to different types of clients. For IPsec environments, NSS offers digital certificate and remote management services to z/OS IKE daemons. For appliances like IBM DataPower SOA appliances, NSS offers digital certificate and SAF-based authentication and authorization services. Access to all of these services is controlled through a thorough set of SAF resources and communications between the NSS server and its clients is protected using TLS/SSL.

- IPsec certificate service

You may configure the z/OS IKE daemon to use an NSS server to perform digital signature and certificate-related operations for one or more TCP/IP stacks. NSS is required for IKEv2 digital signature authentication and is optional for IKEv1. NSS IPsec Certificate Services perform all digital signature and certificate-related services required by IKE, including certificate hierarchy validation, certificate revocation checking through CRLs, and HTTP retrieval of Certificate Revocation Lists, certificates and certificate bundles.

- IPsec remote management service

The NSS IPsec remote management service allows an NSS server to act as a single point of z/OS IPsec management and control. Under direction of the IPsec command or a management program using the NSS Network Management API, the NSS server can request IPsec monitoring data from its NSS IPsec clients (z/OS IKE daemons) and make IPsec control requests, such as activating, deactivating or refreshing security associations.

- XML Appliance certificate service

For appliances like IBM DataPower SOA appliances, this NSS service allows the appliance to generate or verify digital signatures based on X.509 certificates. This service is used for digital certificates associated a secure private key (one that is protected by the master key of a Crypto Express coprocessor).

- XML Appliance private key service

For appliances like IBM DataPower SOA appliances, this NSS service allows the appliance to retrieve a private key from the NSS SAF key ring. This service is used for clear private keys (those not protected by a Crypto Express coprocessor).

- XML Appliance SAF access service

For appliances like IBM DataPower SOA appliances, this NSS service allows the appliance to perform SAF authentication and authorization checks using RACF or another SAF-compliant external security manager.

Recommended reading: For more information about IPsec, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

2.1.8 OpenSSH on z/OS

OpenSSH is primarily developed by the OpenBSD Project⁴, and its first inclusion into an operating system was in OpenBSD 2.6. The software is developed outside the US, using code from roughly 10 countries.

On z/OS, OpenSSH is shipped as part of the Ported Tools for z/OS product, among other well-known UNIX utilities such as bzip2, perl, and PHP. For information about current versions, see IBM Ported Tools for z/OS:

http://www.ibm.com/systems/z/os/zos/features/unix/port_tools.html

OpenSSH is an implementation of the classic UNIX Secure Shell (SSH) family of protocols. SSH is intended to be a secure version of the traditional BSD UNIX command line utilities: **rlogin** (remote login), **rsh** (remote shell), and **rcp** (remote copy).

- ▶ **rlogin** starts a terminal session from the local shell a remote host specified as host. The remote host must be running a rlogind service for rlogin to connect to.
- ▶ **rsh** executes a command, or provides remote shell access, on the specified remote host, which must be running the rshd service.
- ▶ **rcp** copies files between machines. Each file or directory argument is either a remote file name or a local file name.

SSH is actually a client/server protocol and a suite of connectivity tools. The SSH client program is a secure remote login program—that is, a secure alternative to rlogin or rsh. The SSH suite also provides sftp, which is a secure version of FTP, and rcp. The client program requests a connection to the SSHD secure remote login daemon running in the target host. The SSHD daemon handles key exchange, encryption, authentication of both server and user, command execution, and data exchange. By default, SSHD listens to port 22 for incoming connection requests.

OpenSSH, as does SSH, runs on top of the TCP/IP stack, provides security at the application layer, and can be thought of as a protocol with three layered components.

- ▶ The transport layer, which provides algorithm negotiation and key exchange. The key exchange includes server authentication and results in a cryptographically secured connection. It provides integrity, confidentiality, and optional compression of data.
- ▶ The user authentication layer uses the established connection and relies on the services provided by the transport layer. It provides several mechanisms for user authentication. These include traditional password authentication and public-key or host-based authentication mechanisms.
- ▶ The connection layer multiplexes many different concurrent channels over the authenticated connection and allows tunneling of login sessions and TCP-forwarding. It provides a flow control service for these channels. Additionally, various channel-specific options can be negotiated.

⁴ More information about the OpenBSD Project is available at <http://www.openssh.com/>.

Figure 2-8 on page 62 summarizes the interactions between an OpenSSH client and daemon. Notice that a secure channel is established after authentication of the server (the host), and then the communication proceeds with client-to-server authentication:

- ▶ Each host has a host-specific key (RSA or DSA) used to identify the host. Whenever a client connects, the server responds with its public host key. The client compares the RSA host key against its own database to verify if this key value is recorded as owned by this host. Packets sent by the host will be signed using the corresponding private key. Then, a Diffie-Hellman key agreement exchange occurs that results in establishing a shared session key and a session number.
- ▶ The rest of the session is encrypted using a symmetric cipher. The ciphers currently supported are 128-bit AES, Blowfish, 3DES, CAST128, Arcfour, 192-bit AES, or 256-bit AES. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (hmac-sha1 or hmac-md5).
- ▶ The client then authenticates using one of these negotiated authentication methods:
 - Public Key authentication (RSA or DSA)
 - Hosts file at the server
This method is based on the traditional UNIX `.rhosts` and `.shosts` files. It is an inherently insecure authentication mechanism.
 - Password
 - Challenge-response (not supported on z/OS)
- ▶ Then the commands are executed and the data are exchanged.

The following types of keys are therefore involved in authentication and protection of the secure channel:

Host key	An asymmetric key used by the server to provide the server identity
Session key	A dynamically generated symmetric key for encrypting the communication
User key	An asymmetric key used by the client to prove a user identity

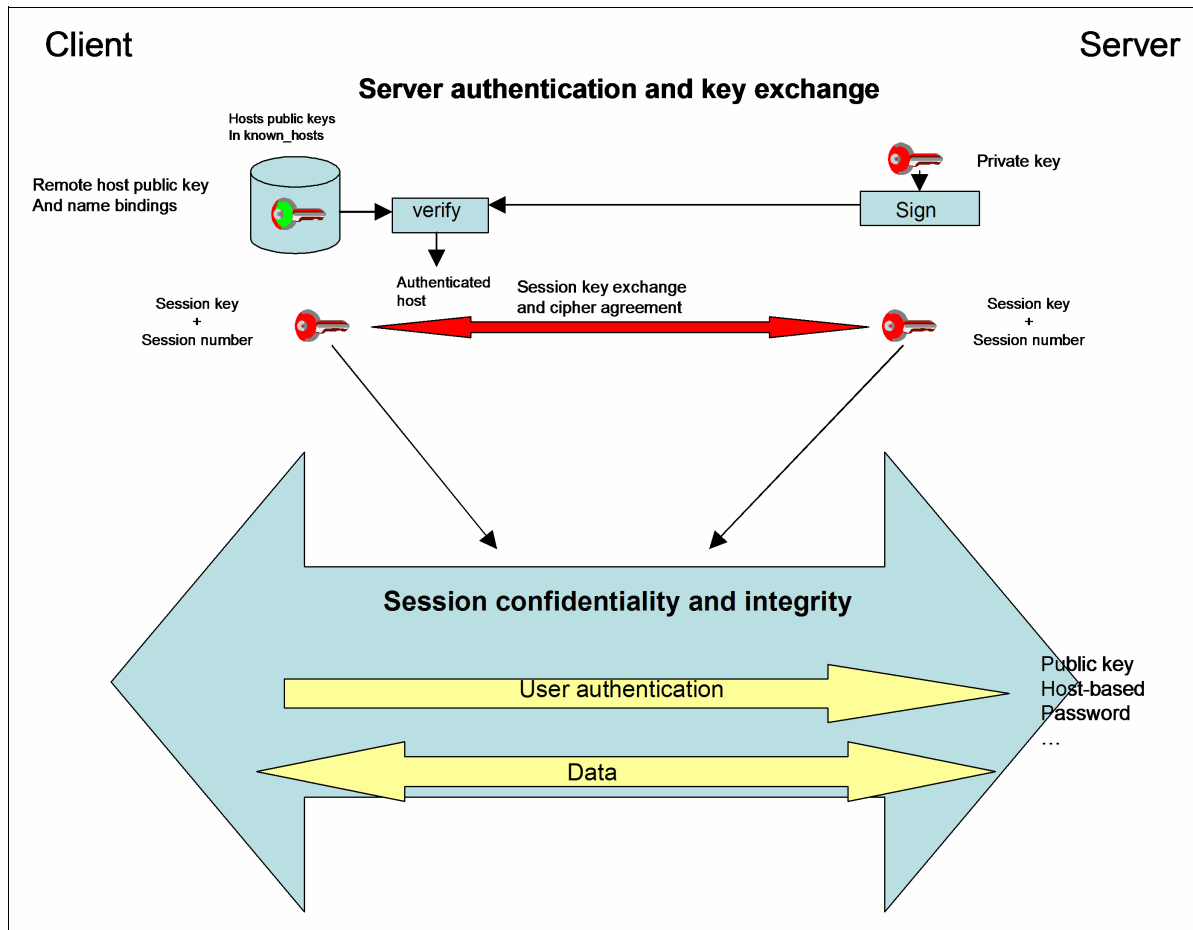


Figure 2-8 OpenSSH processing example

Positioning OpenSSH versus TLS

Making OpenSSH available on z/OS answers the many requests received from UNIX System Services users to have an SSH-like facility.

As shown, SSH was intended to provide a secure alternative to telnet and FTP by setting up an encrypted tunnel between hosts. Whereas TLS by itself does not provide any user service, it actually is a way for programmers who are creating any kind of network applications that use TCP sockets to build *in the application* strong authentication, data confidentiality, and integrity for data exchange.

The two protocols are overlapping in their capabilities as SSH can also provide a secure tunnel to other applications via its TCP forwarding capability, the latter potentially extending the protection to a range of many TCP-based protocols.

It must also be noted that TLS uses public keys packaged in x.509 certificates, and therefore requires using a PKI, whereas OpenSSH accommodates raw key values that are not embedded and certified in a digital certificate (although, strictly speaking, the IETF specifications open SSH to the use of digital certificates). In that respect OpenSSH requires a lighter infrastructure, as key certification is under the responsibility of a local administrator. However, this approach to administering keys might become difficult on a large scale.

In terms of the type and strength of the encryption algorithms, both SSL/TLS and OpenSSH are similar in that they both use common algorithms and key lengths. However, OpenSSH uses OpenSSL (packaged within OpenSSH) for its encryption services.

The z/OS UNIX Telnet server (otelnetsd) does not provide SSL/TLS protection and might not be protected using AT-TLS, because otelnetsd is implemented using socket interface that is not supported by AT-TLS (to be clear, however, TN3270 can be protected by AT-TLS). OpenSSH with its remote login function is an excellent, secure alternative to otelnetsd. In addition, OpenSSH with SFTP is a popular alternative to using TLS-protected FTP, but today z/OS SFTP implementation does not provide access to MVS data sets on its own. However, third-party products are available to bridge that gap.

z/OS OpenSSH implementation

In this section we describe the contents of the z/OS OpenSSH component.

OpenSSH provides z/OS UNIX System Services users with the capability of using an OpenSSH client (ssh) and server (sshd) running on z/OS. It provides the following suite of connectivity tools:

- ▶ Secure remote login (ssh), an alternative to rlogin and rsh.
- ▶ Secure file transfer (sftp and scp), an alternative to FTP and rcp.

Attention: The z/OS OpenSSH client cannot be invoked from the z/OS UNIX shell. This is working as designed, as there is no way to hide passwords entered in the shell.

Other basic utilities, such as ssh-add, ssh-agent, ssh-keygen, ssh-keyscan, ssh-keygen, and sftp-server are also included.

The IBM Ported Tools for z/OS implementation of SSHD support both SSH protocol versions 1 and 2 simultaneously. The default SSHD configuration runs only Protocol Version 2.

OpenSSH client and server

Take a moment to review the OpenSSH client and server:

ssh	This is the OpenSSH client that establishes the secure channel with the OpenSSH server and performs the secure remote login program.
sshd	Secure remote login daemon, a daemon that listens for connections from ssh clients and handles key exchange, encryption, authentication, command execution, and data exchange.

Together, SSH and SSHD provide secure encrypted communications between two hosts over a TCP/IP network. After an SSH session is established, other connections can be forwarded over this secure channel, such as X11, and other TCP-based protocols.

Secure file transfer

Let us take a closer look at secure file transfer using **SFTP** and **scp**. Be aware that there are many acronyms used to denote different implementations of FTP (secure or insecure). The term “sftp,” for example, might also stand for “simple file transfer protocol” rather than “secure file transfer program,” as we use it here:

- ▶ sftp (secure file transfer program)

An interactive file transfer program, similar to the FTP user interface. It performs all operations over an encrypted SSH transport and may also use many features of SSH. SFTP requests flow through the OpenSSH client and requires the sftp-server (SFTP server subsystem) be operating on the server-side of the SFTP protocol. The SFTP server is automatically invoked from SSHD.

- scp (remote secure file copy program)

This program also uses the OpenSSH client and server for data transfer. The command syntax is similar to rcp. However, unlike rcp, SCP asks for passwords or pass phrases, if necessary.

Key management

The code involved is specific to z/OS OpenSSH. There is no reuse of other key management code provided by IBM, such as GSKKMAN, IKEYMAN, and so on. The following utilities are provided for key generation and management:

ssh-keygen	Creates public/private key pairs.
ssh-agent	Holds private keys in memory, saving you from retyping your passphrase repeatedly.
ssh-add	Loads private keys into the agent memory.
ssh-keyscan	Gathers SSH public host keys from the client, via an SSH protected conversation. This is used in preparation of server authentication by the client.
RACDCERT	Has to be used to create a public/private key pair for z/OS OpenSSH when keys are stored in SAF key rings.

Helper applications

These are some of the helper applications:

ssh-rand-helper	Entropy gathering code for random number generation.
ssh-askpass	X11 GUI for passphrase entry, called by ssh-add .
ssh-keysign	Helper program that performs the digital signatures required for host-based authentication.

OpenSSH support specific to z/OS

OpenSSH for z/OS provides some features which are specific to z/OS and make use of z/OS own security infrastructure. These features will enable you to strengthen the security of UNIX System Services access to your systems if you are using OpenSSH. Most of these features were introduced with version 1, release 2 (1.2) of z/OS OpenSSH:

- System Authorization Facility (SAF). z/OS OpenSSH can be configured to retrieve keys from a SAF key ring, rather than a UNIX file. Using a SAF key ring in combination with RACF removes the need for storing sensible keys in the home directories of each user and centralizes them within the RACF database.

When used with SAF key rings, OpenSSH z/OS will also make use of the random number generation facility within the CPACF cryptographic coprocessor.

- Multilevel Security. z/OS OpenSSH supports the concept of multilevel security which allows the classification of data based on a system of hierarchical security levels.
- Systems Management Facility (SMF). z/OS OpenSSH can be configured to collect SMF type 119 records for both the client and the server.

Access to MVS data sets in OpenSSH for z/OS

OpenSSH for z/OS does not provide access to MVS data sets. You cannot open a sftp session to your UNIX System Services environment and touch any MVS data for transport within sftp. This support is limited to the original z/OS FTP and the z/OS shell environment. The OpenSSH for z/OS, which is part of the ported tools package, does not have that capability.

However, there are third-party vendor products that might offer you support for MVS data set access from within an OpenSSH sftp session. One of those products is Dovetailed Technologies Co:Z SFTP. Co:Z SFTP is a port of the OpenSSH for z/OS sftp implementation with MVS data set access included. For more information, see the “Co:Z SFTP” page on the Dovetailed Technologies website:

<http://www.dovetail.com/products/sftp.html>

Recommended reading: For more information about OpenSSH for z/OS, see *z/OS IBM Ported Tools for z/OS User's Guide*, SA23-2246-00.

2.1.9 PKI services

z/OS PKI Services deploys a full certificate authority solution running on z/OS. It was introduced as a base element at z/OS V1.3. The product has evolved since then, with several new functions incorporated in each release. Customers have taken advantage of this solution to issue and manage thousands of digital certificates in-house. z/OS PKI Services is a competitive solution using the robustness, availability, scalability, and security provided by System z. For a general introduction of PKI see “Public Key Infrastructure” on page 41.

Although PKI Services is not a component of the TCP/IP stack or z/OS Communication Server, let us take a closer look at the product to see how it can enhance your certificate management capabilities for network security.

z/OS PKI Services is a complete solution for managing the digital certificate lifecycle running under z/OS V1.3 and later.

Certificate generation and administrative functions are driven via customizable web pages. z/OS PKI Services supports browser and server digital certificates. It provides an automatic or administrator approval process and a user or administrator revocation process. It deploys email notification for requesters and administrators, informing them about certificate request completion, certificate rejects, expiration warnings, and pending requests.

z/OS PKI Services is closely tied to RACF or any equivalent product that supports R_PKIServ callable services. It supports more functions than the RACDCERT command.

z/OS PKI Services supports multiple revocation-checking mechanisms. It deploys CRLs (Certificate Revocation Lists) and OCSP (Online Certificate Status Protocol). These functions are not supported through RACDCERT commands. z/OS PKI Services provides the Trust Policy plug-in. It is an application interface for checking the certificate validation.

z/OS PKI Services is compliant with the PKIX architectural model, which was built by the IETF (Internet Engineering Task Force) PKIX working group. The PKIX working group was established in 1995 with the intent of developing Internet standards needed to support an x.509-based PKI.

According to this open standard, the digital certificate request has to be in PKCS#10 format; the Certificate Revocation List (CRL) has to be in x.509 V2 format within an LDAP directory as repository. The digital certificate format has to be x.509 V3. For more information about the PKIX standard and components, see the IETF documentation⁵.

⁵ Information is available at this website: <http://datatracker.ietf.org/wg/pkix/documents/>

PKI Services structure

Before describing the z/OS PKI Services structure, let us list the z/OS PKI Services solution requirements, and then we fit these elements inside the structure:

- ▶ IBM HTTP Server for z/OS
- ▶ RACF of any functional equivalent supporting R_PKIServ callable services
- ▶ PKI Services daemon
- ▶ OCSF: Open Cryptographic Services Facility
- ▶ OCEP: Open Cryptographic Enhanced plug-in (optional)
- ▶ sendmail (optional)
- ▶ LDAP directory
- ▶ ICSF (optional)

Figure 2-9 illustrates the structure of the PKI Services in z/OS.

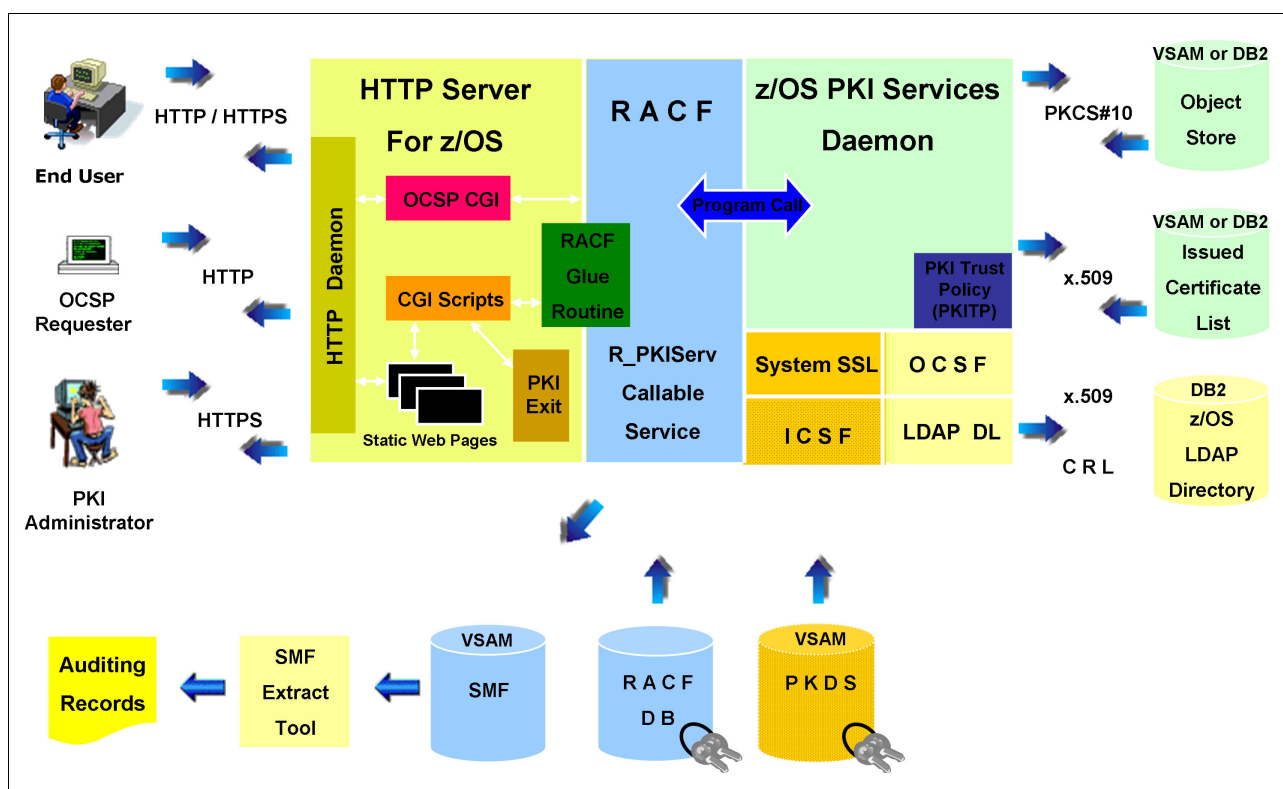


Figure 2-9 The structure of PKI Services within z/OS

IBM HTTP Server for z/OS

User certificate generation and administrative functions are performed through customizable web pages. IBM HTTP Server for z/OS is the web page interface for z/OS PKI Services. It is a z/OS base element.

The web page logic and contents are defined in a certificate template file. This file contains a mixture of true HTML and HTML-like tags. The Common Gateway Interface (CGI) script tasks read the template file to form the web pages and to control the flow. It also reads all of the input values from the web page and constant values from the template file to build the parameter list to call R_PKIServ callable services.

The CGI task also provides a hook to call PKI Exit, an installation-provided exit routine that is available for the following tasks:

- ▶ Providing additional authorization checking
- ▶ Validating and changing parameters
- ▶ Capturing certificates for further processing

SCEP (Simple Certificate Enrollment Protocol) and OCSP (Online Certificate Status Protocol) client interfaces run on CGIs under IBM HTTP Server, but no web pages are associated with these services.

The IBM HTTP Server for z/OS is not involved in generating key pairs for users or for server certificates. This function is driven by the browser Cryptographic Service Provider (CSP) to the device selected by the user (e.g., smart card, token, or the browser software itself). Key pairs for server certificates are generated by the software in the server that is going to use the certificate.

RACF

RACF provides support for the R_PKIServ callable services that is the core component of z/OS PKI Server solution.

R_PKIServ callable services is the interface between the CGIs and the PKI Services daemon. It performs authorization checking and parameter validation.

User functions such as those that request, export, verify, revoke, renew, and suspend certificate and respond to certificate are performed by RACF through R_PKIServ callable services. Administrator functions such as query, approve, modify, reject, suspend, resume, and revoke are also performed by R_PKIServ callable services. RACF is the only z/OS PKI Services component that is licensed.

RACF can also be used as a certificate store, as addition to VSAM or DB2. When certificates are stored in RACF, they can be easily connected to a user or server. This enhances the possibilities of using PKI Services created and managed certificates for use with network security components. For example, certificates to use for TN3270 users who are supposed to perform client authentication can be created and managed using PKI Services, but a copy of these certificates can be stored in RACF also. Within RACF, the certificate can be connected to the user ID and thus be used for verifying the connecting of the user to the certificate.

PKI Services daemon

The PKI Services daemon is provided by the z/OS PKI Services code. It is responsible for managing the services threads for incoming requests. It also controls the background threads for certificate approval and for issuing the Certificate Revocation List (CRL). The PKI Services daemon manages the VSAM databases for certificate requests (Object Store) and Issued Certificate List (ICL).

IBM DB2 for z/OS (optional)

By default, PKI Services uses VSAM data sets to store issued certificates and certificate requests. These data sets are perfectly fine if you want to use PKI Services in a monoplex environment, but become fairly complex to manage in a sysplex environment not only in terms of multiple system access but also in terms of performance. Since z/OS V1.13, IBM has offered a second way of storing this data, and that is via DB2 for z/OS. Using DB2, you are automatically under management of DB2 in terms of backup and recovery and this also might have a positive effect on performance in your environment.

OCSF, Open Cryptographic Services Facility

PKI Services requires OCSF to be installed and configured so that the user ID under which the PKI Services daemon runs can use required services.

OCEP, Open Cryptographic Enhanced plug-in (optional)

You need to install and configure OCEP if your installation plans to write an application to implement the use of PKI Trust Policy (PKITP).

sendmail (optional)

You need to configure sendmail if your installation plans to send email notifications to users for certificate-related events, such as certificate expiration.

LDAP directory

Use of an LDAP server is required to maintain information about PKI Services certificates in a centralized location. The z/OS LDAP server is preferred, but you can use a non-z/OS LDAP server if it can support the objectClass and attributes that PKI Services uses. Typical PKI Services use requires an LDAP directory server that supports the LDAP (Version 2) protocol (and the PKIX schema), such as the z/OS LDAP server.

ICSF (optional)

ICSF is preferred but not required. You can begin using PKI Services without installing ICSF and install it later without reinstalling PKI Services. We strongly suggest you use ICSF to store and protect your certificate authority's private key.

Recommended reading: For more information about PKI Services for z/OS, see *z/OS V1R11.0 Cryptographic Services PKI Services Guide and Reference, SA22-7693-11*.

2.2 Guiding principles for cryptography for network security

In this section, we cover some guiding principles in the area of cryptography for network security on z/OS. These principles should be regarded as a collection of preferred practices that the authors of this book have developed over time in many customer situations. We cover the following topics:

- ▶ Choosing appropriate cryptographic algorithms for network security
- ▶ Defining a cryptography strategy within your organization
- ▶ Choosing Transport Layer Security implementations
- ▶ Things to keep in mind when defining certificates
- ▶ Guiding principles for IPSec
- ▶ OpenSSH on z/OS UNIX, z/OS dependant features implementation

2.2.1 Choosing appropriate cryptographic algorithms for network security

Cryptography has been used in network security for quite a long time now. Information technology has evolved over this time and computing power available to individuals and organizations has increased dramatically and still does. We will now take a look on how this development affects network security when cryptography is used. Cryptographic algorithms are not by default secure. There are several algorithms and variations that are still widely used but seen as unsuitable for protecting critical data not only by subject matter experts but also by government regulation bodies.

The security of cryptographic algorithms is assessed continuously by the industry and computer scientists. Cryptoanalysis describes this approach where cryptosystems and algorithms are examined to prove their security strength and find weaknesses. In this section we give a short overview of the current state of the art for common cryptosystems and algorithms and we will provide notes on how to find relevant information regarding these topics. We will focus on the cryptosystems used in z/OS network security components. Later in this chapter, we will take a closer look at crypto-configuration options for the z/OS network security components described in 2.1.4, “Overview of the z/OS TCP/IP cryptographic infrastructure” on page 44.

Symmetric encryption algorithms

The strength of a symmetric encryption algorithm is based on the length of the key, if you imply that there are no known practical cryptoanalysis attacks against this cryptographic algorithm. The measure of key length is also known as *security strength* of the algorithm. The security strength of a symmetric algorithm is supposed to be at least 128 bits today. Known attacks against a security algorithm can reduce the real security strength as opposed to the key length. It might be the case that an algorithm with a key length of 128 bits only has a security strength of 80 bits because of known attacks that weaken the algorithm. If so, the algorithm should not be used. It is desirable that only the brute-force method is suitable to break the encryption and gather the key used. Therefore, to receive a ciphertext of an unknown clear text and guess the keys that it takes to get useful information out of the ciphertext. This means to receive a ciphertext of an unknown clear text and guess keys values for as long as it takes to find the correct key to get useful information out of the ciphertext. As of today, a recommended symmetrical algorithm will be known to have no other attack than brute force.

The longer the key, the more operations it takes to guess the key from a received ciphertext. For example, take a random ciphertext that has been encoded using a 32-bit symmetric algorithm. Example 2-1 shows how much time it would take statistically to compute the key out of the ciphertext.

Example 2-1 Duration of a brute force attack against symmetric key algorithms

Key size in bits: 32

Number of possible keys: $2^{32} = 4.3 * 10^9$

Time it takes to guess the key at 1 decryption per microsecond:

$2^{32} * 1 \text{ microsecond} = 36 \text{ minutes}$

We can see now, that using a 32-bit symmetric key is really not a good idea because even if you consider just one decryption operation per microsecond it would only take a maximum of 36 minutes in to guess the key. Starting with 128 bits, these figures become more comfortable, as it would then take $5.4 * 10^{24}$ years to guess the key with one decryption per microsecond. Even if today's computing power were able to reduce the time by a factor of 100 or more, the time to decrypt using brute force would still be measures in hundreds or thousands of years to get one single key.

The following list provides a basic overview about widely accepted and used symmetric keys which offer a sufficient amount of encryption strength and are not known to have weaknesses other than brute force attacks. All of these standards also make use of the cryptographic hardware in System z.

- ▶ Advanced Encryption Standard (AES) with key lengths of 128, 192 and 256
- ▶ Data Encryption Standard (DES) with three-key triple encryption known as *Triple DES* (TDES or 3DES)

The following symmetric ciphers used with z/OS network security components have to be considered carefully nowadays because there are known attacks against them or the key length is not sufficient anymore:

- ▶ DES with a key length of 56 bit

DES 56 is not sufficient for encryption because of its small key length. As shown before, using sufficient amounts of computing power, keys used can be easily calculated using brute-force.

- ▶ Rivest Cipher (RC) 2 and RC4

RC2 and RC4 are proprietary and have never been disclosed to public. By using an algorithm that is not published, you weaken your security. There are known attacks which sufficiently reduce the computing power needed to derive a key from a random ciphertext.

- ▶ Two-key triple DES

This algorithm uses three keys for encryption, each 56 bits long. But two of those keys are equal to each other and therefore the security strength is not 168 as you would imagine but really only 112 bits. Also, there is a known attack reducing the overall security strength to 80 bit if the attacker is in possession of 2^{40} plaintext/ciphertext pairs of 64 bit (one block of data used for encryption).

The frequent change of keys during an encrypted data session using symmetric algorithms enhances the security. If the keys are changed regularly and there is no chance of suggesting a key from its predecessors, an attacker will have practically no chance to discover the keys.

A symmetric cipher is always only as secure as the key exchange algorithm used and the quality of the key. Many protocols will use Diffie-Hellman key exchange, where the underlying mathematical problem to generate the key is believed to be practically unsolvable. This is a method of asymmetric cryptography and therefore requires a public key to be sent between both parties. This public key exchange requires proper authentication and signature of the key sent, because an attacker might perform a man-in-the-middle attack and establish separate communication sessions with both parties forging their identity. PKIs including trusted third parties can provide proper authentication and identification of the key-holding parties and ensure the integrity of the key.

Because of the importance of key authentication using PKIs for symmetrical cryptosystems, special care has to be taken with PKIs and their underlying asymmetric cryptosystems, which we describe in the next section.

Asymmetric algorithms

Asymmetric cryptosystems use an approach where there is a public key and a private key that is kept secret by his owner. We will only take a closer look at Rivest Shamir Adleman (RSA) cryptographic algorithm in this chapter. This algorithm is essential to modern public key infrastructures using certificates.

The security strength of RSA is based on the problem of factorizing large prime numbers. A part of the public and private key, let's call it N is the product of two large prime numbers p and q . N is the modulus of the RSA operation and known to everybody. The second part of the public and the private key, e and d , are derived from p and q . Let (e, N) be the private and (d, N) be the public key pair. It is not possible to derive d , knowing N and e and vice versa. This is due to the fact that large prime numbers cannot be factorized efficiently. If that was possible, N could be factorized into p and q , and (e, d) could be simply computed again from (p, q) . The size of N is commonly known as the *key length* of RSA.

The key length of RSA is directly connected to the size of the underlying prime numbers, simply because it is the factor of both. Therefore, the key length must be sufficiently high to have the underlying prime numbers as large as possible. The larger these numbers are, the harder it gets to factorize them using appropriate algorithms. A key length of 1024 bit (960 decimal digits) is not sufficient anymore and is commonly rated as providing a security strength of only 80 bits. Current industry recommendations are to use RSA keys of at least 2048 bits.

Elliptic curve algorithms

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).

The security strength of an elliptic curve algorithm with a key length of 160 bit is, just as with RSA, only 80 bits. To efficiently protect data secured by elliptic curves, at least 224 bit of key length should be used.

Secure hash functions

For a brief description of hashing algorithms, see 2.1.3, "Applications of cryptosystems for network security" on page 40.

A hashing algorithm takes an arbitrary-length message as input, and produces a small, fixed-length DigestString (usually 128 bits or more). This hash can be thought of as a summary of a message. There are two important things to remember 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 unfeasible to produce another message that would produce the same message digest as another message.

The following are the common message digest algorithms:

- ▶ 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. Because 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. The use of MD5 is no longer recommended, because a variety of weaknesses, including collisions, have been found in it.

- ▶ SHA-1

Developed by the National Security Agency (NSA) of the US 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). Although the integrity of the SHA-1 algorithm remains intact, the security of a hash algorithm against collision attacks is one-half the hash size. 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 applications such as digital signature and HMACs. Because of this, in recent years, the industry has discouraged the use of SHA-1 and has begun to recommend the use of SHA-2, with its longer key lengths, for these purposes.

- ▶ SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)

Developed by the NSA of the US Government. SHA-2 addresses the weakness of the SHA-1 digest length as described previously. Extensions to the SHS have been developed to generate hashes of 224, 256, 384, and 512 bits, respectively.

Examining your existing network cryptography-related configurations

Within your network cryptography infrastructure, you may have several product configuration points where parameters and definitions for network cryptography are stored. The authors of this book recommend, that you revisit your existing cryptography configurations and examine the cipher suites that are used. Often, product parameterization is performed once, when the product is installed, or only rarely when new functions are implemented. The problem is that cryptography-related configurations are not always audited, except when there is a problem. If there is no attention on these configurations, old and possibly security-weakening values might be used for a long period of time.

As you have seen before, there are tremendous differences in security strength within the range of cryptographic algorithms that are supported on z/OS. For network cryptography, the following component configurations should be examined:

- ▶ TN3270
- ▶ FTP
- ▶ AT-TLS
- ▶ IPSec

The cryptography-related configuration is either stored in the product configuration files and profiles, or within the policy-based networking infrastructure of z/OS Communication Server.

Recommended reading: For more information about configuring network cryptography parameters, see *z/OS Communications Server: IP Configuration Reference, SC27-3651-00*.

Where to find more information

Government regulation bodies and research facilities issue recommendations about the latest developments and offer best practices and guidelines on which cryptographic algorithms and which security strengths to use. A security-aware enterprise should consult these sources regularly to keep the encryption in the network up to date. If you are under regulation of some kind of government agency, you might be obligated to adhere one of these standards.

The following list provides some sources for further information:

- ▶ NIST publications, including *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, SP 800-131 A and *Recommendation for Key Management*, SP 800-57
<http://csrc.nist.gov/publications/PubsSPs.html>
- ▶ Recommendation for cryptography issued by the German Bundesamt für Sicherheit in der Informationstechnik (federal office for security in information technology)
https://www.bsi.bund.de/EN/Home/home_node.html
- ▶ A guide on how to determine key lengths for public key cryptosystems used to transfer symmetric keys.
<http://tools.ietf.org/html/rfc3766>
- ▶ A collection of cryptography recommendations issued by the National Security Agency (NSA) of the US.
https://www.nsa.gov/ia/programs/suiteb_cryptography/

2.2.2 Defining a cryptography strategy within your organization

The following is not a technical guiding principle but more related to organizational issues that we want to make you aware of. Recall section 2.1.2, “Definition of a secure communication model for networks” on page 39 where we defined a communication model for secure networks introducing the communicating parties and the potential attacker. This communication model can be easily expanded to represent an enterprise network communication model. The communicating parties may be users, servers, network segments, or even enterprises. In between, the communicating parties may be network infrastructure, such as routers, firewalls, and gateways.

The communication model also describes endpoints for security, meaning that information transported over the network is somehow transformed by cryptography. From this point forward, no clear text is sent over the network until the data arrives at the opposite security endpoint. This is not necessarily the designated receiver; it might also be a network device on the way.

For example, if a branch employee wants to communicate with a server in the internal network of the company, all data leaving the computer of the employee can be secured by using cryptography. When this data enters the internal network of the customer, the gateway can be the security endpoint. Incoming data can be decrypted by the gateway and then sent over the internal network in clear text.

When talking about end-to-end security, people might think about security from sender to the actual receiver, but that might not be the case in reality. Figure 2-10 on page 74 illustrates several scenarios for security endpoints within a network infrastructure.

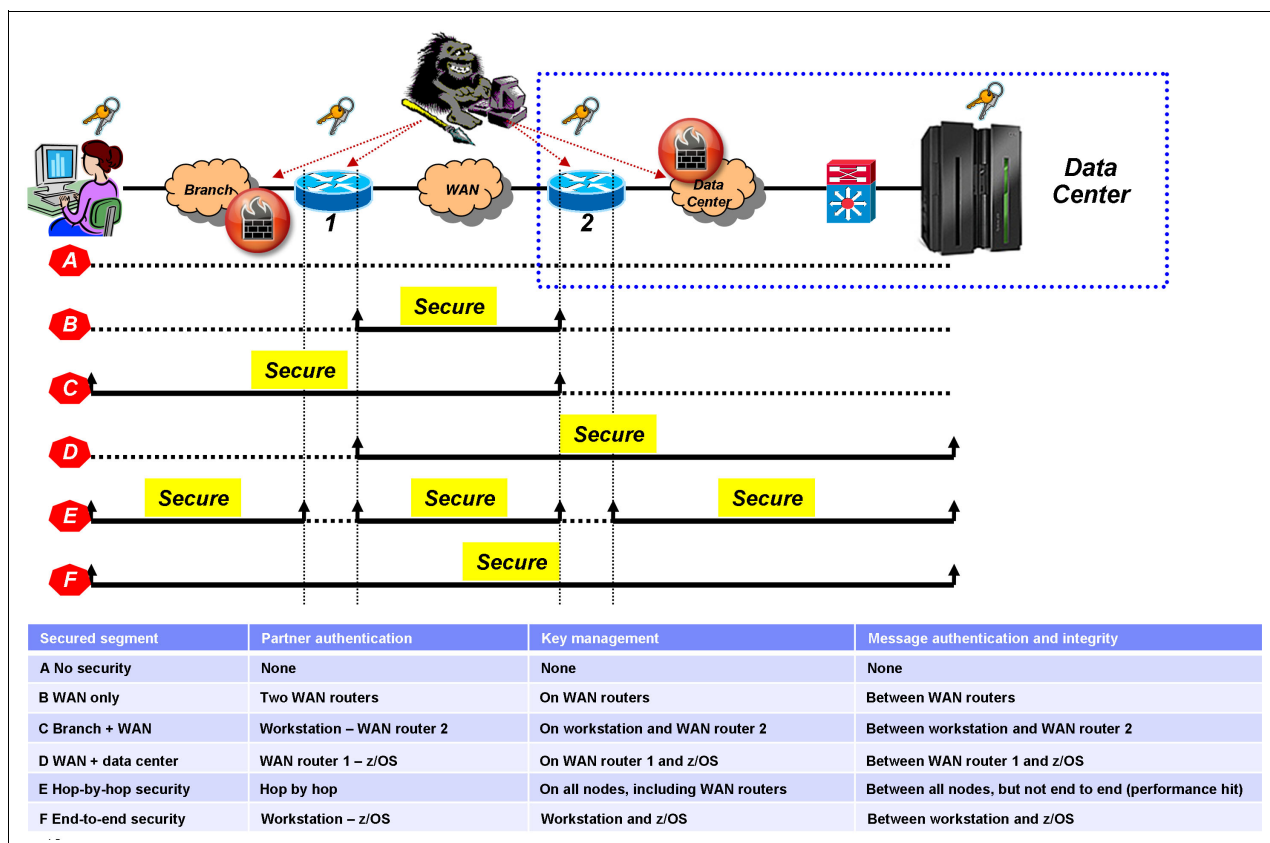


Figure 2-10 End-to-end enterprise network security

Take for example scenario B. Between the gateway from the branch office to the internet and the gateway to the internal company network, a VPN is to be implemented. This VPN secures data transferred over the internet. But neither within the branch office nor in the internal network of the company, the data is secured. This type of approach should be used only in situations where the integrity and security of the private networks (the one within the enterprise and the one within the branch office) is assured. Without such assurances, an attacker sitting behind one of the gateways could easily read and manipulate the data and there is also no authentication of the entities communicating to each other. The user and the server will not securely confirm their identities and thus the communication cannot be seen as fully trusted.

A complete end-to-end security is shown in scenario F. From the workstation of the user until the arrival at the server, all communication is secured. Complete end-to-end security is often required based on the sensitivity of the data that flows over the network or due to regulatory compliance. But to accomplish this state, several things have to be taken care of. The following list provides guidance when you get into discussions with network specialists and architects within your enterprise:

- How do your company's firewall, deep packet inspection, and network security policies fit in with the options?

The capabilities of stateful firewalls, intrusion detection systems, network analyzers, and content-based routers might be affected when end-to-end encryption is used.

- ▶ Does corporate security policy dictate a specific technology or requirement?
We have seen customer situations where an enterprise policy enforcing the use of deep packet inspection would not allow for an end-to-end encryption to be implemented, because the policy is violated when the packet inspecting device is unable to inspect encrypted traffic. It might also be that your security policy demands TLS for certain traffics and IPSec for other traffic. Reconsider if these requirements are really satisfying your needs and if your applications are capable of providing these technologies.
- ▶ What are the capabilities of the hosts and network equipment? Both endpoints of a secure connection must support the same cryptographic algorithms.
As we have seen before, several cryptographic algorithms have shown weaknesses against attacks or are generally seen as unsafe. Some applications in your enterprise might rely on those. Do take this into account when planning for enterprise-wide encryption. Carefully assess all participating applications and infrastructure to ensure that the desired cryptographic standards are supported.
- ▶ What are your communication partners willing and able to use?
Applications have different needs for certain technologies. For example, some applications might offer no security at all and therefore need to be secured using IPSec or similar. As another example, many UNIX or LINUX shops will permit only the use of SSH's sftp protocol to protect file transfers.
- ▶ Are relative security infrastructures already in place? For example:
 - Is there already a Public Key Infrastructure (PKI) in place?
 - Is TLS or IPSec already deployed anywhere in the network?
 - What method will you use to distribute public keys for SSH?
- ▶ Do the security protocols support the transport protocols?
TLS works great for TCP, but nothing else, whereas IPSec protects any IP traffic, regardless of transport protocol.
- ▶ Is the application already enabled for network security?
TLS-enabled applications might offer features based on the TLS integration and if not, consider application-transparent technologies.
- ▶ What do you want to authenticate?
Application or user identity or IP node identity? (TLS authenticates the identity of the application or user and is visible to the application whereas IPSec authenticates at the host IP node level.)
- ▶ How are the different technologies implemented on the platforms involved? Possibilities:
 - Performance optimization, such as hardware crypto and other acceleration technologies
 - Exploitation of other platform-specific features (secure key, SAF, and so on)
- ▶ What are the responsibilities for different security areas within your organization?
Just because RACF has been the traditional *seat* of security policies in the organization, it doesn't mean that the security administrator should implement networking security.

There might be many more things to consider within your enterprise. The guiding principle here should just be a reminder that even though you might have cryptography in place, you might need to define a crypto strategy end-to-end for your enterprise. Doing this, and including a detailed assessment of the current structure and implementation, will help you strengthen your network security and take control over this crucial part of IT within the company.

2.2.3 Choosing Transport Layer Security implementations

As we discussed in 2.1.5, “Transport Layer Security on z/OS” on page 46, z/OS offers three different approaches to TLS/SSL protection of TCP traffic: System SSL, JSSE, and AT-TLS. Let's examine some of the factors you might consider when choosing between these three approaches.

System SSL

Applications written in C or C++ can call the functions provided by System SSL. System SSL is a complete TLS/SSL implementation that is part of the z/OS Cryptographic Services component and ships with the base operating system.

JSSE

As we discussed in “TLS implementation on z/OS” on page 47, the Java Secure Socket Extension (JSSE) enables secure Internet communications using TLS/SSL for Java applications. It is a complete TLS/SSL implementation written 100% in Java (a completely separate implementation from System SSL). The main benefit of using JSSE for Java applications is that, because it is 100% Java, TLS/SSL operations are eligible to run on zAAP special purpose processors, rather than executing it in a central processor (CP). This is a cost effective way of enabling your Java applications to use TLS/SSL.

AT-TLS

AT-TLS is part of the Communication Server policy-based networking infrastructure.

AT-TLS offers several advantages over direct calls to System SSL. First of all, you do not have to deal with the TLS implementation at the source code level. You are not obligated to invoke any API calls in your application. You also do not have to take care of the correct configuration of TLS within your application. This is done transparently for all TLS-enabled applications using the Policy Agent. The configuration of AT-TLS is therefore outside the scope of application development and in scope for network management, where it belongs. This leads to a consistent TLS configuration across your application infrastructure. It reduces the risk of applications using weak encryption techniques by enabling the network security administrator to ensure consistency across AT-TLS policies for different applications.

AT-TLS makes the vast majority of System SSL features available to applications and as new features are added to System SSL, AT-TLS will usually be enhanced to expose those new features. Because of this, your applications can take immediate advantage of such enhancements in System SSL, usually with no development effort. All you need to do is update your AT-TLS policy to use new features.

Let's review the advantages of AT-TLS over basic System SSL use:

- Reduce cost

Using AT-TLS eliminates or greatly reduces the amount of code you need to write in your application programs to achieve TLS protection. Only applications that require some awareness or control of the TLS sessions need to be modified. This saves development cost not only for the initial TLS enablement, but also over time as System SSL features become available. Also, AT-TLS policy can reduce administrative costs associated with TLS protection because it presents a consistent and centralized administrative model across applications. Because of this, a single person skilled in configuring AT-TLS for one application can easily do it for other applications.

- ▶ Up-to-date exploitation of System SSL features

AT-TLS is regularly enhanced to expose new System SSL features. Because of this, applications protected by AT-TLS can take advantage of new System SSL features simply by updating the relevant AT-TLS policies. Source code changes are rarely required.

- ▶ Performance benefits

The interactions between AT-TLS and System SSL have been optimized and will often perform better than an application's direct use of System SSL.

- ▶ More control over the TLS implementations in your system

There is no need for a unique TLS/SSL implementation in any of your applications, but the TLS/SSL configuration is administered centrally for all applications in the system.

Because of these benefits, we recommend that you use AT-TLS to provide TLS protection to TCP applications that have not already been modified to use System SSL. And even for your applications that do use System SSL directly, we suggest carefully considering conversion to AT-TLS the next time you have to make source code changes to use a new System SSL feature. You might find that the cost of converting to AT-TLS is justified by the longer-term benefits described.

Many IBM-supplied applications and products, including these, use AT-TLS:

- ▶ z/OS Communications Server applications and features:
 - ▶ TN3270
 - ▶ FTP (client and server)
 - ▶ NSS daemon
 - ▶ IKE daemon (when acting as an NSS client)
 - ▶ CSSMTP server
 - ▶ Load Balancing Advisor
 - ▶ Centralized Policy Server
 - ▶ CICS sockets

Other IBM products:

- ▶ IBM DB2
- ▶ IBM IMS Connect
- ▶ IBM JES Network Job Entry
- ▶ IBM RACF Remote Sharing Facility
- ▶ IBM Tivoli MultiSystem Manager
- ▶ IBM Tivoli NetView Management Console
- ▶ IBM Debug Tool for z/OS

Case study: Converting a System SSL application to use AT-TLS

Let us now consider a real-world example of converting a System SSL application to use AT-TLS.

The z/OS Communication Server TN3270 server, FTP server and FTP client used System SSL for TLS/SSL protect before AT-TLS existed. When AT-TLS was developed, these applications were modified to use AT-TLS as the preferred TLS protection mechanism. Let's look at TN3270 to see how this was done.

Note: To maintain compatibility for customers who were already using the System SSL support, AT-TLS enablement was added as an option, rather than completely replacing the direct System SSL integration. Whether or not you would need to maintain such compatibility with an earlier version for your own application depends on your individual application requirements.

With respect to TN3270, AT-TLS offers several advantages over System SSL:

- ▶ Support for the latest TLS versions and cipher suites offered by System SSL
- ▶ Dynamically refresh a key ring
- ▶ Support new or multiple key rings
- ▶ Specify the label of the certificate to be used for authentication rather than using the default certificate
- ▶ Support SSL session key refresh
- ▶ Support SSL session reuse
- ▶ Support ID caching for SSL sessions in a sysplex
- ▶ Trace decrypted SSL data for Telnet in a data trace
- ▶ Receive more granular error messages in syslog for easier debugging

To enable AT-TLS support, you have to edit the TN3270 profile within z/OS Communication Server configuration, to instruct the server to rely on the AT-TLS policy, rather than what might be specified in the configuration. This is done by specifying the TTLSPORT statement rather than the SECUREPORT statement within the TN3270 configuration. Example 2-2 shows a sample AT-TLS enabled Telnet 3270 configuration.

Example 2-2 TELNETPARMS definition for AT-TLS port in TN3270 configuration file

```

TELNETPARMS
  TTLSPORT 4992           1 ; Port 4992 support AT-TLS
  CONNTYPE SECURE
; key ring SAF TCPIP/SharedRing1 2 ; omit - defined in Policy Agent
; CLIENTAUTH NONE          2 ; omit - defined in Policy Agent
; ENCRYPT SSL_DES_SHA       2 ; omit - defined in Policy Agent
;       SSL_3DES_SHA
; ENDECRYPT
  INACTIVE 0
  TIMEMARK 600
  SCANINTERVAL 120
  FULLDATATRACE
  SMFINIT 0  SMFINIT NOTYPE119
  SMFTERM 0  SMFTERM TYPE119
  SNAEXT
  MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 4992
  DEFAULTTLUS
    SC33DT01..SC33DT99
  ENDDEFAULTLUS

USSTCP  USSTEST1          ; Use USSTABLE USSTEST1
  ALLOWAPPL SC3*           ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *              ; Allow all applications that have not been
                           ; previously specified to be accessed.
ENDVTAM

```

As you can see in the example, when AT-TLS is chosen as the TLS protection mechanism for TN3270, several of the configuration parameters that serve as input the direct System SSL integration are not longer used. AT-TLS obtains these values from the relevant AT-TLS policies

Recommended reading: For more information about configuring AT-TLS for Telnet 3270, see *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996.

2.2.4 Things to keep in mind when defining certificates

Digital certificates play an important role in network security. Certificates are used by the TLS/SSL and IKE protocols to authenticate security endpoints, such as servers and clients (TLS/SSL) and IP nodes (IKE). We briefly describe what a certificate is and then lay out important considerations when certificates are defined in your environment.

A certificate is a digital document that proves the identity of the certificate owner and binds that identity to a specific public/private key pair. The commonly used type of certificate is X.509, standardized by the IETF (see RFC 2459). Certificates make use of public-key cryptography. A public key is contained in every certificate. The corresponding private key resides in the key database of the owner of the certificate.

The owner of a certificate is uniquely defined using an X.500 distinguished name (DN). This is a hierarchical representation of the owner within a hierarchy (typically the enterprise hierarchy represented in LDAP). The DN defining the owner of a certificate is called *Subject DN*. The DN defining the issuer of the certificate (that is, the trusted third party) is called *Issuer DN*. The issuer signs the certificate using its private key. The public key of the issuer is contained in its certificate, which must be available to all communicating partners. Anybody can now verify the validity of the certificate by using the issuers public key.

Choosing encryption algorithms when defining certificates

Be aware that a common mistake made when working with certificates is to choose encryption algorithms or key lengths which might not be supported in all participating products. These two parameters need to be supported by all products involved in secure communication which uses certificates. If a peer does not support the signature algorithm or key length in your certificate, then you will be unable to establish a connection with that peer. To prepare in advance for situations like these, make your mandatory encryption algorithms a policy to which all application owners and your administration agree on. See 2.2.2, “Defining a cryptography strategy within your organization” on page 73 for more information.

Default key length to be used when creating certificates

Be aware that the default key length offered by z/OS certificate management functions for public/private key pairs might not be sufficient for your individual security needs. As for RACDCERT command in RACF, this is 1024 bit for a private RSA key. This is not sufficient where strong security is required, as we explained in 2.2.1, “Choosing appropriate cryptographic algorithms for network security” on page 69. Be sure that your certificates at least use a key length of 2048 when using RSA (which is most common). The documentation provides guidance about how to relate the key length of RSA to other algorithms.

Figure 2-11 shows a screen capture of the RACF Panels, option 7.1 where a new certificate is created and the default key length is provided with 1024 for non-ECC keys. It is common for users to skip these values and leave the default value.

RACF - Generate a Digital Certificate

COMMAND ==>

More: - +

Enter the decimal size of the private key:

_____ (Default is 1024 for non ECC keys, 192 for ECC keys)

Select the key type to be generated:

- RSA(default)

- RSA in PKDS with an optional PKDS label or *:

- RSA in specified TKDS token:

- RSA Modulus-Exponent in PKDS with an optional PKDS label or *:

- DSA

- NIST ECC

- NIST ECC in PKDS with an optional PKDS label or *:

- NIST ECC in specified TKDS token:

- Brainpool ECC

- Brainpool ECC in PKDS with an optional PKDS label or *:

Figure 2-11 RACF certificate creation panels, default key length

PKI Services for z/OS will also use default key lengths of 1024 bit for non-ECC keys. If you are using PKI Services in your z/OS environment, check your templates for the default key lengths.

If you are using the **gskkyman** utility in your installation, it presents the user with a selection of different certificate types, including several key lengths. The possibility of using only the default is not an option when using this utility. Figure 2-12 shows the certificate type selection panel for **gskkyman**.

Certificate Type

1 - CA certificate with 1024-bit RSA key

2 - CA certificate with 2048-bit RSA key

3 - CA certificate with 4096-bit RSA key

4 - CA certificate with 1024-bit DSA key

5 - User or server certificate with 1024-bit RSA key

6 - User or server certificate with 2048-bit RSA key

7 - User or server certificate with 4096-bit RSA key

8 - User or server certificate with 1024-bit DSA key

Select certificate type (press ENTER to return to menu): 5

Figure 2-12 gskkyman certificate type selection panel

Renewing keys

When a certificate approaches its expiration date, you can renew the certificate and continue using it. You can choose to renew the certificate using the same private key, thereby extending the life of the private key. Or you can retire the private key and replace it with a new private key (also called certificate *rekeying* or *key rollover*).

A certificate renewal must be performed *before* a certificate expires. When the certificate has expired, renewal is not possible. Also, the certificate *must* not be revoked. If it is revoked, it becomes unusable.

When you renew a certificate using a new private key, you retire the private key and replace it with a new one. This process is commonly called certificate rekeying or key rollover. You choose this option to prevent a private key from being overused. (The more a key is used, the more susceptible it is to being broken and recovered by an unintended party.)

All information in the renewed certificate is updated to reflect the renewal, including the key ring connection information. After you retire and replace the old certificate, you can now begin to use the new certificate and its private key. You can continue to use the old, retired certificate until it expires to verify previously generated signatures. However, you cannot use the retired certificate to create new signatures. Additionally, do not connect the retired certificate to any key rings as the default certificate.

In conclusion let us just state that it is a better approach to always renew the certificate, resulting in a new private/public key pair. If a private key is always reused when the certificate is renewed, this is comparable to never changing a password. If a password is reused over and over again, which is also against password policy in most organizations, this weakens security. The same holds for private keys in certificates. It provides an attacking vector, because an intruder might have more time to guess the private key and misuse it.

Taking care of expiration dates and certificate validity

If you are not using PKI Services on z/OS or a similar solution to manage certificates, you are running the risk of missing certificate expiration dates in your installation.

When a certificate expires without someone taking proper preparatory steps, it can cause significant disruption to your system or application availability and can have a large impact on the users of those systems or applications. We have seen customer situations, where the security administrator was called on emergency, because, for example, a Server certificate in RACF had expired in the middle of the night or over a weekend. You want to avoid situations like these. This can do serious damage to your business because it provokes downtime of the application.

Although RACF itself does not provide a mechanism to verify certificate expiration dates and provide warnings to a security administrator regarding certificate expiration, a new health check was added in 2.1 to help in this area. IBM HealthChecker for z/OS provides a check to make sure that certificate expiration dates are not missed. If you have implemented HealthChecker for z/OS, make sure this check is performed regularly and that its results are examined. *IBM Health Checker for z/OS V1R12 User" Guide, SC23-6843-00*, provides guidance for implementing the product and the relevant checks. The certificate expiration check offers the following measure:

- ▶ **RACF_CERTIFICATE_EXPIRATION** allows RACF to identify all certificates which have expired, identify all certificates which are going to expire within the next few days (number of days can be set as a parameter to the check), and ensures that the user has defined a proper baseline set of protections within the z/OS environment.

It is also useful to carefully review all the current expiration dates in RACF. PKI Services for z/OS and other certificate management solutions might provide fixed templates for certificates. Meaning that, for example, a standard user certificate for TN3270E might be issued for one year. Then it is not possible to issue such a certificate for a longer time period. *IBM Security zSecure* also provides templates for the creation of certificates since version 2.1. Using certificate template mechanisms provides an enhancement in security, as it becomes easier to adhere to policies for certificate creation. As described in “Renewing keys” on page 81, private keys are in some ways analogous to passwords. Therefore, providing users or servers with certificates that are valid for a long time is comparable to providing them with passwords that will never expire.

If you act as your own CA within your organization or plan to, you must pay close attention to the expiration dates of your CA certificates also. Keep in mind that no certificate can have a validity longer than the signing certificate. If your CA is valid for five years, you will not be able to sign certificates that are valid for more than five years. To avoid putting your certificate infrastructure at risk and to make sure that you can always sign certificates using your specified templates, make sure that the CA certificate is valid for at least double time of your user and server certificates. You must find a balance between operational requirements (certificates need to be created) and security requirements (you do not want to have keys that are active forever).

RACF notes on digital certificates creation, change, and activation

Activate and RACLIST the DIGTCERT class if you use digital certificates with applications that require high performance, such as applications that access IBM WebSphere Application Server. If the DIGTCERT class is not RACLISTed, digital certificates can still be used but performance might be impacted when applications that retrieve certificates from RACF must wait while RACF retrieves them from the RACF database rather than from virtual storage.

After creating a new digital certificate, refresh the DIGTCERT class by issuing the SETROPTS RACLIST(DIGTCERT) REFRESH command. If you do not refresh the RACLISTed DIGTCERT profiles, RACF will still use the new digital certificate. However, performance might be impacted because applications that retrieve certificates from RACF will wait while RACF retrieves the new certificate from the RACF database.

Restriction: Any RACLISTed digital certificates that you alter, re-add or delete will not reflect your changes until you refresh the DIGTCERT class. This is because RACF uses RACLISTed profiles before profiles in the RACF database. Therefore, to make your changes effective, refresh the DIGTCERT class.

In general, after running any of the RACDCERT commands that update certificates or key rings, if the DIGTCERT and DIGTRING classes are RACLISTed, you must issue the following command:

```
SETROPTS RACLIST(DIGTCERT DIGTRING) REFRESH
```

PKI Services on z/OS versus RACDCERT for certificate management

We have described the PKI Services for z/OS component in detail in 2.1.9, “PKI services” on page 65. PKI Services z/OS is one possible way of managing certificates on z/OS.

Nevertheless, the most common approach to certificate management on z/OS is the use of the RACF RACDCERT command. We want to give you a short guideline on how to decide which way of managing certificates is the right way for you.

PKI Services is the choice to make when it comes to managing vast amounts of certificates on the mainframe. It provides all of the functions required of a full certificate authority and has several capabilities and management functions that create an advantage over using RACDCERT in these situations. Also, PKI Services is compliant with the PKIX standard for PKIs. Especially if you are under regulation by government or similar, that might be requested anyway.

If you already deployed a PKI in your organization where users and administrators can create and retrieve certificates themselves, think about integrating PKI Services into your existing infrastructure for host-related certificates as a sub-CA. This way, you will have the possibility of enabling the PKI users with functions similar to what they are used to. PKI Services offer a highly customizable, web-based user interface.

We have also seen customers who were struggling to scale up their existing PKI solutions. They were not able of handling fast growing numbers of certificates with appropriate performance. In these situations, it makes great sense to evaluate a change in the infrastructure to PKI Services. This is especially the case when existing PKI solutions have been developed a long time ago where performance and scalability for huge amounts of certificates (in some industries, this can mean millions of certificates) has not been taken into account. These solutions might not scale up to the level you need. In modern network security environments, most of communication parties use public key cryptography, for example: network devices, routers, firewalls, security gateways, users, smart cards, PIN-code readers, credit card readers, servers, applications, and so on. The figures for certificates in these environments can reach high numbers very fast.

Several customer situations around the world have proven PKI Services to deliver the necessary performance in demanding environments.

For a usage comparison between PKI Services z/OS and RACDCERT certificate management, we provide an overview in Table 2-2.

Table 2-2 PKI Services versus RACDCERT for certificate management

RACDCERT	PKI Services z/OS
Smaller numbers of certificates to be generated	Need to generate a large number of certificates
You can keep track of expiration dates yourself ^a	Notification of certificate expiration should be sent automatically
Certificates are centrally generated and distributed from z/OS	You want to provide self-service solutions to users
You do not need a certificate revocation checking mechanism	Revoked certificates should be posted into a Certificate Revocation List (CRL)
You only need basic extensions in your certificates	You need extended support for extensions

a. Note: If you are using the z/OS Health Checker in z/OS 2.1, there is a check available that fulfils this requirement.

2.2.5 Guiding principles for IPSec

In 2.1.7, “IPSec” on page 54, we introduced various protocols that are collectively known as IPSec, including AH, ESP, and IKE. Next, we lay out guiding principles for the implementation of IPSec.

Comparison of dynamic and static tunnels for IPSec

We now describe the differences between dynamic mode tunneling and static tunneling with IPSec.

As you can see from Table 2-3, when static tunnels are used, secret keys must be defined locally at both endpoints of the communication. They are hardcoded and cannot be changed dynamically during an IPSec session. On the other hand, dynamic tunnels make use of the IKED to agree on shared secret keys between endpoints and to exchange keys dynamically during a communication session. This adds to the security of your IPSec implementation.

Table 2-3 Comparison of dynamic and static tunnels for IPSec

Dynamic tunnel mode	Static tunnel mode
SA attributes are agreed to through IKE negotiation	SA attributes are agreed through out-of-band communication and must be predefined locally
Cryptographic keys are established through IKE negotiation	Cryptographic keys are established through out-of-band communication and must be predefined locally
Provides authentication through pre-shared keys or digital signatures	Provides authentication through shared secret keys
Cryptographic keys are automatically refreshed in a nondisruptive manner	Cryptographic keys must be refreshed out-of-band and require the deactivation of the VPN to take effect

Comparison of protocol versions

The z/OS Communications Server supports IKEv2 in addition to IKEv1. IKEv2 has better performance and operational characteristics than IKEv1. Many government agencies expect the vendors who do business with them to use IKEv2 to establish secure communications with them. Consider the following differences:

- ▶ IKEv2 does not interoperate with IKEv1.
- ▶ IKEv1 supports AH in combination with ESP, but IKEv2 does not.
- ▶ The z/OS IKE daemon can support both IKEv1 and IKEv2 simultaneously.
- ▶ The z/OS IPSec policy file can contain both IKEv1 and IKEv2 policies.
- ▶ Each TCP/IP stack can support tunnels activated by IKEv1 and IKEv2.
- ▶ Security Association (SA) terminology differences:
 - IKEv1
 - Phase 1 SA (ISAKMP SA)
 - Phase 2 SA (IPSec SA)
 - IKEv2
 - Phase 1 SA (IKE SA)
 - Phase 2 SA (Child SA)

- ▶ The negotiation modes used by IKEv1 and IKEv2:
 - IKEv1
 - Main
 - Aggressive
 - Quick
 - IKEv2
 - Initial exchanges
 - Subsequent exchanges
- ▶ The encapsulation mode (tunnel or transport)
 - IKEv1: A negotiated attribute of the SA. Local value must match peer's value
 - IKEv2: Negotiated based on topology and user preference and SA. Local and peer can have different values

Advantages of IKEv2 over IKEv1

IKEv2 is a more current implementation of the IKE concepts. The protocol standard has several advantages over IKEv1:

- ▶ IKEv2 requires fewer network flows in most cases compared to IKEv1.
- ▶ IKEv2 can do rekeying without reauthentication.
- ▶ IKEv2 has built-in dead-peer detection.
- ▶ IKEv2 supports avoidance of duplicate or orphaned tunnels.
- ▶ IKEv2 supports ECDSA signature mode.
- ▶ All IKEv2 messages have an associated response, unlike IKEv1 which uses several notification messages that do not have an associated response
- ▶ IKEv2 has solved several other problems in IKEv1.

Comparison of dynamic tunnel authentication methods

As we described in “Internet Key Exchange protocol” on page 57, when using dynamic tunnels for IPSec you can either use a pre-shared secret for authentication of the peers of a communication or you can use digital signature mode making use of X.509 certificates and RSA or ECDSA. Table 2-4 compares the modes of authentication and lists advantages and disadvantages.

Table 2-4 Comparison of dynamic tunnel authentication methods

Authentication method	How authentication is performed	Advantages	Disadvantages
Pre-shared keys	By creating hashes over exchanged information	Simple	Shared secret must be distributed prior to IKE negotiations Key refresh must be done manually Identity depends on what is acceptable to peer platform (Some platforms permit IP address only.)
Digital signature (RSA or ECDSA)	By signing hashes over exchanged information	Can use IDs other than IP address Partner certificates do not need to be available before IKE negotiations Better security. Key refreshes are a natural part of an X.509 certificate infrastructure	Requires certificate operations Requires X.509 certificate infrastructure (PKI) and management. More complex configuration

2.2.6 OpenSSH on z/OS UNIX, z/OS dependant features implementation

As we have seen in 2.1.8, “OpenSSH on z/OS” on page 60 OpenSSH on z/OS provides some features which are specific to z/OS and make use of z/OS own security infrastructure. Let us now take a closer look on those features and describe the implementation and the benefits in more detail. We have seen several customers who are using OpenSSH on z/OS but were unaware that they are able to use some z/OS specific security aspects with OpenSSH. The guiding principle for us is to draw attention on those features.

Additional material: For more information about setting up OpenSSH z/OS specific features for z/OS, see *IBM Ported Tools for z/OS: OpenSSH User's Guide, SA23-2246-00*

z/OS specific configuration files for OpenSSH

z/OS obtains z/OS-specific system-wide OpenSSH client configuration data only from the `/etc/ssh/zos_ssh_config` configuration file. It contains sections separated by *Host* specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

z/OS obtains z/OS-specific per-user client configuration data in the following order:

1. User-specific client options from either of these options:
 - The command-line specification using the `-o` option of the `scp`, `sftp`, or `ssh` command.
 - The file specified with the `_ZOS_USER_SSH_CONFIG` variable. The default is `~/.ssh/zos_user_ssh_config`.
2. System-wide client options from the `/etc/ssh/zos_ssh_config` file.

z/OS obtains z/OS-specific daemon configuration data in the following order:

1. Command-line specification using the `sshd -o` option.
2. A configuration file specified with the environment `_ZOS_SSHD_CONFIG` variable. The default is `/etc/ssh/zos_sshd_config`. For each keyword, the first obtained value is used.

Keywords: z/OS-specific keywords cannot be specified in the `sshd_config` and `ssh_config` configuration files, such as the system-wide configuration file or the user-defined configuration file that is specified with the `sshd -f` option.

RACF key ring support for OpenSSH on z/OS

Using UNIX files to store the keys is the common method supported on all OpenSSH implementations. Consider what other OpenSSH hosts you will be communicating with; that is, are they z/OS or non-z/OS? Also consider whether the z/OS systems are using key rings. Nevertheless, an intermix between z/OS key rings and UNIX files is possible.

Key rings provide commonality with other z/OS products that store keys in the security product. They can be real or virtual key rings. To use SAF key rings, you must have RACF or an alternative SAF-compliant security product. Appropriate authority must also be given to user IDs to manage the key rings.

Key rings might provide additional security in your environment. If you do not have a sophisticated UNIX security environment in your installation, authentication keys for OpenSSH might be in danger of exposure. There is no strong mechanism of control whether or not OpenSSH administrators make use of key-based authentication rather than password authentication. The storage of OpenSSH keys for authentication in RACF key rings, rather than UNIX files, might give your administration team more control over the use of keys.

SMF records for OpenSSH on z/OS

Providing an audit trail for system logon and data transfer from and to the system are dictated by common regulation standards across all industries. As OpenSSH on z/OS is a common way of accessing the system and transferring files from and to the system, especially for users in UNIX System Services, the audit trail needs to be available here also. SMF is the standard way on z/OS systems to provide audit information. Therefore, the recommendation is to use the SMF records provided by OpenSSH on z/OS to gather the necessary information.

OpenSSH collects SMF Type 119 records for file transfer activity and login failure information. You can control the collection of these records by using the configuration keywords *ClientSMF* and *ServerSMF* in z/OS-specific client and daemon configuration files, respectively. These keywords also indicate whether system-wide SMF record exit IEFU83 or IEFU84 receives control. The following list provides an overview of the records created:

► SMF server transfer completion record (Type 119 - Subtype 96)

The server transfer completion records are collected when the **sftp-server** (regular or *internal-sftp*) or the server side of **scp** completes processing of one of the following file transfer subcommands:

- Creating, uploading, downloading, renaming or removing files
- Creating and removing directories
- Changing the file permissions, UIDs, or GIDs
- Creating symbolic links

► SMF client transfer completion record (Type 119 - Subtype 97)

The client transfer completion records are collected when the client side of **sftp** or **scp** completes processing of one of the following file transfer operations:

- Uploading files
- Downloading files

► SMF login failure record (Type 119 - Subtype 98)

Login failure records are collected after each unsuccessful attempt to log in to the **sshd** daemon. A login failure record is collected for each authentication method and attempt that fails. A login failure reason code within the SMF record provides information about the cause of the login failure. Only failures during user authentication are collected with the following exception: Records are not collected for a *none* authentication failure if it is the first authentication method attempted.

Multilevel Security in OpenSSH for z/OS

Some installations might be obligated to implement multilevel security within z/OS RACF. This is mostly required for high security environments, that are operated under a certain security standard, for example the Common Criteria Operating System Protection Profile (OSPP).

The OpenSSH on z/OS daemon (**sshd**) can be used on a multilevel-secure system to control a user's security label at login. Review z/OS Planning for Multilevel Security and the Common Criteria before using the daemon on a multilevel-secure system.

The OpenSSH daemon will attempt to derive a security label from the user's port of entry, as defined in a SERVAUTH NETACCESS profile. To successfully log in to a multilevel-secure system, the login user ID must be permitted to the security label defined in the NETACCESS profile for the client IP address. These checks are performed for any user invoking **ssh**, **scp**, or **sftp** to perform remote operations on the multilevel-secure system. For more information about NETACCESS profiles and running daemons in a multilevel-secure environment, see *z/OS Communications Server: IP Configuration Guide, SC27-3650-00*.



TCP/IP security

In this chapter we provide a security concepts and architecture overview for TCP/IP on System z. Some concepts of the applications and components listed here will be provided as well. Then we describe some guiding principles for configuring TCP/IP security.

This chapter includes the following sections:

- ▶ 3.1, “Introduction” on page 90
- ▶ 3.2, “Sockets and APIs” on page 91
- ▶ 3.3, “Telnet Server” on page 93
- ▶ 3.4, “FTP” on page 98
- ▶ 3.5, “InetD, the Internet daemon” on page 102
- ▶ 3.6, “Virtual IP addressing” on page 105
- ▶ 3.7, “z/OS IP filtering” on page 107
- ▶ 3.8, “IPSec” on page 111
- ▶ 3.9, “z/OS Intrusion Detection Services” on page 116
- ▶ 3.10, “IP resource security” on page 125

Important: A more general overview of TCP/IP related topics is addressed in Chapter 1, “Mainframe network concepts and functions” on page 1. Make sure that you are familiar with these concepts.

3.1 Introduction

The IT backbone of every organization is a strong IP network infrastructure. Securing your z/OS IP network resources is key to ensure smooth business operations. In this chapter, we describe the particular controls that you will be able to use as systems programmer to protect the IT crown jewel of your organization.

3.1.1 IP network design

When discussing IP security, the first step is to address the overall network design. Because System z usually contains multiple components of an enterprise service flow, there are particular challenges on how to design this part of the network.

The most common type of network design is a three-tiered network design. This design consists of a *presentation layer*, an *application layer*, and a *database layer*. The presentation layer usually resides within a DMZ and can be considered the front end to your applications. The other two layers provide an interesting challenge to System z network engineers, because both of these layers usually reside within System z.

It is important that you review your corporate policies to see whether this configuration is allowed, or if an exception needs to be made. There are methods available, such as designating some System z LPARs as application LPARs and others as database LPARs, to segregate the network traffic to meet this common IT architectural standard. Unfortunately there is no silver bullet that can be offered without truly understanding the applications in your environment. However, there are other security controls that we describe in this chapter, such as SAF controls, which can be used to mitigate security issues. We also describe other design choices that deal with network architecture.

3.1.2 System z in a DMZ

Normally, it is not a preferred practice to deploy a System z resource within the network DMZ. It is better to create a bastion host, even if it just acts as a proxy. This approach can front-end the network traffic to ensure that you are not wasting any processor resources needlessly in deflecting attacks. This approach can also limit the exposure of other services if your System z is placed within an enterprise network zone. If it is deemed necessary to place an LPAR inside the DMZ it should be segmented from other System z resources, and every network packet should have to traverse a third-party firewall and intrusion prevention device.

3.1.3 Mixing environments

In many deployments different stages of systems, such as *production*, *test*, and *development*, are placed together within the same network hierarchy. This is generally considered a bad practice because test systems have much lower levels of security placed on them compared to systems in development or production.

This could potentially allow an attacker to access the test system and gain a stronghold behind the network defenses that you have put in place to protect access to your production and development systems. Breaking these areas apart allows for stronger security controls to be placed on your different environments, which in turn allows you to better adhere to your organization's security policies.

We also advise that you do not place distributed servers within the same local network segment where the Systems z LPARs reside. Because distributed servers have been known in the past to become susceptible to malicious attacks more easily, it is not wise to have them deployed on the same local network segments that your System z.

3.1.4 HiperSockets

IBM HiperSockets technology is an internal memory buffer within the System z that is used as *internal Queued Input/Output* (iQDIO). This concept can potentially produce network throughput at memory speeds to pass traffic between virtual servers on the same hardware platform. This is not just limited to z/OS, but is also available for Linux on System z and z/VM. HiperSockets is an elegant solution for transferring data securely between virtual servers without the need for encryption because this memory buffer cannot be sniffed or compromised.

Preferred practice: We advise that communication between LPARs on the same physical System z footprint should rely on HiperSockets communications.

3.2 Sockets and APIs

In a TCP/IP based network applications rely on two major concepts to communicate with each other, Sockets and APIs.

► Sockets

A *socket* is an endpoint for communication able to be named and addressed in a network. From the perspective of the application program, it is a resource allocated by the address space; it is represented by an integer called the *socket descriptor*.

The socket interface was designed to provide applications a network interface that hides the details of the physical network. The interface is differentiated by the different services provided:

- Stream
- Datagram
- Raw sockets

Each interface defines a separate service available to applications. A socket uniquely identifies the endpoint of a communication link between two application ports.

A port represents an application process on a TCP/IP host, but the port number itself does not indicate the protocol being used: TCP, UDP, or IP. The application process might use the same port number for TCP or UDP protocols.

To uniquely identify the destination of an IP packet arriving over the network, you must extend the port principle with information about the protocol used and the IP address of the network interface; this information is called a socket. A socket has three parts, *protocol, local-address, local-port*.

Figure 3-1 illustrates the concept of a socket.

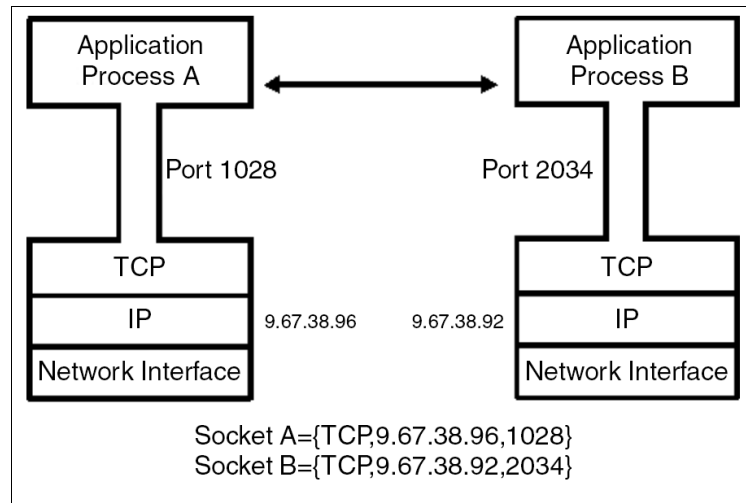


Figure 3-1 Socket concept

The term *association* is used to specify completely the two processes that comprise a connection as this example shows:

(protocol,local-address,local-port,foreign-address,foreign-port)

The terms *socket* and *port* are sometimes used as synonyms, but the terms *port number* and *socket address* are not like one another. A port number is one of the three parts of a socket address, and it can be represented by a single number, such as 1028, but a socket address can be represented by tcp,myhostname,1028.

A *socket descriptor* (sometimes referred to as a *socket number*) is a binary integer that acts as an index to a table of sockets; the sockets are currently allocated to a given process. A socket descriptor represents the socket, but is not the socket itself.

► APIs

z/OS Communications Server provides several application programming interfaces (APIs) to access TCP/IP sockets. These APIs can be used in either or both integrated and common INET PFS configurations. In a common INET PFS configuration, however, they function differently from z/OS UNIX APIs. In this type of configuration, the z/OS Communications Server APIs always bind to a single PFS transport provider, and the transport provider must be the TCP/IP stack provided by z/OS Communications Server.

The following TCP/IP socket APIs are included in z/OS Communications Server:

– Pascal API

The Pascal APIs enable you to develop TCP/IP applications in Pascal language. Supported environments are normal MVS address spaces. The Pascal programming interface is based on Pascal procedures and functions that implement conceptually the same functions as the C socket interface. The Pascal routines, however, have different names than the C socket calls. Unlike the other APIs, the Pascal API does not interface directly with the Logical File System (LFS). It uses an internal interface to communicate with the TCP/IP protocol stack. The Pascal API only supports AF_INET.

- CICS sockets

The CICS socket interface enables you to write CICS applications that act as clients or servers in a TCP/IP-based network. Applications can be written in the C language, using the C sockets programming, or they can be written in COBOL, PL/I or assembler, using the Sockets Extended programming interface. CICS sockets only support AF_INET.

- C sockets

The C socket interface supports socket function calls that can be invoked from C programs. However, note that for C application development, IBM recommends the use of the UNIX C socket interface. These programs can be ported between MVS and most UNIX environments relatively easily if the program does not use any other MVS specific services. C sockets only support AF_INET.

- IMS sockets

The Information Management System (IMS) IPv4 socket interface supports client/server applications in which one part of the application executes on a TCP/IP-connected host and the other part executes as an IMS application program. The IMS sockets API supports AF_INET.

- Sockets Extended macro API

The Sockets Extended macro API is a generalized assembler macro-based interface to sockets programming. It includes extensions to the socket programming interface, such as support for asynchronous processing on most sockets function calls. The Sockets Extended macro API supports AF_INET and AF_INET6.

- Sockets Extended Call Instruction API

The Sockets Extended Call Instruction API is a generalized call-based, high-level language interface to sockets programming. The functions implemented in this call interface resemble the C-sockets implementation, with some extensions similar to the sockets extended macro interface. The Sockets Extended Call Instruction API supports AF_INET and AF_INET6.

- REXX sockets

The REXX sockets programming interface implements facilities for socket communication directly from REXX programs by using an address rxsocket function. REXX socket programs can execute in TSO, online, or batch. The REXX sockets programming interface supports AF_INET and AF_INET6

3.3 Telnet Server

Telnet is a terminal emulation protocol that enables you to log on to a remote system as though you were directly connected to it. Telnet enables users to have access to applications running on that system. The TN3270 Telnet Server provides access to IBM VTAM SNA applications on the z/OS host using 3270 or Linemode protocol.

Traditionally, non-mainframe ASCII-based platforms have used the Telnet emulation protocol for achieving this connectivity. However, because the mainframe is an EBCDIC-based platform with special 3270 terminal-type data stream requirements for its connected terminals, the TN3270 function was developed to support mainframe data streams.

Telnet is provided on z/OS by Telnet 3270 (TN3270), which is a terminal emulation technology that supports access to SNA applications on mainframe computers using TCP/IP (Telnet) protocols.

A z/OS Communications Server TN3270E Telnet Server can be implemented in your mainframe environment to allow TN3270 clients to access SNA applications using TCP/IP (Telnet) protocols.

As illustrated in Figure 3-2, TN3270E Telnet Server is one of the standard applications provided with the z/OS Communications Server. It uses z/OS UNIX socket application programming interfaces (APIs), a logical file system (LFS) and a physical file system (PFS) to transfer data from application layers to transport layers, and to manage incoming client requests.

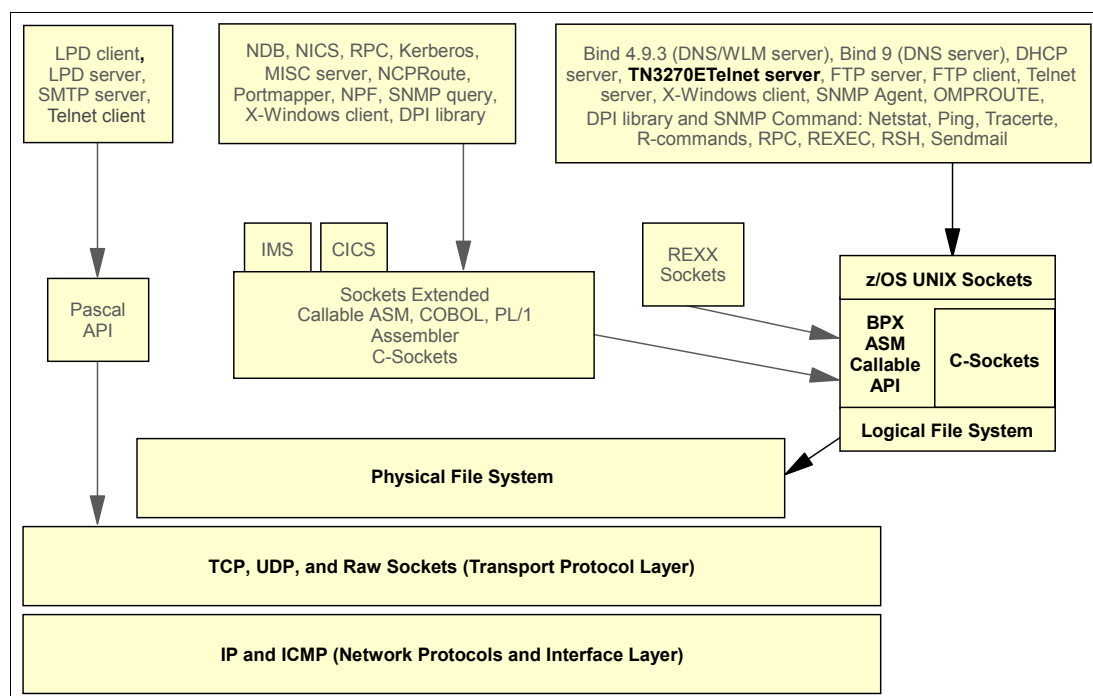


Figure 3-2 z/OS TN3270E Telnet Server application services

3.3.1 Security concepts and architecture

Telnet is a protocol used worldwide for client access in the TCP/IP environment. To reduce the liability of your environment you must secure your Telnet applications. The traditional Telnet TN3270E differ in two main areas:

- ▶ 3270 terminal emulation uses block mode rather than line mode.
- ▶ 3270 terminal emulation uses the EBCDIC character set rather than the ASCII set.

Traditionally, non-mainframe ASCII-based platforms have used the Telnet emulation protocol for achieving this connectivity. However, because the mainframe is an EBCDIC-based platform with special 3270 terminal type data stream requirements for its connected terminals, the TN3270 function was developed to support mainframe data streams. The TN3270E Telnet Server has implemented these special terminal data stream requirements for clients using the 3270 emulation protocol.

What's the “E” about? The meaning of the *E* as used in TN3270E and in IBM terminal devices, such as the IBM-3279-2-E can be confusing. In both cases, the *E* represents *extended* capabilities. However, there is no correlation between the extended capabilities of TN3270E and those of an IBM-3279-2-E display station.

327x device types that end in *E* are terminals that support extended field attributes, such as color and highlighting. TN3270E server support includes several important enhanced capabilities over TN3270E.

Also, the z/OS TN3270E Server supports RFC 1647 TN3270 Enhancements (July 1994) and RFC 2355 TN3270 Enhancements (June 1998), but not the RFC 1646 TN3270 Extension for LU-name and Printer Selection (July 1994). If you want to migrate from a channel-attached router or Communications Server for IBM AIX, Linux, and Windows to z/OS TN3270E server, the older version of the TN3270 emulator software might need to be replaced.

The TN3270 server is enabled for both SSL and AT-TLS, and the data path in the IP network to Telnet is protected using the SSL protocol. Figure 3-3 provides a TN3270E Telnet Server security overview.

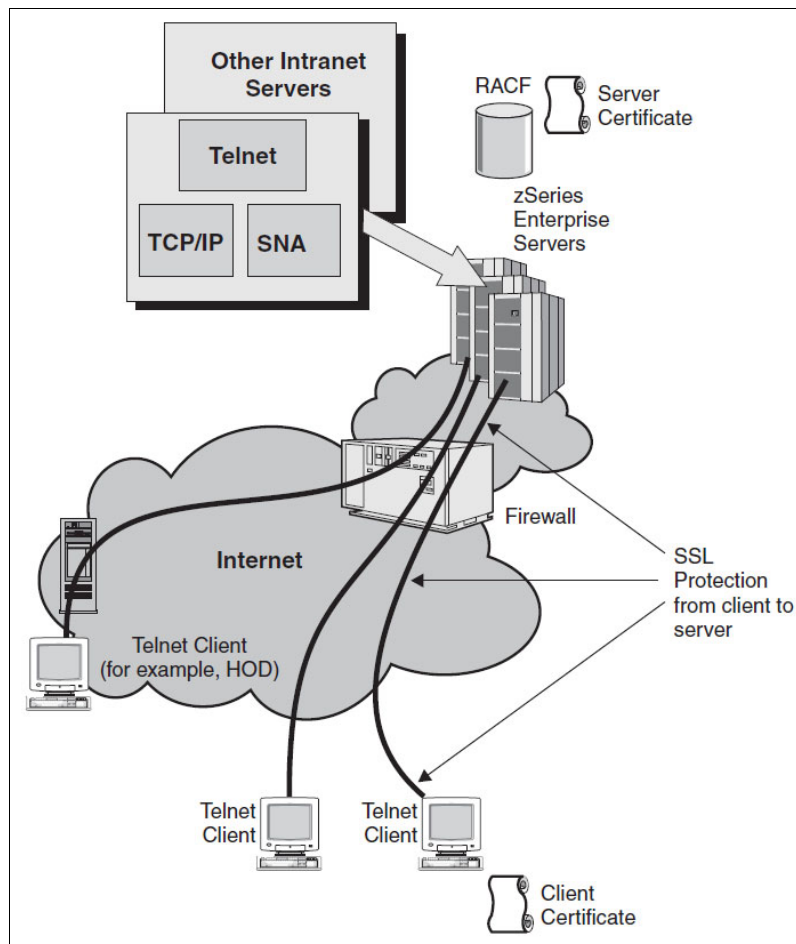


Figure 3-3 TN3270E Telnet Server security overview

The TN3270E Telnet Server support provides several extensions for RACF-based access control to Telnet. These extensions can prevent a client from seeing the USSMSG (log on screen) unless the client is authorized.

Preferred practice: To configure the TN3270E Telnet Server extension to prevent a client from seeing the USSMSG (log on screen) define the client certificate to RACF using RACF digital certificate services. The first level of authorization checking verifies that the RACF user ID presented by the client certificate is defined to RACF. The next level of authorization requires that this RACF user ID be permitted to access the Telnet port. The Telnet port is represented as a RACF resource using the SERVAUTH class.

The TN3270E Telnet Server security can be enhanced using both multiple ports support and single port support:

► Multiple port support

TN3270E Telnet Server provides multiple port support. It can be used to enable the combination of secure and non-secure traffic. To use multiple port support, you define separate ports; one port is dedicated to non-secure traffic and another port is dedicated to secure traffic. Ports with the designation SECUREPORT or TTLSPORT can be secure. Intranet clients are not required to be secure. Intranet clients connect to the BASIC port (port 23, as shown in Figure 3-4). All clients connecting from the Internet are required to be secure. These clients use the SECUREPORT (port 1023, as shown in Figure 3-4). Packet filtering is used at the firewall that separates the intranet and Internet to control access to the Telnet ports.

To prevent Internet access to the BASIC port, port 23 is blocked at the firewall. The SECUREPORT, port 1023, is permitted at the firewall. In this scenario, the best security is achieved when SSL client authentication with the Telnet RACF extensions is used. This support ensures that the client has the authority to attempt to log on to SNA applications through Telnet. Regardless of the method of authentication used, the SNA application should identify and authenticate the user using the RACF before any application access is granted. If you are using SSL encryption services, the used ID and password are encrypted.

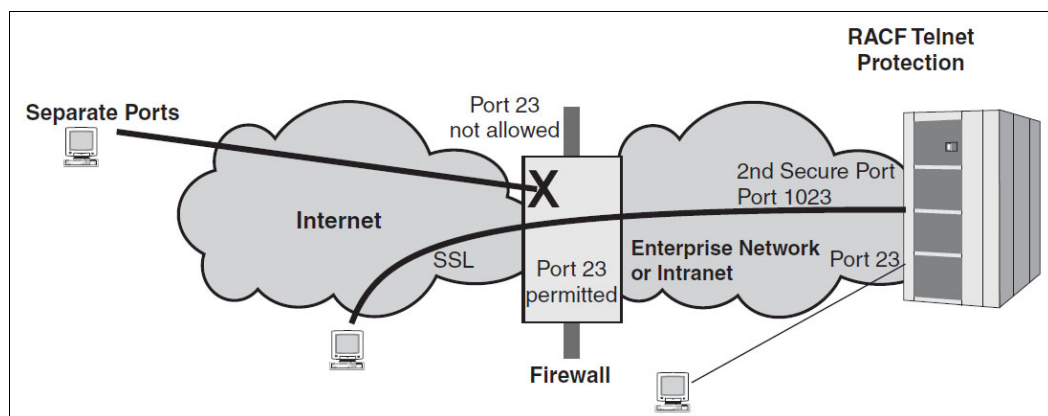


Figure 3-4 Using multiple Telnet ports to separate secure and non-secure traffic

You can enhance security even more on remote access from the Internet to SNA applications combining IPSec and Telnet security. This scenario is shown on Figure 3-5. IPSec AH protocol is used for authentication between the user's PC and the firewall. The firewall is open for port 1023 for traffic that is authenticated with only IPSec and discards the traffic that cannot be authenticated by IPSec. The additional security provided by IPSec protects the System z server from unauthorized access attempts and denial of service attacks by hosts outside the VPN.

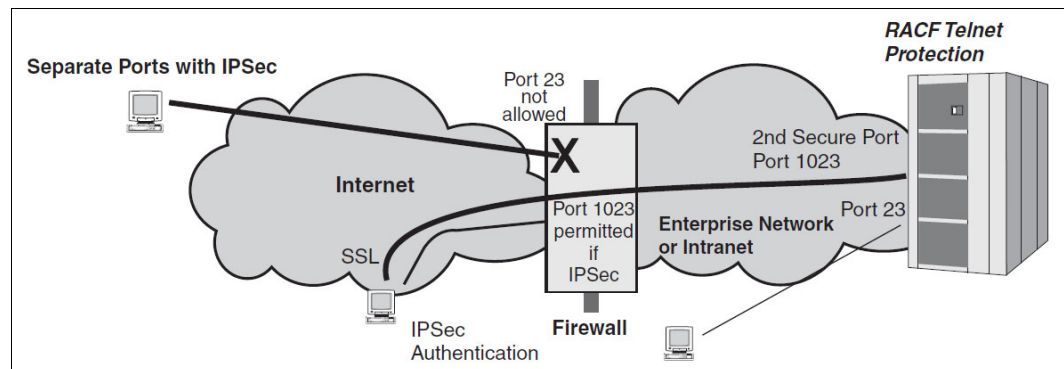


Figure 3-5 Combining Telnet security with IPSec client-to-firewall authentication

► Single port support

A single port can be used to support a mix of secure and non-secure traffic. The port has the designation SECUREPORT or TTLSPORT. To support the configuration of various security policies for a single port, the SECUREPORT or TTLSPORT designation indicates that the port can use TLS/SSL, but the port does not have to use TLS/SSL.

Telnet supports both negotiated and non-negotiated TLS/SSL. Negotiated TLS/SSL is an IETF-defined extension to the TN3270 protocol. With negotiated TLS/SSL, the decision to use TLS/SSL for a connection is based on the outcome of a negotiation between the Telnet client and server using TN3270 protocols. This negotiation is performed after the Telnet connection is established, and if TLS/SSL is negotiated, the TLS/SSL handshake is performed. With non-negotiated TLS/SSL, a TLS/SSL handshake is required immediately after the connection is established. A single port can concurrently use both negotiated and non-negotiated TLS/SSL connections.

Figure 3-6 on page 98 shows a single Telnet port that allows a mix of secure and non-secure traffic. Intranet clients are not required to be secure. All clients connecting from the Internet are required to use SSL. Both intranet and Internet clients connect to the port designated as SECUREPORT (port 23 in this example).

In this scenario, IPSec AH protocol is used for authentication between the user's PC and the firewall. The firewall is open for port 23 for traffic that is authenticated with only IPSec. The firewall discards traffic for port 23 that IPSec cannot authenticate. In this scenario, packet filtering without IPSec cannot be used at the firewall that separates the intranet and the Internet to control access on the basis of port, because only one port is used. Without IPSec AH, all access control checks are deferred to Telnet. The additional security provided by IPSec at the firewall protects the System z server from unauthorized access attempts and denial of service attacks by hosts outside the VPN.

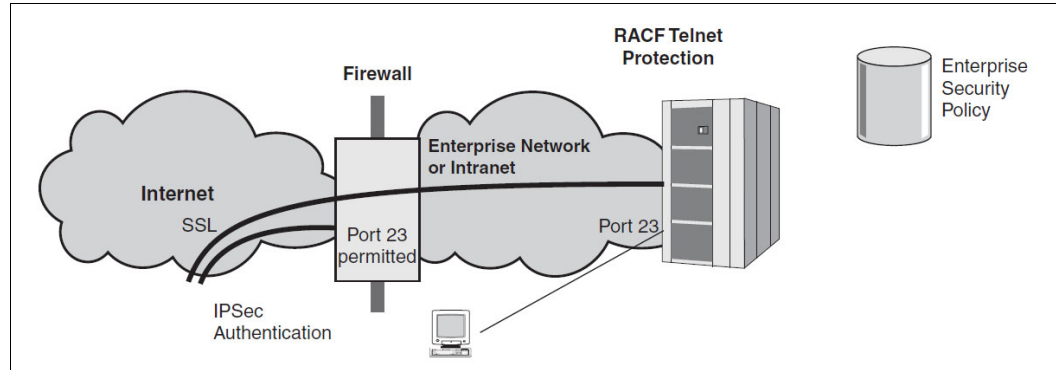


Figure 3-6 Secure and non-secure traffic using a single Telnet port

3.4 FTP

File Transfer Protocol (FTP) allows you to transfer files between any computers (mainframe, midrange systems, PC servers, and desktop systems) that support the TCP/IP-standard FTP protocols. FTP is the name of both the application and the protocol that allows you to transfer files over in a TCP/IP network. The FTP protocol uses a client/server model and the z/OS Communications Server ships with both FTP server (FTPD) and an FTP client.

FTP provides a fast and reliable method to transfer files in a TCP/IP network. FTP also allows for communication between platforms that have different character encoding and file structures by hiding such details from the user. With a few simple FTP commands, you can easily transfer a file from one platform to another regardless of whether the two platforms are similar. FTP is one of the standard applications provided with the z/OS Communications Server, which are depicted in Figure 3-7.

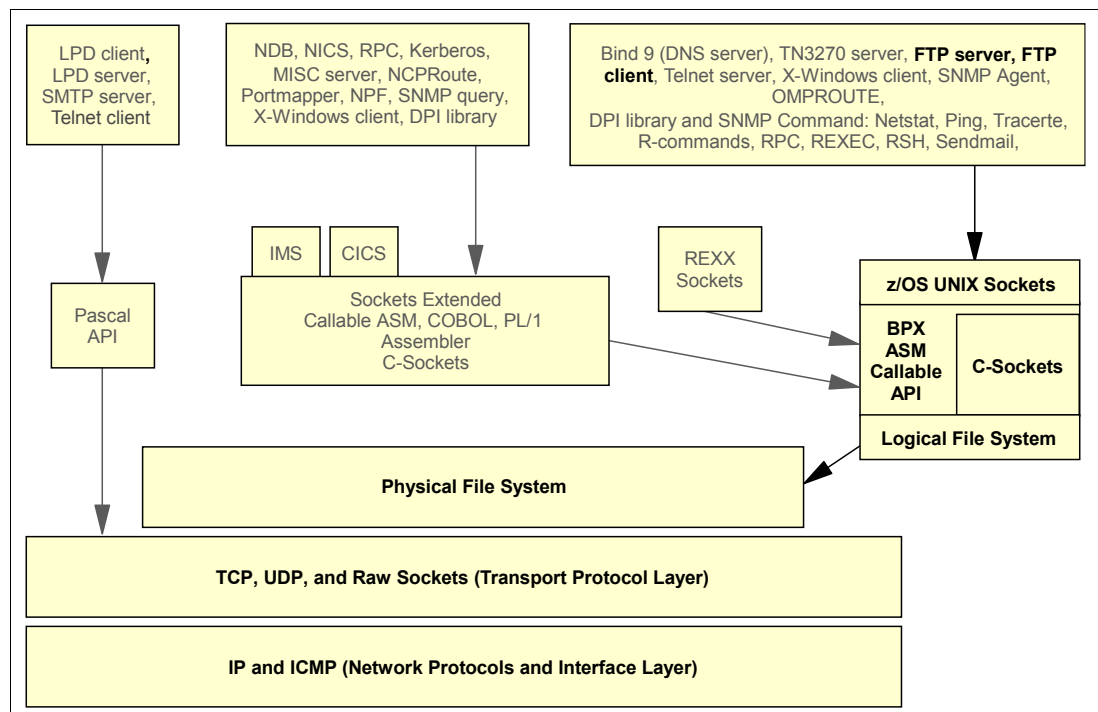


Figure 3-7 z/OS FTP client/server application services

3.4.1 Security concepts and architecture

System administrators can implement security precautions to protect data being transferred from one client to another. Rules can be applied to client devices, server devices, application platforms, and network firewalls to administer the security policies in the organization. This section focuses on the security measures that can be applied to the z/OS FTP client and server applications.

Security exposures can exist in an FTP environment, so it is imperative to analyze and correct them using available tools. The FTP application relies on the use of an external security manager, such as RACF, for certain levels of security. The client and server environments provide built-in security functions for additional security. We identify a few of the common security exposures and tools that can be used to address the issues. The major emphasis is on Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Application Transparent Transport Layer Security (AT-TLS) features supported by the z/OS FTP application.

Throughout this section, we define the term *secure FTP* with the following statements:

- ▶ FTP data transmissions can be secured by requiring the client to connect to the server indirectly by passing through an FTP SOCKS proxy server that can control the connections and provide an extra level of security itself on the path toward the final destination server.
- ▶ FTP clients can allow or require server authentication whereby the server identifies itself to the client by passing a digital certificate to the client, which verifies the certificate against pre-established criteria. Any data exchanged between the client and server can then be encrypted using encryption keys established at connection time.
- ▶ FTP servers can allow or require the same server authentication process. Data encryption is usually the intent of this process, although not always required.
- ▶ FTP servers can require client authentication whereby the client identifies itself to the server by passing a digital certificate to the server who verifies the certificate against pre-established criteria. The certificate information can be used by the server to provide secure application logons on behalf of the client, thus avoiding the exchanges of user ID and password in clear text.

To get a better view of the FTP security architecture, it helps to understand how FTP security works.

SOCKS proxy protocols enable an FTP client to access a remote FTP server, protected by a blocking mechanism, that is otherwise unreachable. The client does not have direct network access to the final destination server, but the intermediate (proxy) server does have the necessary access. Therefore the client, having access to the proxy server, can connect to the proxy by using a special SOCKS protocol, pass through the proxy, and thereby gain access to the intended final destination.

Preferred practice: Although it can be a little complicated, consider configuring your FTP server to use a SOCKS proxy to enhance the security of your overall FTP environment.

Clients and servers can identify (authenticate) each other by using digital certificates, and can encrypt data exchanges by using encryption keys obtained from these certificates.

In addition to native TLS support, the FTP server and client can use AT-TLS to manage TLS security. TLS managed by AT-TLS (TLSMECHANISM ATTLS) supports more security functions than TLS managed natively by the FTP (TLSMECHANISM FTP). Be aware of these AT-TLS capabilities and requirements when planning the preferred AT-TLS support for FTP:

- ▶ Specify the label of the certificate to use for authentication rather than using the default.
- ▶ Support SSL session key refresh.
- ▶ Support SSL sysplex session ID caching.
- ▶ Trace decrypted SSL data for FTP in a data trace.
- ▶ Receive more detailed diagnostic messages in syslogd.
- ▶ There are no restrictions for the FTP ATTLS function.
- ▶ Policy agent must be active for FTP ATTLS to work.
- ▶ TLS security defined natively to FTP will continue to be available in addition to AT-TLS.

FTP security can be applied by implementing restricted access to FTP servers with firewalls, and then requiring the use of a SOCKS proxy server to control client access to the intended destination server.

Digital certificates can be used for client and server authentication and to enable the use of data encryption.

If you already have TLS implemented for an existing instance of the FTP server, it is easy to migrate from TLS to AT-TLS support. AT-TLS support is implemented by moving most of the TLS definitions from the FTP server's FTP.DATA file into the policy agent's AT-TLS configuration profile section. The FTP server is defined as an AT-TLS controlling application to the policy agent, and can retain its control of the secure relationship with the client. The AT-TLS implementation provides more flexibility and functions than basic TLS, and is therefore the preferred method of implementing digital certificate support for the FTP server. The IBM Configuration Assistant GUI is used to create the appropriate policy agent statements for AT-TLS.

If the FTP client must pass through a SOCKS proxy, the client must use the SOCKS protocol to successfully navigate through the proxy server. The client must be able to determine which servers it must contact using the SOCKS protocol. These servers are specified in a SOCKS configuration file that the client reads to make this determination. The SOCKSCONFIGFILE statement in the client's FTP.DATA file is used to point the client to the configuration file where these servers are identified.

Some dependencies must be covered to use FTP with a SOCKS proxy:

- ▶ The SOCKS protocol is supported for IPv4 only, not IPv6.
- ▶ The SOCKSCONFIGFILE is applicable to the client only: the server ignores the statement.
- ▶ If the SOCKSCONFIGFILE is not present in the client's FTP.DATA file, the client does not use SOCKS to contact any servers.
- ▶ The configuration file can be an HFS file or an MVS data set.

Using the FTP client with SOCKS allows users to contact FTP servers that are protected by a SOCKS firewall that are otherwise unreachable. For more information and guidance on how to configure FTP with SOCKS proxy, check *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-8099.

In addition to these methods you can also configure the basic z/OS FTP server to support native TLS-based security by enabling various RACF profiles and then authenticate and encrypt the FTP sessions. Both the FTP server and client in the z/OS Communications Server support TLS and Kerberos security. FTP can also be made secure with z/OS Communications Server AT-TLS or IP Security (IPSec).

Secure FTP versus sftp: The SSL/TLS implementation of FTP is known as *secure FTP* and invokes security in z/OS Communications Server using z/OS System SSL. It is based upon extensions to the base FTP RFC 959:

- ▶ RFC 2246, “The TLS Protocol Version 1”
- ▶ Internet draft RFC, “On Securing FTP with TLS (draft 05)”
- ▶ RFC 2228, “FTP Security Extensions”
- ▶ RFC 4217, “Securing FTP with TLS”

You need to understand the difference between *secure FTP* and the OpenShell procedure known as *sftp*. The *sftp* procedure operates over an encrypted Secure Shell (SSH) transport and does not use FTP protocols as described in RFC959 and its related RFCs.

By default, any valid z/OS user ID can log in to the FTP application.

Preferred practice: For security purposes, you might want to allow only certain user IDs to log in to FTP on a certain host. z/OS FTP currently provides the following methods to allow users to log in to FTP:

- ▶ Set up the FTP server for TLS level 3 authentication when sessions are secured with TLS.
- ▶ Code and install an FTCHKPWD exit routine that screens the user IDs that are allowed to log in to FTP.
- ▶ Code VERIFYUSER TRUE in the server's FTP.DATA and use RACF profile to protect the FTP server port.

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing, SG24-7996 explains how to set these configurations.

3.5 InetD, the Internet daemon

The Internet daemon (InetD), also known as the *Internet super daemon*, is an application that listens for connection requests on behalf of other applications. By handling the initial connection process, InetD passes the connection to the application associated with the targeted port.

InetD is one of the standard applications provided with the z/OS Communications Server. It is a generic listener that can be used by any server that does not have its own listener.

Figure 3-8 shows the relationship between InetD and its supported applications.

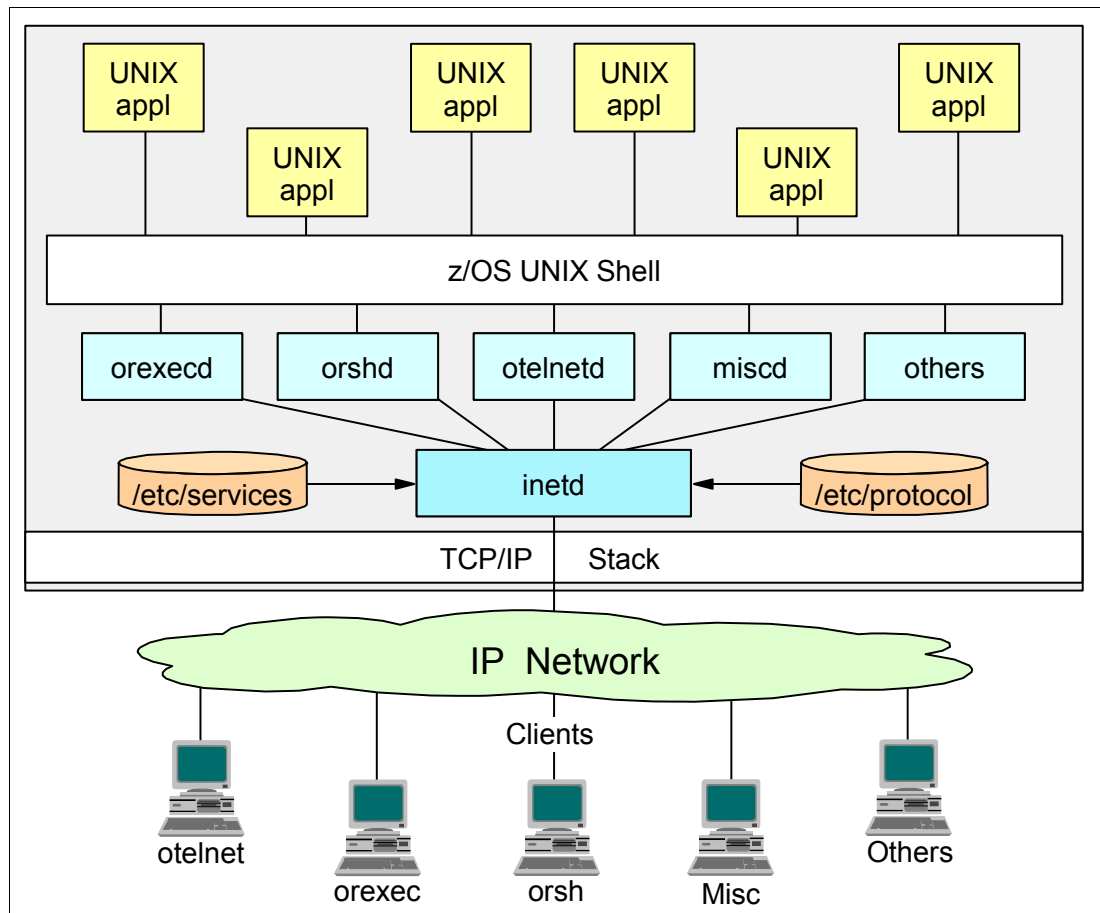


Figure 3-8 InetD and its supported applications

The InetD servers provide access to the z/OS UNIX shell using **otelnetd**, **rexecd**, or **rshd**, allowing you to then run other UNIX commands and applications from there. The z/OS Communications Server includes three applications and several internal services that require InetD, as listed on Table 3-1 on page 103.

Table 3-1 Applications that use InetD

Application	Description
z/OS UNIX Telnet server	See section 4.3 3.3, "Telnet Server" on page 93
z/OS UNIX REXEC Server	Remote Execution Protocol
z/OS UNIX RSH server	Remote Shell Protocol
sendmail and popper mail servers	Mail servers available on z/OS Communication Server
echo	Repeats any data received back to the sender
discard	Throws away any received data
chargen	Sends predefined or random data and discards any received data
daytime	Sends the current date and time in user readable form
time	Sends the current date and time in machine-readable form
popper	Post Office Protocol 3 (POP3) mail delivery agent

3.5.1 Security concepts and architecture

InetD calls `fork()` to run in the background and disassociates itself from the controlling terminal. z/OS UNIX appends a 1 to the job name, provided the job name is fewer than eight characters. For example, if InetD is started with job name InetD, after the application forks the job name is INETD1. The correct job name of InetD is important to note because the TCPIP.PROFILE data set should reserve ports for InetD using the job name after InetD forks.

InetD can accept two command line parameters:

- ▶ **-d**
- ▶ A file name

Both parameters are optional. If **-d** is specified, InetD does not `fork()` at start and all error messages are written to `STDERR`. If a file name is specified, then InetD uses the specified file as the configuration file rather than the default `/etc/inetd.conf`.

Restriction: Only one Inet daemon can be run in one MVS image. The process file that the InetD uses is `/etc/inetd.pid`. You cannot change this file to another file name using either shell commands or environment variables, and it can be used only by one InetD at the same time.

InetD reduces system load by invoking other daemons only when they are needed and by providing several simple Internet services internally without invoking other daemons. Some applications depend on InetD to provide a listener, and those applications cannot be started without InetD.

InetD uses a configuration file in the z/OS UNIX file system to determine for which services InetD will listen. A sample configuration file can be found in the `/samples/inetd.conf` file. The format of the file is:

```
<service> <socket type> <protocol> <wait/nowait> <user> <application> <arguments>
```

All parameters except the last one (arguments) are required on each line. Table 3-2 shows a brief description of each parameter.

Table 3-2 Configuration file parameters

Parameter	Description
service	Name of the Internet service. This name must match an entry in /etc/services. By default, InetD assumes you want to listen on all IP addresses. However, you can optionally specify the service parameter in the format of IP_address:service_name to force a particular service to listen on a particular IP address.
socket type	Options are stream or dgram. Applications that use the TCP protocol are stream applications. Applications using UDP are dgram.
protocol	The IP protocol used by the application. The options are TCP, UDP, TCP4, TCP6, UDP4, or UDP6. If TCP6 or UDP6 is specified then the socket will support IPv6. You can also optionally specify the maximum receive buffer in the format of protocol,sndbuf= <i>n</i> , where <i>n</i> is the number of users.
wait/nowait	Wait indicates the server is single threaded and the application will issue an accept() API call itself and process connections one at a time. Nowait indicates the application is multi-threaded. In nowait mode, InetD issues the accept() API call and passes a connected socket to the application. The application in nowait mode can process multiple connections at a time. You can also optionally specify the maximum number of simultaneous users allowed by the application by using the format nowait. <i>n</i> or wait. <i>n</i> , where <i>n</i> is the number of users.
user	The user ID the application should run under.
application	The full path to the executable file for the application InetD should start.
arguments	Up to 20 optional arguments that can be passed to the application.

Preferred practice: The INET daemon configuration file contains all the information regarding the applications running on its behalf. Make sure *not to grant write permission* to this file to any person. Keep the access to this file at minimum whenever possible. You can control access to it using ACL rules, like shown in this example.

1. Permit user Joe and group Admins to the file named /etc/inetd.conf with read and write authority:

```
setfacl -m user:joe:rw-,group:admins:rw- /etc/inetd.conf
```

The -m option modifies ACL entries, or adds them if they do not exist

2. Set the base permission bits to prevent access by anyone other than the file owner:

```
setfacl -s user::rw-,group::---,other::---,user
user:joe:rw-,group:admin:rw:rw- /etc/inetd.conf
```

We recommend starting the InetD by using a shell script in /etc/rc, which causes InetD to be started automatically when z/OS UNIX is started.

The best method to verify that InetD is running correctly is to issue a **netstat** command to see the listener connections for InetD.

This section does not intend to explain how you can configure and start the InetD on z/OS Communication Server. You can check the IBM Redbooks publication titled *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 2 Standard Applications*, SG24-7997 for this.

3.6 Virtual IP addressing

In TCP/IP networking, Internet Protocol (IP) addresses are typically assigned to physical network interfaces. If a server has two physical interfaces, a separate IP address is assigned to each of them.

IBM introduced the concept of virtual IP addressing (VIPA) for its z/OS environment to support the use of IP addresses, representing TCP/IP stacks, applications, or clusters of applications, that are not tied to any specific physical interface. The association between VIPA and an actual physical interface is subsequently accomplished using either the Address Resolution Protocol (ARP) or dynamic routing protocols (such as OSPF).

3.6.1 Security concepts and architecture

VIPA provides a foundation technology solution for enhancing the availability and scalability of a z/OS system by providing the following capabilities:

- ▶ Automatic and transparent recovery from device and adapter failures

When a device (for example, a channel-attached router) or an adapter (for example, an OSA-Express adapter) fails, then another device or link can automatically provide alternate paths to the destination.

- ▶ Recovery from a z/OS TCP/IP stack failure

Assuming that an alternate stack is installed to serve as a backup, the use of VIPAs enables the backup stack to activate the VIPA address of the failed stack. Connections on the failed primary stack will be disrupted, but they can be reestablished on the backup using the same IP address as the destination. In addition, the temporarily reassigned VIPA address can be restored to the primary stack after the cause of failure has been removed. When Dynamic VIPA (DVIPA) is used, the VIPA takeover can be automatic, while the static VIPA must be taken over with an operation (using the OBEYFILE command).

- ▶ Limited scope of a stack or application failure

If DVIPA is distributed among several stacks, the failure of only one stack affects only the subset of clients connected to that stack. If the distributing stack experiences the failure, a backup assumes control of the distribution and maintains all existing connections.

- ▶ Enhanced workload management through distribution of connection requests

With a single DVIPA being serviced by multiple stacks, connection requests and associated workloads can be spread across multiple z/OS images according to Workload Manager (WLM) and Service Level Agreement policies (for example, quality of service, or QOS).

With the use of a DVIPA, nondisruptive movement of an application server to another stack is allowed so that workload can be drained from a system in preparation for a planned outage.

Preferred practice: When implementing VIPA follow these practices:

- ▶ Always define a VIPA and a VIPA backup to guarantee your environment availability.
- ▶ After the VIPA is implemented, exercise a takeover and takeback function in both automatic and manual operation to make sure that it will work in case it is needed.
- ▶ In a scenario where you are using an event-stacked DVIPA, an alternative for unique application-instance applications that do not contain the logic to bind to a unique IP address is to use the BIND parameter on the PORT reservation statement. It is usually a good practice to reserve a port for the listening socket of a server application.
- ▶ When using DVIPA, if possible, try to distribute it among several stacks. This can guarantee you that if one stack fails it will affect only the subset of clients connected to that stack. If the distributing stack experiences the failure, a backup assumes control of the distribution and maintains all existing connections.

The IBM z/OS Communications Server supports two types of virtual IP addressing (VIPA).

Static VIPA

A *static VIPA* is an IP address that is associated with a particular TCP/IP stack. Using either ARP takeover or dynamic routing protocol (such as OSPF), static VIPAs can enable mainframe applications communications to continue unaffected by network interface failures. As long as a single network interface is operational on a host, communication with applications on the host persist.

Static VIPAs have the following characteristics:

- ▶ They can be activated during TCP/IP initialization or VARY TCPIP, OBEYFILE command processing, and are configured using an appropriate set of DEVICE/LINK/HOME or INTERFACE statements and, optionally, OMPROUTE configuration statements or BSDROUTINGPARMS statements for IPv4 and IPv6 Static VIPAs.
- ▶ Using the SOURCEVIP configuration option, or SOURCEVIP parameter on the INTERFACE configuration statement, static VIPAs can be used as the source IP address for outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests.
- ▶ They can be specified as the source IP address for outbound TCP connection requests for *all* applications using this stack with TCPSTACKSOURCEVIP, or just a specific *job* and a specific *destination* through the use of the SRCIP profile statement block.
- ▶ The number of static VIPAs on a stack is limited only by the range of host IP addresses that are available for that host.
- ▶ They can be moved to a backup stack after the original owning stack has left the XCF group (such as resulting from some sort of failure), by using VARY TCPIP,,OBEYFILE command processing to configure the VIPA on the backup stack.

Note: Static VIPA does not require a sysplex (XCF communications) because it does not require coordination between TCP/IP stacks.

Dynamic

A *dynamic VIPA* (DVIPA) can be defined on multiple stacks and moved from one TCP/IP stack in the sysplex to another automatically. One stack is defined as the primary or owning stack, and the others are defined as backup stacks. Only the primary stack is revealed to the IP network.

TCP/IP stacks in a sysplex exchange information about DVIPAs, their existence, and their current location, and the stacks are continuously aware of whether the partner stacks are still functioning. If the owning stack leaves the XCF group (such as resulting from some sort of failure), then one of the backup stacks automatically takes its place and assumes ownership of the DVIPA. The network simply sees a change in the routing tables (or in the adapter that responds to ARP requests).

In this case, applications associated with these DVIPAs are active on the backup systems, thereby providing a hot standby and high availability for the services. DVIPA addresses identify applications independently of which images in the sysplex the server applications execute on and allow an application to retain its identity when moved between images in a sysplex.

Figure 3-9 shows a simple example of DVIPA that illustrates a failed TCP/IP stack and associated applications.

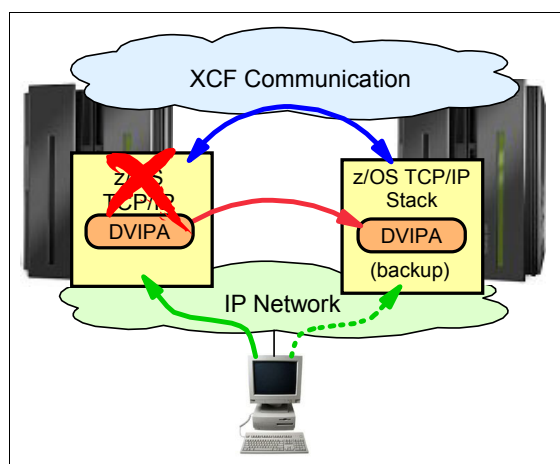


Figure 3-9 DVIPA address movement

Dynamic VIPAs are divided in three types:

- ▶ Stack-managed DVIPA
- ▶ Event-activated DVIPA
- ▶ Distributed DVIPA (Sysplex Distributor)

For a more detailed definition of Virtual IP addressing (VIPA), check the IBM Redbooks publication titled *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998.

3.7 z/OS IP filtering

IP filtering provides a means of *permitting* or *denying* IP messages (packets) into and out of the z/OS Communications Server environment at an early stage in message handling (and, therefore, efficiently). By *early* we mean at a point when the packet is just about to enter the protocol stack and before any applications are executed or any work is performed.

Depending on your security policies, IP filtering capabilities 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 to identify the IPSec behavior to apply to all traffic.

z/OS Communications Server IP filtering support is packaged with IP Security (IPSec) and is referred to as integrated IP Security because there is a 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.

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 two functions:

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

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), which is a table that maintains the set of filter rules, for a matching rule against the TCP/IP header. Filter rules have conditions and actions.

Filter rules contain the conditions to apply the rules based on any combination of the following attributes:

- ▶ Packet information
 - IP source or destination address (or masked address)
 - Protocol
 - Source or destination port address
- ▶ Direction of flow (inbound/outbound)
- ▶ Time of day

Filter rules also contain the actions to specify whether the packets that match the filter rule conditions are denied or permitted. The following possible actions can be taken:

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

What to do when traffic is blocked: When an IP packet is blocked, the source of the packet is usually not informed. However, a possibility is to configure a packet discard action to send an ICMP error when certain traffic is blocked.

To create the IP filtering policy, you must know the resources available in the network, the resources available in a z/OS image, and how they relate to others hosts. Figure 3-10 on page 109 illustrates the basic concept of IP filtering.

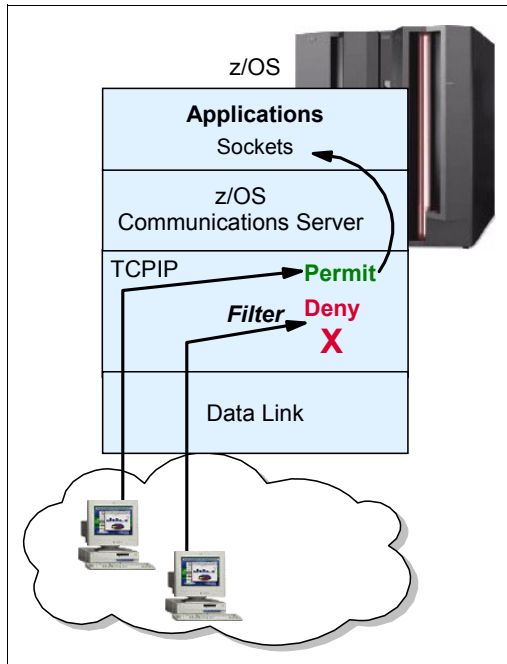


Figure 3-10 IP filtering at the z/OS data endpoint

Preferred practice: 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*. In other words, it is *not* the best use of a z/OS image to have it function as a firewall router (also called a secure gateway). You should rather *use* specialized network devices that are more suitable and cost-effective as routing firewalls.

The following resources are available in the z/OS image:

- ▶ TCP/IP address space and stack (can be more than one stack).
- ▶ The network interfaces and their respective IP addresses.
- ▶ The servers or clients, which are the address spaces running programs that will either access or be accessed by other hosts (such as TN3270 server, 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 following network resources should be mapped outside the z/OS image:

- ▶ 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.

Watch out: The implicit deny rules are always created using either the default or the filter policy. Using the IPSECURITY option in the IPCONFIG statement creates the implicit rules that deny all the inbound and outbound TCP/IP traffic in that z/OS image. Therefore, if you code IPSECURITY without either IPSEC statements or filter policies, the stack will be completely inaccessible.

Firewalls are so common that a definition is hardly needed; however, in a large organization the term should be formally defined. A firewall is an implementation (or extension) of an organization's security policies. A firewall controls and limits access between networks of different security classifications, and sometimes even within a network that is already protected by a firewall. Firewalls can filter based upon port numbers and IP addresses (or networks).

3.7.1 Security concepts and architecture

z/OS IP filtering defines the security parameters based on filter rules. This set of filter rules is called Security Policy Database. The security policy database provides two types of filters policies: the *default IP filter policy* and the *IP security filter policy*.

- The default IP filter policy

This policy provides only a basic filtering function (only permit rules and no VPN support). The default IP filter policy is defined in the TCP/IP profile; it contains an implicit rule to deny all traffic. It is intended to filter packets when the IP security filter policy is not available, in such situations that IP security filter policy is configured but the policy agent has not been started or completely initialized.

- IP security filter policy

The policy is intended to be the primary source of filter rules, and it also contains an implicit rule to deny all traffic. It is defined in a policy agent IPsec configuration file and can be generated by the z/OSMF Configuration Assistant.

You can use the **ipsec** command to switch between the default IP filter policy and IP security filter policy. Figure 3-11 shows a functional view of IP filter policy on z/OS.

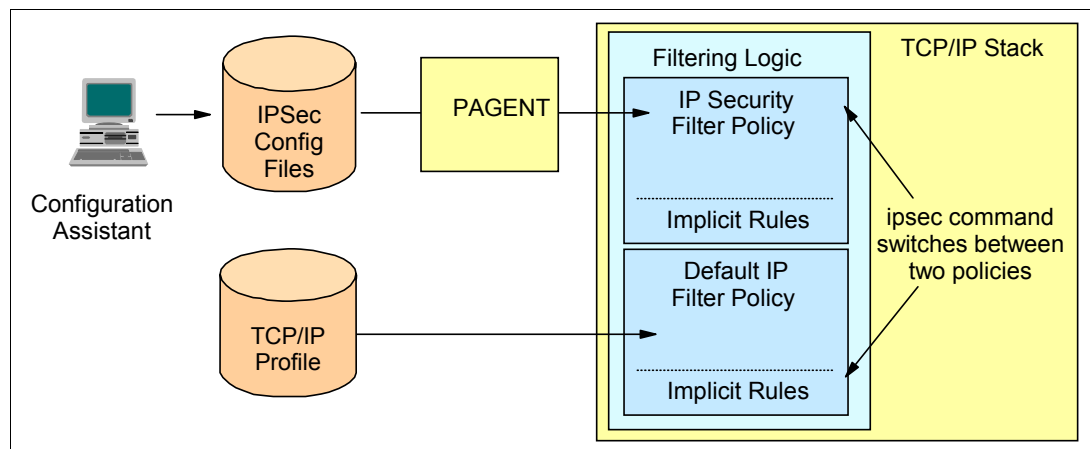


Figure 3-11 IP filter policy on z/OS

Preferred practice: Consider defining a default policy that has an IPSECRULE to allow one administrative IP address to connect to the TCP/IP stack. In this way, if PAGENT fails to start, you will still have a way to access the stack using TN3270.

3.8 IPSec

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 network layer of TCP/IP. IPSec provides the framework to define and implement network security based on policies defined by your organization.

This protocol is mostly used for implementing a virtual private network (VPN). This is due to the fact that IPSec works at the IP networking layer and 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.

Preferred practice: IPSec provides flexible building blocks that can support a variety of configurations. There are several protocols and encryption algorithms provided by IPSec to suit the security requirements of your installation. The IPSec security policies can be either defined manually or by using the z/OS MF Configuration Assistant, which creates the IP security configuration file for you. We recommend avoiding a manual configuration to provide a more reliable configuration.

The IPSec standards have also been structured so that they can accommodate newer, more powerful cryptographic algorithms as they become available in the future.

IPSec technology is used to build what is referred to as a virtual private network, because the technology 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. Organizations can 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 unavailable.

Background information: IBM z/OS Communications Server IP filtering support is packaged with IP Security (IPSec) and is referred to as integrated IP Security because there is a 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. To configure IP filtering, you must indicate that you are configuring IPSec in the z/OSMF Configuration Assistant. For more information, see 3.7, “z/OS IP filtering” on page 107.

3.8.1 Security concepts and architecture

The IPSec architecture provides a framework for security at the IP layer of IPv4 and IPv6. IPSec defines a unidirectional logical connection between two endpoints. Such secure logical connections between pairs of endpoints are often called *tunnels*. z/OS Communications Server IPSec implementation refers to two types of tunnels:

- ▶ Manual IPSec tunnels

The security parameters and encryption keys are configured statically and are managed by a security administrator manually. Manual tunnels are not commonly implemented. Although simple to establish, manual tunnels are not considered entirely secure because manually configured keys can be compromised easily. Also, with manual tunnels, IPSec keys cannot be changed without deactivating and reactivating the secure tunnels, thus opening up a window of vulnerability during the data transfer.

- ▶ Dynamic IPSec tunnels

The security parameters are negotiated, and the encryption keys are generated dynamically. Inadvertent or intentional breach of the security keys with dynamic tunnels is nearly impossible. Thus, dynamic IPSec tunnels are preferred for robust security implementations. The Security Associations (SA) and the encryption keys are negotiated using the IKE protocol, which is discussed in more detail in the RFC5996 bullet on page 114.

Preferred practice: Consider using dynamic tunnels rather than manual tunnels. Manual tunnels are not considered a secure form of IPSec.

The concept of a Security Association is fundamental to IPSec, defining the security characteristics of the traffic that is carried across the tunnel. The span of protection of an SA can vary. For example, the SA can protect traffic for multiple connections (all traffic between networks), or the SA can protect traffic for a single connection.

IPSec can provide a secured connection and an encrypted payload with its implementation. The authentication proves data origin authentication, data integrity, and replay protection, which are explained as follows:

- ▶ *Data origin authentication* confirms that the data origin was from a device that knows the correct cryptographic key.
- ▶ *Data integrity* proves that the contents of a datagram have not been changed because the authentication data was created.
- ▶ *Replay protection* prevents an attacker from sending bogus IPSec packets resulting in unnecessary cryptographic operations. For example, if an attacker kept retransmitting the Encapsulating Security Payload (ESP) last packet sent, replay protection will prevent that packet from being decrypted and authenticated each time. The sequence number in the IP header is always in clear text.

The Authentication Header (AH) protocol is the IPSec-related protocol that provides authentication. The Encapsulating Security Payload (ESP) protocol provides data encryption, which conceals the content of the payload. ESP also offers authentication. The Internet Key Exchange (IKE) protocol exchanges the secret number that is used for encryption or decryption in the encryption protocol.

As shown in Figure 3-12 on page 113, AH and ESP support two modes types:

► Transport mode

This mode tells IP how to construct the IPSec packet. Transport mode is used in host-to-host scenarios in which the two endpoints of the tunnel (the security endpoints) are the same as the host endpoints (the data endpoints). With transport mode, the IP header of the original transmitted packet remains unchanged.

► Tunnel mode

Tunnel mode is most frequently used when either the endpoint of the tunnel is a router or a firewall. With tunnel mode the security endpoints are separate from the data endpoints. With tunnel mode, a new IP header is constructed and placed in front of the original packet.

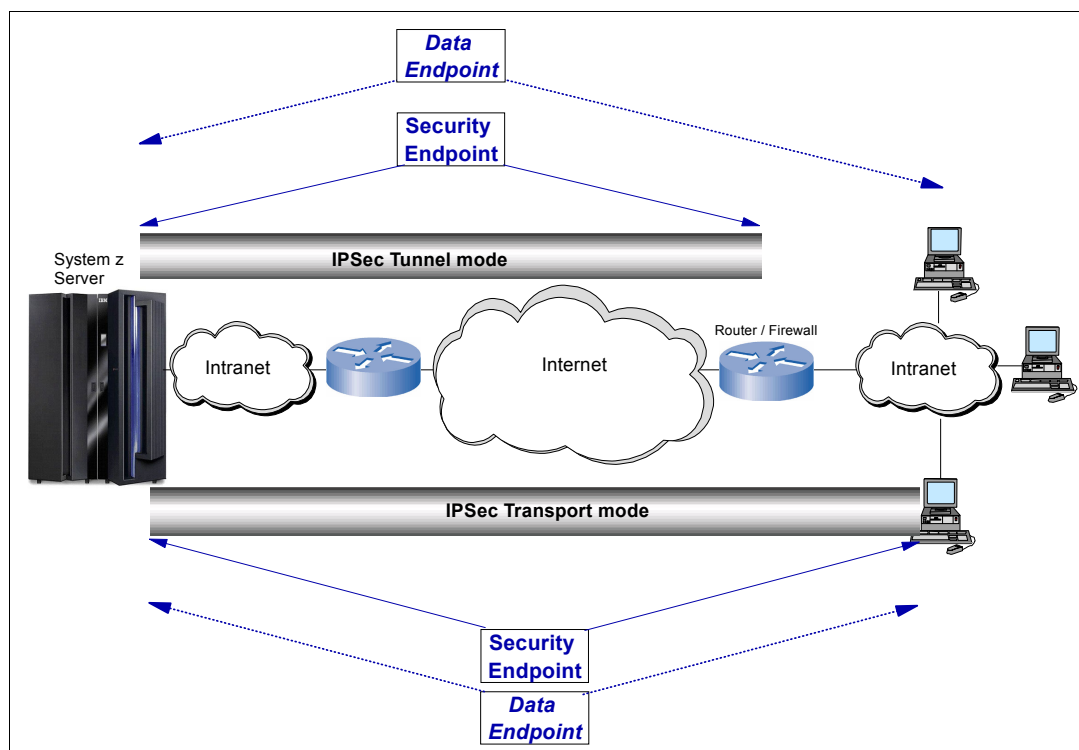


Figure 3-12 Transport and tunnel mode

Key components

Below we provide a brief description of the three most important IPSec components implemented by z/OS Communications Server:

► RFC 4302

The first component is the IP Authentication Header (AH) protocol. It provides data authentication, IP header authentication, and data origin authentication. IPSec AH authenticates IP packets, ensuring that they came from a legitimate origin host and that they have not been changed. IPSec AH also provides the following features:

- Data integrity by authenticating the *entire* IP packet using a message digest that is generated by algorithms such as HMAC-MD5 and 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.

► RFC 4303

The next component is the IP Encapsulating Security Payload (ESP). It provides for data authentication, data origin authentication, and data privacy (encryption). The ESP provides additional protection beyond (or in addition to) AH, as follows:

- Encapsulating and encrypting the IP packet.
- Authenticating the IP datagram portion of the IP packet, including most of what is listed under AH — data integrity for all but the IP header, data origin authentication, and replay production. For most users, the authentication protection provided by ESP is already being used for encryption

In the 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.

► RFC 5996

The third component is the Internet Key Exchange (IKE), which provides protocols for automated encryption key management. The IKE protocol, which is implemented in z/OS Communications Server by the IKE daemon, manages the transfer and periodic changing of security keys between senders and receivers and is required when implementing IPSec dynamic tunnels.

Preferred practice: When the IKE daemon has obtained the IPSec policy, the policy agent can be stopped without impacting the IKE daemon. However, any changes to the IPSec policy are not detected until the policy agent is restarted. The IKE daemon reconnects to the policy agent when it is restarted. Avoid doing that to keep the policies always set and running.

Key exchange, defined in IKE, is normally a multi-step process, as described here and as shown in Figure 3-13 on page 115:

- a. First, the partners establish a secure logical connection, an SA, and decide on security parameters such as encryption, hashing algorithms, and authentication methods (IKE SA). This connection is sometimes referred to as the *phase 1 tunnel* or *IKE tunnel*.
- b. After the appropriate security parameters are negotiated, the partners set up a second SA for the actual data transfer (IPSec or child SA). This connection is sometimes referred as *phase 2* or *IPSec tunnel*.
- c. Thereafter, the SAs and the session keys are renegotiated periodically. The IKE daemon uses the IP security policies that you define in the policy agent (PAGENT) and manages the keys dynamically that are associated with dynamic IPSec VPNs.

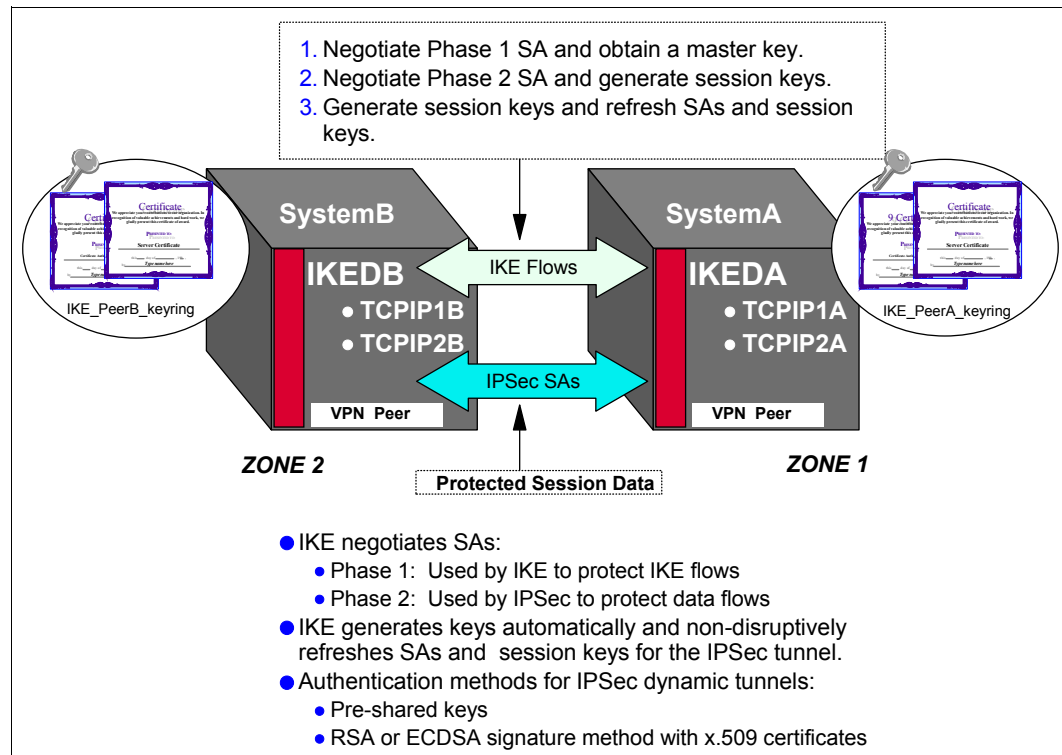


Figure 3-13 IKE concepts

There are two methods of authenticating the IPsec when using dynamic tunnels:

- Authenticate the IPsec partner with a pre-shared key
- Authenticate the partner using digital signature mode

The term *pre-shared key* can be considered a misnomer, because the so-called “keys” are more akin to secret “passwords” that are shared between the IPsec peers to authenticate the partner during the Phase 1 IKE exchange and to provide a value to the Diffie-Hellman exchange that produces a cryptographic key to protect and authenticate the Phase 1 IKE negotiations.

Digital signature mode authentication relies on x.509 certificate exchanges to provide verification of the trusted partner. z/OS supports two digital algorithms: RSA and ECDSA. The certificates are stored in the IKE key rings of the peers, as shown in Figure 3-13. The local identity of an IPsec peer must be configured in the IPsec policy, and it must represent an identity established in x.509 certificate on the local IKE key ring. The remote identity of an IPsec peer must also be configured in the IPsec policy, and it must represent an identity established in the x.509 certificate presented by the remote IKE peer during Phase 1 negotiations.

When implementing digital signature mode on z/OS, certificate management services can be provided at the local z/OS IKE daemon (IKED) or by a Network Security Services daemon (NSSD) running locally or in another, more secure network zone on z/OS. IKED supports local digital signature mode for IKE version 1 (IKEv1) negotiations only, but NSSD supports both IKEv1 and IKE version 2 (IKEv2).

Preferred practices:

The z/OS Communications Server supports IKEv2 in addition to IKE version 1(IKEv1). IKEv2 has better performance and operational characteristics than IKEv1. Many government agencies expect the vendors who do business with them to use IKEv2 to establish secure communications with them. Consider using IKEv2 to increase security over IPSec use.

If you are setting up a production environment, consider setting the `IkeSyslogLevel` and `PagentSyslogLevel` to low to avoid a performance impact from excessive logging.

You can find more detailed explanations of how to configure and implement IPSec in the IBM Redbooks publication titled *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999.

3.9 z/OS Intrusion Detection Services

In this section we cover the z/OS Intrusion Detection Services (IDS). *Intrusion* is a term used to describe undesirable activities in an IT environment. The objective of an intrusion might be to acquire information that a person is not authorized to have. It might be to gain unauthorized use of a system as a stepping stone for further intrusions elsewhere. It might also be used to cause an organization harm by rendering a network, host system, or application unusable. Most intrusions follow a pattern of information gathering, attempted access, and then destructive attacks. Intrusion detection services (IDS) guard against these intrusions, providing protection against potential attackers.

Intrusion Detection Services (IDS) is a z/OS Communications Server security protection mechanism that inspects inbound and outbound network activity, and identifies suspicious patterns that might indicate a network or system attack from someone attempting to break into or compromise a system. IDS is integrated into the TCP/IP stack's processing and can detect malicious packets that are designed to be overlooked by a firewall simplistic filtering rules. IDS provides a reactive system whereby it responds to the suspicious activity by taking policy-based actions that include the ability to drop attack packets. It can also limit TCP and UDP traffic by dropping packets that exceed the configured connection limit or receive queue length.

3.9.1 Security concepts and architecture

Figure 3-14 shows an overview of the IDS architecture with all the components involved.

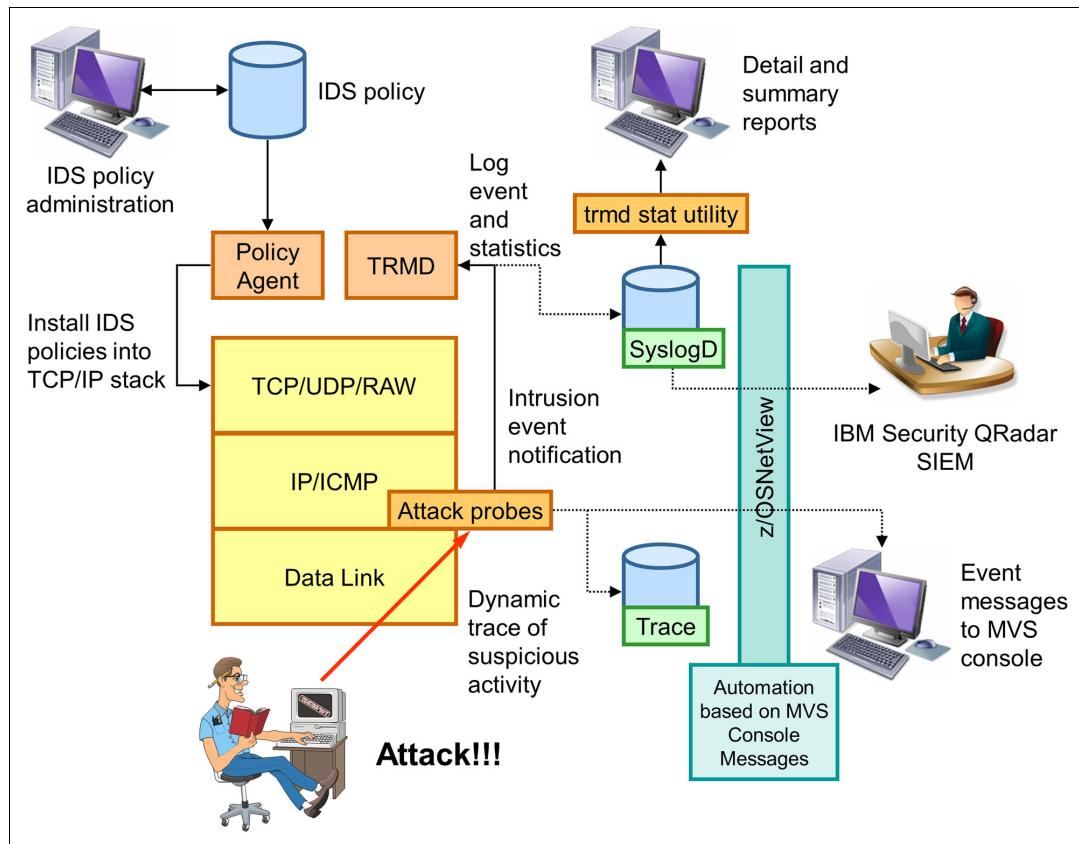


Figure 3-14 Intrusion Detection Services architecture

IDS functions can be divided into the following three areas:

- ▶ Scan detection
- ▶ Attack detection
- ▶ Traffic regulation

A well configured IDS can add another layer to the security structure of your environment. On the other hand, it might cause you a lot of problems and false-positive findings if not configured properly and not well adapted to your business goals.

IDS is managed through policies. The policies are determined by the network administrator and are based on preconceived events. The policies must include factors such as who, what, where, when and how:

- ▶ *Who* is allowed to connect to the host?
- ▶ *What* applications or ports are clients allowed to use?
- ▶ *Where* is the attack, intruder, or 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?

IDS policies can be defined and stored in a *policy agent configuration file* and an *LDAP server*. This solution provides an IDS policy solution that is consistent with other policy types for installations that do not have an LDAP infrastructure in place or that favor using configuration files rather than an LDAP configuration.

The policy agent configuration file can be created with the stand-alone version of the z/OSMF Configuration Assistant or the one running under z/OSMF. In this section, we cover only the policy agent configuration file that is created with the z/OSMF Configuration Assistant running under z/OSMF.

The policy information is loaded into the policy agent application during PAGENT startup. All IDS policies allow the logging of events to a specified message level in syslogd or the system console. Most IDS policies support discarding packets when a specified limit is reached, and they support writing statistics records to the INFO message level of syslogd on a specified time interval, or if exception events have occurred.

Preferred practice: Check your LDAP server log or command output for errors encountered when your policies are loaded into LDAP. Some LDAP servers treat consecutive blank lines in an LDIF file as end-of-file; ensure that all of the policy objects in your LDIF files are acknowledged by LDAP.

In addition, check your policy agent log file for errors while processing your policy.

And finally, use the **pasearch** command to verify that the intended policies are active and have the expected attributes for the target stack.

Most IDS policies also 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 can involve multiple packets. The correlator value helps to identify which messages and packets are related to each other.

Preferred practice for creating policies: The GUI can potentially reduce the amount of time that is required to create IDS policy files, contributing to ease configuration and maintenance. Because of the inherently complex nature of z/OS security, using the GUI can help you ensure that you have a consistent and easily manageable interface for implementing your IDS.

The IDS implementation provides you with policy samples, which you can use rather than configuring your own from scratch. We recommend starting by using the samples and to adapt them whenever possible. This can help you build less error-prone policies because you will have to deal with a very large variety of attributes.

Preferred practice for acting on intrusions: The IDS and its scan policies do not provide the ability to reject a connection. The actual rejection of the connection based on the source IP address must be configured in the traffic regulation policy or firewall.

If possible, try to use Tivoli NetView to manage your IDS. Tivoli NetView for z/OS provides local management functions for IDS and the ability to trap IDS messages from the system console or syslogd and take predefined actions based on IDS type events such as:

- ▶ Route IDS messages to designated NetView consoles
- ▶ Email notifications to security administrators
- ▶ Run trmdstat and attach output to email
- ▶ Issue pre-defined commands

See “Configuring NetView for z/OS” for more information:

<http://ibm.co/1VbmDRd>

Next, we examine the three types of policies:

- ▶ Scan policy
- ▶ Attack policy
- ▶ Traffic regulation policy

Scan policy

Scans are used to gather information about a system. Scans are usually not harmful and can be part of normal operations, but many serious attacks are preceded by information-gathering scans. For example, a port scan can determine which ports are open on a system and, therefore, are potentially vulnerable to attacks.

z/OS Communications Server detects a scan when multiple unique information gathering events from a single source IP address occur within a defined period of time. Because most 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 in the policy. Two categories of scans are supported:

- ▶ Fast scans

During a *fast scan*, many resources are rapidly accessed over a short time period. They are usually program-driven and take less than 5 minutes.

- ▶ Slow scans

During a *slow scan*, resources are accessed intermittently over a longer period of time (sometimes over many hours). A slow scan can occur when a scanner is trying to avoid detection.

A fast scan scenario might 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 can be the basis for a future attack.

A slow attack is more deliberate. Occasional packets can 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 with the same source IP address generates 20 connections to port 23 (TN3270 telnet server) it will not be considered a scan due to the fact that one unique resource has been accessed.

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 using an exclusion list
- ▶ Specify a sensitivity level by port or port range (to reduce performance impacts)
- ▶ Notify the installation of a detected scan through a console message or syslogd message
- ▶ Trace potential scan packets

The policy allows the administrator to set a sensitivity level, which is known as *policy-specified sensitivity*. This sensitivity level is used in parallel with the categorization of

the individual packets to determine whether a packet should be counted as a scan event. The event classification defines a 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 the threshold has been met or exceeded in a specified time interval.

Scan events

Scan events are classified into Internet Control Message Protocol (ICMP), UDP port, and TCP port scan categories. The scan events 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.
- ▶ **ICMPv6 scan**
ICMPv6 echo requests are used to obtain or map network information.
- ▶ **TCP port scans**
TCP is a stateful protocol. Most events can 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 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.

False positive scans

The 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 can be 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 values (client IP address, client port, server IP address, and server port) and the IP protocol are unique 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.

Attack policy

An *attack* is defined as an assault on system security. An attack is usually an intelligent act that includes a deliberate attempt to evade security services and violate the security policy of a system. In practice, however, an attack can be inadvertent. For example, a malfunctioning host or application could generate a flood of SYN packets to a host. The security policies will not differentiate between a malicious intent and an accident. And, they should not differentiate because attack policies exist to protect the z/OS host.

In fact, one of the difficulties of attack policies is how to differentiate between a valid user application repeatedly attempting to make a connection and a malicious program trying to inflict damage and penetrate the IT defenses. An attack can be in the form of a single packet or multiple packets. There are two types of attacks, active and passive¹:

- ▶ An *active attack* is designed to alter or steal system resources or affect their operation.
- ▶ A *passive attack* is designed to learn or make use of system information, but not affect the system performance and integrity.

¹ These definitions are excerpts from the *Internet Security Glossary, Version 2*, RFC 4949:
<https://tools.ietf.org/html/rfc4949>

The attack policies designed for IDS are based on active attacks. Scanning can be considered a passive attack.

The 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.

The IDS attack types that can be enabled on the z/OS Communications Server are described in Table 3-3 on page 121.

For each enabled attack type, you configure the types of notification that are provided if an attack is detected. Also, you configure whether the packet should be discarded or not.

Table 3-3 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.	<ul style="list-style-type: none"> ▶ TCP/IP stack: Always discards malformed packets. ▶ IDS policy: Can provide notification.
Inbound IPv4 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 88 bytes of a packet.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause the packet to be discarded.
IP protocol restrictions	There are 256 valid IP protocols. Only a few are in common use today. This support allows you to protect your system against future attacks by you defining those protocols as restricted. IP protocol restrictions are configured independently for IPv4 and IPv6 because the headers differ. For IPv4, there is an IP protocol restrictions attack type. For IPv6, there is a next header restrictions attack type.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause the packet to be discarded.

Category	Attack description	Actions
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 IPv4 and IPv6 options is performed on all inbound packets, even those forwarded to another system. IP option restrictions are configured independently for IPv4 and IPv6 because IP options are implemented differently. For IPv4, there is a single IP option restrictions attack type. For IPv6, there is a hop-by-hop option restrictions attack type and a destination option restrictions attack type.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause the packet to be discarded.
UDP perpetual echo	Certain UDP applications unconditionally respond to every datagram received. In certain cases, such as Echo, CharGen, or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases, it might 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.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause packet to be discarded.
ICMP redirect restrictions	ICMP redirect packets can be used to modify your routing tables.	<ul style="list-style-type: none"> ▶ TCP/IP stack: Will discard ICMP redirects if IGNOREREDIRECT is coded in the tcpip.profile. ▶ IDS policy: Can provide notification and disable redirects (this can optionally be coded as a parameter in the tcpip.profile).

Category	Attack description	Actions
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. Generally, you should restrict the TCP protocol on the outbound raw rule. Outbound raw restrictions are configured independently for IPv4 and IPv6.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause the packet to be discarded.
TCP SYN flood	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.	<ul style="list-style-type: none"> ▶ TCP/IP stack: Provides internal protection against SYN attack. ▶ IDS policy: Can provide notification.
Interface flood	An interface flood is detected when a large number of the incoming packets over an interface is being discarded. Enabling the flood attack allows you to receive notification about the attack.	<ul style="list-style-type: none"> ▶ TCP/IP stack: Discards the packets. ▶ IDS policy: Can provide notification.
Data hiding	Detects inbound IP packets that might contain hidden data. Checking is done for both IPv4 and IPv6 packets.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause the packet to be discarded.
TCP queue size	Detects attacks targeting TCP/IP send, receive, or out-of-order queues. Constraint triggered when data on queue for at least 60 seconds or a "fixed" limit of data on queue for at least 30 seconds.	<ul style="list-style-type: none"> ▶ TCP/IP stack: Storage marked page eligible. ▶ IDS policy: Can provide notification and cause the connection to be reset.
Global TCP stall	A global TCP stall condition is detected when at least 50% of the active TCP connections are stalled and at least 1000 TCP connections are active.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action. ▶ IDS policy: Can provide notification and cause all stalled connections to be reset.
EE malformed packet	Malformed EE packets are attempts to crash the EE protocol stack.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action ▶ IDS policy: Can provide notification and cause the packet to be discarded

Category	Attack description	Actions
EE LDLC check	EE LDLC control packets must flow over the EE signaling port. Control packets received on another port might represent an attacker's attempt to probe or crash a system.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action ▶ IDS policy: Can provide notification and cause the packet to be discarded
EE port check	EE packets are expected to use the same source and destination port numbers. Packets received with differing port numbers might represent an attacker's attempt to probe a system.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action ▶ IDS policy: Can provide notification and cause the packet to be discarded
EE XID flood	An attacker might attempt to consume system resources, or block or delay legitimate EE connections, by generating a flood of EE XIDs. This attack type monitors for EE XID timeouts that can result from an attacker's spurious XIDs.	<ul style="list-style-type: none"> ▶ TCP/IP stack: No default action ▶ IDS policy: Can provide notification

Traffic regulation policy

The two types of traffic regulations policies are as follows:

- ▶ TCP traffic regulation policies

The IDS traffic regulation 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.

- ▶ UDP traffic regulation policies

Traffic regulation (TR) policies for UDP are used to limit memory use and queue delay time.

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.

Preferred practice for reporting, analytics, statistics, and enterprise integration:

Intrusion Detection Services for z/OS have a lot of reporting options available that you can use to report about every captured event. It gives you the ability to log the events on the system console, to syslogd, or even both. Try to determine what is the best way for you to keep track of these events.

IDS for z/OS also provides a packet trace feature. Make sure to have all packets traced, so that you can analyze each and every packet destined to your network with malicious intentions.

You can keep statistics for various IDS events that occur. Those statistics can come in handy whenever you need to review your policies or run any forensic activity.

Last but not least you need to understand that the IDS capability on your z/OS mainframe has to be regarded as an additional intrusion defying barrier for malicious attacks. You must also deploy sophisticated network intrusion detection and intrusion prevention services (IPS) for your general network infrastructure, beginning in your DMZ and following all the way through to your subnetworks.

Those IDS and IPS services typically collect overwhelming amounts of data that can best be used in a centralized fashion for enterprise-wide security intelligence. A typical solution for this centralized capability is called a Security Information and Event Management system (SIEM), like IBM Security QRadar® SIEM (shown previously in Figure 3-14 on page 117).

3.10 IP resource security

The next stage to securing your enterprise IP network is to protect the resources on your IBM system. This can be done through several controls, which range from RACF to IP configuration options. In this section, we explain different controls to give you a better understanding on how to use them within your ecosystem.

3.10.1 SAF controls

The base of all security on z/OS comes from the Security Access Facility (SAF). The IP stack and the facilities it provides are considered to be a protected resource in the same sense as data sets or other authorized functions of the system. The z/OS IP stack uses SAF to protect itself, through resource profiles defined in the SERVAUTH resource class. This is controlled by a security manager such as IBM RACF. The SERVAUTH resource class can control a wide variety of IP functions, from displays of network information to the ability to connect to other resources within your enterprise.

This is the general profile format of the SERVAUTH class for TCP/IP:

```
EZB.resource_category.system_name.job_name.resource_name
```

Where:

- ▶ EZB designates that this is a TCP/IP profile.
- ▶ resource_category is the capability area to be controlled.
- ▶ system_name is the name of the operating system, and can be used as a wild card.

- ▶ `job_name` is the job name associated with the resource access request (stack name) and can be wild carded.
- ▶ `resource_name` is optional and can further refine the resource to be protected.

For example, access to stack TCPIP1 on MVS1 can be protected by defining the resource EZB.STACKACCESS.MVS1.TCPIP1 and granting individual users access to the resource in the SAF-based product in use.

These are the main categories of resource protection (`resource_category`):

STACKACCESS	The STACKACCESS control category regulates access to the stack.
PORTACCESS	The PORTACCESS control category is used with a resource name specified on the PORT or PORTRANGE statement after the SAF keyword.
NETACCESS	The network access control category is used with the NETACCESS block of statements to divide the external network into security zones and limit access by zone (<code>resource_name</code>).
NETSTAT	The NETSTAT access control category is where access to particular netstat options may be limited by defining profiles with the options as the <code>resource_name</code> .
PAGENT	The PAGENT access control category limits access to the <code>pasearch</code> command. Access may be limited by policy type (IDS, QoS) by specifying the profile with the policy type as the <code>resource_name</code> .
SNMPAGENT	The SNMPAGENT access control category allows the ability to control use of SNMP subagents that connect to the TCP/IP SNMP agent.
MODDVIPA	The utility program control limits the ability to create new dynamic VIPAs within a VIPARANGE by using the MODDVIPA utility.
FTP	The FTP SITE command access control category limits the ability to use STE DUMP and DEBUG commands, because these can write significant amounts of output. The <code>resource_name</code> is specified as <code>SITE.DUMP</code> and <code>SITE.DEBUG</code> , respectively.
FRCAACCESS	The Fast Response Cache Accelerator (FRCA) access control category limits the ability to create an FRCA cache.

SAF security packages and resource definitions

TCP/IP will always make the resource authorization check when one of those categories of access is encountered. The SAF interface provides for notification of three different conditions as a result of the check:

- ▶ 0, the access is permitted
- ▶ 4, the resource is not defined
- ▶ 8, the access is not permitted

z/OS TCP/IP treats the 4, *resource not defined*, condition as meaning that the access is permitted. In other words, defining the resource means that access control is wanted, but in the absence of such a definition, resource access control is not wanted. Unfortunately, not all security products return all three conditions. ACF2, for example, treats a resource being undefined as an access check failure, and returns condition 8, *access not permitted*.

The way to deal with this is to define enough resource profiles to satisfy the security needs, and then to permit unrestricted access to those categories where access control is not an issue. If the entire stack is to have unrestricted access initially, the profile EZB.*.*.* could be defined, with unrestricted access, and the security checks would always result in access being permitted.

3.10.2 Multi-level security

Multilevel security is the strictest method of implementing platform hardening for an operating system. It involves labeling all of the resources and users within a system and basing their security access on the labels that were assigned.

Unlike single-level security systems (such as workstations), where there is no ability to properly label information and user sessions, multi-level security (MLS) systems allow you to tag data and user sessions to ensure that the information is controlled and information *write-down* is prevented. Information write-down is the accidental or intentional declassification of information by allowing it to be written to a resource with an incorrect security label.

There are a group of attributes that must be configured for MLS to determine the type of data and if an application or user should have access and the ability to write that information to another place.

- ▶ Security level
This is the familiar term dealing with the sensitivity of information and a person's clearance to process it.
- ▶ Category or compartment
There might be a category for accounting, another for logistics and another for cryptographic methodology. Categories are used to enforce *need to know* policies.
- ▶ Security label
A security label (seclabel) is an eight character name. It represents a particular security level and a set of categories that are defined in the security server.
- ▶ Equality, equivalence and dominance
These are special checks that are done using a seclabel to ensure that information is handled appropriately.
- ▶ Profile name
Authorized programs that manage access to data, network and other resources are assigned resource classes by the security server. Those programs then document how to form a profile name that represents the resources it wants to protect.

Some of the applications that have this capability are:

- ▶ OMPROUTE
- ▶ Ping
- ▶ UNIX FTP Server
- ▶ UNIX Telnet Server
- ▶ TN3270

Although this security technique provides a lot of security, it also incurs a large cost in administrative overhead. It is not recommended that you use this technique unless your requirements dictate that you implement this level security.

3.10.3 OSA-Express connection isolation

In a typical data center environment, OSA-Express interfaces are shared among multiple TCP/IP stacks due to their large available bandwidth (1 GbE or 10 GbE) and as a means to minimize infrastructure costs. In some instances, the sharing of resources also presents a security risk because the OSA-Express adapter can internally route IP packets directly to a TCP/IP stack that shares the same port.

OSA-Express connection isolation is a way to prevent IP communication between two TCP/IP stacks that share the same OSA-Express port. It also provides extra assurance against a misconfiguration that might otherwise allow such traffic to flow. When OSA-Express connection isolation is in effect, the OSA-Express feature discards any unicast packets when the next-hop address is registered by a TCP/IP stack sharing the same port and prevents any multicast or broadcast packets from being internally routed between the TCP/IP stacks sharing the port.

OSA-Express connection isolation can also be useful if you want to ensure that traffic flowing through the OSA adapter does not bypass any security features implemented on the external LAN.

Note: Dynamic routing protocols such as OSPF are not aware of OSA-Express connection isolation, which can be an issue if static routes are not used and traffic needs to flow between the TCP/IP stacks that share the OSA adapter using connection isolation.

3.10.4 IP Profile Controls

Within the TCP/IP profile, there are mechanisms that the system programmer can enable to reduce the security risk exposure. In this section, we describe some of the more useful parameters available to the system programmer to secure the mainframe environment.

FinWait2 timer

TCP is a connection-oriented protocol. Thus, before a new connection is established there is a three-phase handshake that must occur. The handshake consists of the following phases:

- ▶ Hello Phase
- ▶ Ready to Start
- ▶ Begin Connection

The Hello Phase is sent from the client to the server. The Ready to Start phase is sent back to the client from the server. Finally, the client sends the Begin Connection message back to the server.

Malicious attackers have used this innocent process to attack systems using what is called a SYN flood attack. A SYN-flood is when you have one or more clients initiating a TCP connection to a server but never sending the final phase. A stack can quickly be overwhelmed by keeping requests open while attempting to serve new connections. On the TCPCONFIG statement there is an option called *FINWait2time* that can be used to tune the time that a system will allow a TCP handshake to be between the second and third phase before closing the connection.

Preferred practice: The recommendation is to not keep the default, which is five minutes, but change the value to be 75 seconds. This provides ample time to complete the TCP handshake protocol, and it frees resources quickly enough to avoid a successful SYN-flood, in most cases.

RestrictLowPorts

The two most common transport layer protocols, TCP and UDP, both use port numbers to separate which application is talking to which client. There are a defined group of *well known ports* that are a set of services defined by the IETF — ports 1-1023. These ports are used for applications such as TN3270, FTP, ISAKMP, and many other basic services. It is a common practice for an attacker to start a new service *masquerading* as a well-known application bound to one of these well known ports. Instead of connecting to a legitimate application, a user would get the malicious application giving up information like user IDs and passwords unknowingly. We can configure the IBM Communication Server for z/OS to protect these well known ports from any random user accessing them.

Preferred practice: Enable the RestrictLowPorts keyword in the TCPCONFIG and UDPCONFIG statements to prevent any service from binding to one of the well-known ports unless they are defined in the PORT list in the IP profile.

Port definition

Port numbers are the doorway into your system for the TCP and UDP protocol. It is important to control port access, so that only specified applications can access specific ports. You can reserve ports using the PORT or PORTRANGE profile statements. These statements allow you to reserve ports for well known or configured ports for the applications that need to bind to them. For example, let's say you have a TN3270 server, an FTP daemon, and a local server. You could code the following ports:

23 TCP TN3270S	Only an application called TN3270S will be able to bind to this port.
21 TCP FTPD*	Any FTP daemon that starts with FTPD can use this port.
5000 TCP *	Any server can bind to this port.
5001 TCP RESERVED	The port is not available for use by any user.

If you wanted to further secure these ports, you can add an optional SAF parameter to provide additional access control. If you specify the SAF keyword in the PORT or PORTRANGE statement, it can provide additional access control by verifying that the user ID associated with an application at the time of a bind to the port is authorized to access the port.

You need to define an SAF SERVAUTH profile, such as this example:

```
EZB.PORTACCESS.sysname.stackname.port_safname
```

Where port_safname is the same value that you specify on the SAF keyword of the PORT or PORTRANGE statement. The user ID that is associated with the application at the time of the bind request must have READ access to this resource for the application to be able to bind to the port.

Now, to modify the previous example: If we have a group called STCGRP that all of the started task user IDs were part of, we could define the following:

- Define RACF PORTACCESS profiles:

```
RDEFINE SERVAUTH EZB.PORTACCESS.LPAR.TCPIP.STARTED UACC(NONE)
PE EZB.PORTACCESS.LPAR.TCPIP.STARTED CLASS(SERVAUTH) USER(STCGRP) ACCESS(READ)
```

- Modify the port statement:

```
23 TCP TN3270S      STARTED
21 TCP FTPD*        STARTED
5000 TCP *
5001 TCP RESERVED
```

Only the started tasks that match both the name and the started task user ID will be allowed to bind to ports 21 and 23.

UNRSV

Because there are over 65,000 ports to a stack, it would be time-consuming to reserve all of the ports. So you can use the UNRSV keyword to lock down all of the other ports on your stack. Note, you only want to do this for server ports because clients will need to grab a random port to connect to another server. It is recommended that at the end of your port statements you add the following instruction:

```
UNRSV TCP DENY WHENLISTEN
```



SNA security

In this chapter, we provide an overview of security concepts and architecture for SNA on System z. Then, we describe some of the guiding principles for configuring SNA security.

The chapter includes the following sections:

- ▶ 4.1, “Introduction” on page 132
- ▶ 4.2, “SNA encryption versus IP encryption” on page 132
- ▶ 4.3, “Security controls using VTAM start options” on page 133
- ▶ 4.4, “Transport security” on page 135
- ▶ 4.5, “TN3270 Security” on page 138
- ▶ 4.6, “Searching security” on page 141
- ▶ 4.7, “Application security” on page 149
- ▶ 4.8, “Recap of recommendations” on page 154

General overview: See Chapter 1, “Mainframe network concepts and functions” on page 1 for a more general overview of SNA-related topics. Be sure that you are familiar with these concepts.

4.1 Introduction

In today's ever expanding business environment, securing an IT environment takes center stage in an overall business strategy. However, as the IT industry has enhanced security defenses and best practices for IP networks over the last decade, configuring secure SNA environments have not been pursued with the same zeal by some organizations.

In the past, SNA system programmers were able to rely on a hierarchical architecture, strong physical controls, and a limited amount of access to protect their critical business services. As the IT industry developed much larger networks, these assumptions no longer hold true. The introduction of new technologies that enhanced the availability of SNA with more dynamic network recovery and the use of the faster IP infrastructure, has *opened* the SNA networking environment. This loss of physical security, however, can be replaced by a mix of other strong security options that are integrated into the different layers of SNA.

SNA at its core was designed with the ability to wrap different layers of connections with a blanket of security. To communicate within an SNA environment you would first have to connect to a node and establish and maintain a link connection into the network. You then have to correctly negotiate a proper session and then handle the flows within the session itself. At each level, there are different security controls that can govern the connections and protect the session information.

From network border search controls, tuning options and the use of encryption ciphers, you can harden the walls of your SNA network from the inside. Even the data transactions are wrapped within several layers of connections that can contain multiple security checks.

SNA is also enhanced to take advantage of IP networking technology. To protect the transactions that flow over an IP network, you can deploy a mix of SNA and IP security. Using these controls, you can authenticate the traffic flow and protect the data from prying eyes using advanced key management and cipher algorithms.

4.2 SNA encryption versus IP encryption

Within an SNA environment, there are currently two types of transmission links that can be used. Those that run over native SNA links (XCF, AHHC, Ethernet, Token-ring) and those that tunnel over IP links (TN3270, DLSw, Enterprise Extender). IP encryption and key management techniques have advanced at a much greater pace than those within the native SNA environment. SNA encryption can still be used to secure the whole path of an LU-LU session regardless of the number of hops in between the LU partners, whereas IP security can only protect portions of the path that traverse IP. So, the question becomes: Which type of encryption should I use, SNA or IP?

SNA certainly supports encryption and authentication, and can be used to protect the entire LU-LU session path. However, everything is done using manually defined symmetric keys. The passing of the key, refreshing and distribution takes a large amount of manual effort and must be guarded from hackers.

IP security such as *IPSec* is limited by the length of the LU path it can protect. However, an administrator can take advantage of dynamic key distribution, refreshing, or standard cipher key protections through the use of certificates and protocols such as the Internet Key Exchange (IKE). IP security can also take advantage of AES encryption that is quickly becoming the new standard for encryption ciphers compared to SNA, which can support only up to Triple DES encryption at the time of writing this book.

Using encryption from the IP or SNA side, or a mix of both, is the choice of any organization's risk governance on the requirements of securing their SNA data and network.

For more information about network cryptography on z/OS, see Chapter 2, "Cryptography for network security" on page 37.

4.3 Security controls using VTAM start options

VTAM has several *start options* relevant to security. They can be divided roughly into two areas:

- ▶ Crypto-based start options
- ▶ Access control start options

4.3.1 Crypto-based start options

The start options that fall under this heading are ENCRYPTN, ENCRPREF, VERIFYCP, and SECLVLCP. The latter two, VERIFYCP and SECLVLCP, are described in 4.7, "Application security" on page 149, because they involve session partner authentication, although for CP-CP sessions. Use these options if there is a requirement to verify the authenticity of any partner CP that is currently connected to the node and the link is not an Enterprise Extender connection where the stronger IPsec authentication can be used.

ENCRYPTN determines whether VTAM supports cryptography and which cryptographic API it expects to use. ENCRYPTN should be set to ENCRYPTN=CCA in the ATCSTRXX member. This will allow VTAM to use Triple DES encryption, which is the recommended cipher suite. The value of CCA also means that, upon initialization, VTAM will attempt to detect what crypto-products are available, and decide the level of encryption that will be supported. Before the activation of the LPAR you must have installed the cryptographic hardware and had your service group define it properly on the System z hardware. Also ensure that ICSF is properly configured and active on each LPAR where encryption is to be used. As a general practice, it is best to start ICSF before any other subsystem.

ENCRPREF merely defines a prefix used for the labels of the master symmetric keys used to generate the session keys. Its default is *no prefix* and there is no reason to change this.

4.3.2 Access control start options

These start options establish default behavior for permitting or denying access to the system from remote SNA resources. These options can be regarded loosely as the SNA equivalent of an IP packet filter, bearing in mind the difference between connectionless IP and connection-oriented SNA. Many of these options have the characters **DYN** within their names. They can be overridden on individual connections by means of VTAM definitions, but the start options allow you to define the SNA equivalent of that *deny all* filter that protects against anything you forgot to think about.

There are seven options to be considered, roughly falling into three categories:

- ▶ APPN-related options are DYNADJCP and CDRDYN
- ▶ Subarea-related options are CDRDYN, DYNASSCP and SSCPDYN
- ▶ LEN-related options are DYNLU, ALSREQ and CPCDRSC

CDRDYN controls whether this VTAM is allowed to create dynamic CDRSCs that represent resources owned by other nodes in the network. CDRDYN can be specified as a start option or on the host CDRM definition; if the CDRDYN start option is specified, then it overrides the CDRDYN value specified on the host CDRM definition. CDRDYN is set to YES in virtually all VTAM installations because of the huge amount of work that would be involved in predefining all remote session partners as CDRSCs.

DYNADJCP (defaulting to YES) allows VTAM to define adjacent control points (CPs) dynamically. The behavior of this option can be overridden on individual link station definitions. If DYNADJCP is set to NO, each adjacent CP to which VTAM connects must be defined in an ADJCP major node. You need to balance the security requirements against the effort in coding definitions. Code DYNADJCP=NO as a start option, override it on those link stations (for example, Switched major node PU) where you are sure of the identity of the nodes that are native to VTAM, and code an ADJCP major node for any remaining valid partners.

DYNLU controls whether VTAM allows resources represented by dynamically created CDRSCs to use peripheral (type 2.1) links. If DYNLU for a link is coded to NO, either at the start option level or on the link definition, and then you must code Cross Domain Resource (CDRSCs) definitions on that host for every cross-domain resource that is allowed to use peripheral (type 2.1) links for sessions. Although this might sound like a good way to control the use of external links, this method will work only when ISR routing is used on the link. If you allow HPR, which is recommended for APPN links, the value coded for DYNLU is not an effective security control.

SSCPDYN allows VTAM to add a known partner CDRM to any adjacent SSCP table if that partner sends in a session request. DYNASSCP lets VTAM create adjacent SSCP tables dynamically. Unlike the APPN case, there is no way that VTAM will search a partner CDRM unless that CDRM has been predefined. To minimize the likelihood that one of those partners has been hacked and now contains a bad LU, set both of these start option to NO to ensure each resource is searched ONLY in the partner networks where it is expected to be found.

ALSREQ and CPCDRSC are only useful when you have LEN connections to this VTAM, which should be a rarity these days. ALSREQ (default NO) allows VTAM to accept session requests from a remote LU that has not been predefined with the correct link station name. CPCDRSC (default NO) allows VTAM to establish outbound sessions to an adjacent LEN CP that has not been predefined. Because of the lack of authentication methods for LEN resources (there are no CP-CP sessions), optimum security demands that everything is predefined (ALSREQ=YES and CPCDRSC=NO) unless there is business justification for doing otherwise.

DYNPU and Connection Networks

One other definition in this category, although not implemented as a start option, is DYNPU. DYNPU (coded as a keyword online group definitions, and usually defaulting to NO) allows a remote link station to be created dynamically without matching a switched major node PU definition. For EE connections, where most inbound switched PUs are found these days, set DYNPU as NO and predefine all connection partners.

A Connection Network is exempt from the DYNPU rules by design. However, a Connection Network cannot be used to carry CP-CP sessions, so a *locate* request for a session would never flow over this type of link. This requires that any node participating in the Connection Network will always have been predefined by another switched major node to its NN server and thus can be authenticated by that server. Also, due to the nature of Connection Networks, you can control the network flow using IP Security controls such as encrypted tunnels or IP packet filters.

4.4 Transport security

In this section, we describe the three most common ways that SNA communication occurs within today's enterprise IT environments.

4.4.1 Enterprise Extender

Enterprise Extender is the latest technology available to enable organizations to transport SNA traffic over an IP network. Enterprise Extender is not a product but an extension to the Advanced Peer-to-Peer Network (APPN) and High Performance Routing (HPR) protocol. Enterprise Extender technology provides an encapsulation of SNA application traffic within UDP datagrams by HPR-capable devices at the edges of an IP network. To the IP network, the SNA traffic is broken down into UDP datagrams that are transmitted through the IP backbone. To the user, it is a normal SNA session with the same Class of Service (COS) as in a traditional SNA network. By wrapping the SNA application traffic in this way, Enterprise Extender enables SNA data to be carried over an IP backbone without changing either the SNA application or the IP infrastructure.

Types of Enterprise Extender connections

There are two types of Enterprise Extender connections:

- ▶ Simple Enterprise Extender connection (static or dynamic PUs)
- ▶ Connection Network

Regardless of the type of Enterprise Extender connection defined on z/OS, each requires the creation of a VTAM External Communication Adapter (XCA) major node to identify the following essentials, at a minimum:

- ▶ UDP Ports for Enterprise Extender traffic are located at port numbers 12000-12004 in the stack and should not be changed.
- ▶ Type of Service (TOS) settings in the IP header (defaults are 20, 40, 80, C0) that are used to define the SNA Class-of-Service defined by the application as the SNA traffic crosses the IP network. The IP routed network must honor the IP TOS settings (as does the OSA Express in QDIO mode) to preserve the priority of the SNA traffic; otherwise, there is no priority.
- ▶ Logical Data Link Control Timers are timer intervals (in seconds) to determine how long to wait before considering an Enterprise Extender connection inoperative.

Simple Enterprise Extender connection

A simple Enterprise Extender connection on z/OS is defined with a VTAM switched major node using either static (predefined) PU definitions or dynamic PU definitions. When comparing the two strategies, static PU definitions are the most secure but do require more initial effort by the system programmer. Table 4-1 shows a comparison of static and dynamic PU definitions.

Table 4-1 Static versus dynamic definitions

Static (Predefined) definitions	Dynamic definitions
Individual definitions to each remote Enterprise Extender endpoint which include remote CPNAME	No individual PU definitions are required
More secure	Less Secure
Ability to specify unique naming convention for SNA resources	Limited ability to specify naming convention for SNA resources
Flexibility to define different link characteristics per remote peer	Less flexibility to define link characteristics
Ability to use TG numbering to identify remote partner	No control over TG numbering before System z V1R10
Might require large number of definitions to configure and maintain	Minimal definition

By using the static method, the system programmer can define the expected CPNAME, TG number, and link characteristics for each remote Enterprise Extender endpoint. Only those connections calling in matching those traits will be successful and thus provide some level of security. Static definition is the most common Enterprise Extender implementation method.

Although the dynamic method requires the least amount of configuration, it also provides the least amount of security to validate the remote Enterprise Extender endpoint.

Enterprise Extender Model major node

If the dynamic method is selected for defining Enterprise Extender PUs, consider deploying a *model* major node. A model major node provides the ability to define specific TG characteristics of the dynamic PU (such as assigning the TG number), DISCNT=NO to preserve the EE physical link after the last session terminates, and when the link fails, automatic re-dial will occur.

Also, due to the dynamic nature of the connection from the SNA perspective, it might be prudent to add some IP controls. These could be in the form of IPsec policy filters or virtual private networks (VPN) to ensure that the proper IP addresses of the incoming Enterprise Extender connections are allowed to connect to this node.

Connection Network

A Connection Network is an Enterprise Extender implementation across a shared access transport facility, or *SATF*, (that is, an IP network) involving a common virtual routing node (VRN) for communication. Each participant in the Connection Network defines the same VRN name. When a session request is initiated, an Enterprise Extender dynamic link is created as needed between the endpoints if they share the same SATF. The benefit an organization receives by using a Connection Network is the linear growth of definitions in a fully meshed network compared to exponential growth. In other words, Connection Network requires $2n$ definitions, but the alternative method requires $n(n-1)$ with n nodes.

However, this dynamic environment can be used because there are no controls on who can connect to a Virtual Routing Node. As mentioned in 4.3.2, “Access control start options” on page 133, there are several methods in both IP and SNA to control access to the host through a Connection Network. These controls are discussed in detail in the subsections that follow.

4.4.2 UDP/IP considerations

Enterprise Extender encapsulates the SNA data into UDP datagrams to route across the IP network. For each IP stack acting as an Enterprise Extender endpoint, configuring the following changes are recommended for UDP.

- ▶ Reservation of UDP ports 12000-12004 for the Enterprise Extender traffic
- ▶ Sufficient UDPRCVBUFRSIZE and UDPSENDBFRSIZE sizes (recommended to use 65535 for both)

4.4.3 Network Address Translation considerations

Network Address Translation (NAT) introduces additional complexity to an Enterprise Extender solution in that each endpoint must establish connectivity to a static IP address and therefore, only static NAT is supported. Dynamic NAT is not appropriate because an IP address needs to be determined at configuration time. If a NAT boundary is associated with an Enterprise Extender Connection Network path, hostname-based Enterprise Extender definitions are required to establish network connectivity because the EE Connection Network protocol carries IP identities within the payload.

4.4.4 Enterprise Extender IP security

Enterprise Extender allows for SNA data to be encapsulated within UDP datagrams and transferred over an IP network. As a result, Enterprise Extender traffic is exposed to the same threats as all other IP traffic. There are two methods that can be used to control the UDP/IP traffic:

- ▶ Policy Filters
- ▶ IPSec

The simplest way is by using a packet filtering firewall to only allow EE UDP datagrams in from particular IP addresses for UDP ports 12000-12004. This can be done from the router or at the host using the IPSec Policy Filters. These filters use information in the IP and UDP packet headers to either permit or deny the traffic coming into the enterprise. Usually this is deployed when an encrypted data link is already deployed; usually through a third-party ISP, to connect two organizations. Because there is no need to encrypt or authenticate the traffic coming across this link, packet filtering is sufficient. However, because the security end point is moving closer and closer to the end points of a connection, simple policy filters might no longer be sufficient.

To truly authenticate and encrypt the Enterprise Extender traffic from both endpoints, IP Security via IPSec is the recommended solution. IPSec is an industry standard protocol that provides end-to-end authentication and encryption. It is available on multiple platforms, including but not limited to, z/OS, iSeries, Linux on System z, Windows, and network routers. Using IPSec, the Enterprise Extender traffic can be protected as it flows over the unsecured part of the network or the complete EE connection path.

To enable IPsec on z/OS Communications Server, a Policy Agent daemon (PAGENT) started task must be configured with a set of policies. The IPsec policies define the IP addresses of the virtual private network (VPN) endpoints, UDP ports for application data transmission, cipher suites used for encryption and authentication, the location of the server and certificate authority (CA) certificates, and when encryption keys should be refreshed. The server and CA certificates enable the VPN endpoints to authenticate their identity and to thus negotiate a dynamic tunnel connection for encrypting the Enterprise Extender traffic. After the dynamic tunnel is established, all Enterprise Extender traffic transmitted within the VPN will be protected from unauthorized access. This is the recommended way to protect Enterprise Extender links.

4.5 TN3270 Security

This section deals with the most common type of user connection into an environment, the TN3270 connection. We explain the ways a system programmer can modify the TN3270 server to create an environment that meets their security requirements.

4.5.1 Background

The most common way to connect from workstations to SNA applications in a traditional enterprise is to use a 3270 session. This connection in the past was made via hardware devices like a 3290 through a front-end processor such as a 3745 and into the MVS host.

This type of connection, by its nature, provided a great deal of physical security because these links were protected within the infrastructure of a building. However, as time went on and the requirement for greater availability to applications grew, the TN3270 emulation was created. This is a session that runs over a TCP/IP network connection and can take the form of anything from a basic green screen to a complex set of web pages created by IBM Rational Host Application Transformation Services (HATS). Although this has done wonders in modernizing how enterprise networks can scale their business, it has also opened up security issues within an enterprise.

Because the 3270 emulation mimics the same data flows as the older hardware, it also took on many of the assumptions that the hardware did. The 3270 emulation passes many critical pieces of data in the clear, including user ID and password information. Also, this emulation at its base has no way of authenticating where the session originated from. These were reasonable assumptions, because before TN3270, all connections were hardware-based, the origins of these sessions were well known, and the transmissions from these terminals could not be easily captured. In today's environment of shared Ethernet LANs and wireless networks, the assumptions made back in the 1960's and 70's no longer holds good.

4.5.2 Securing TN3270 IP flow

It was critical that security technology be deployed within a TN3270 session flow to replace the security that was lost in this new environment. The Secure Sockets Layer (SSL) protocol was enabled within the TN3270 server to provide the ability to add encryption and authentication for a session. The SSL-enabled TN3270 server protects all data between the server and TN3270 SSL-enabled clients (or terminal emulators) such as IBM Host OnDemand and IBM Personal Communications.

4.5.3 SSL/TLS support

The SSL/TLS protocol can provide data encryption, data origin authentication, and message integrity for TCP applications in a network. This is accomplished using X.509 certificates, which can be used to trade an encrypted session key using asymmetric encryption. Figure 4-1 shows a high-level view of how SSL/TLS negotiates a session between an enabled host and client. Properly negotiated SSL sessions will, by default, authenticate the server to the connecting client. Optionally, the server can be configured to request a certificate of the client to authenticate itself to the server. Also during the handshake, security session parameters, such as cryptographic algorithms, are negotiated and session keys created. After the handshake, the data is protected during transmission with data origin authentication and optional encryption using the session keys.

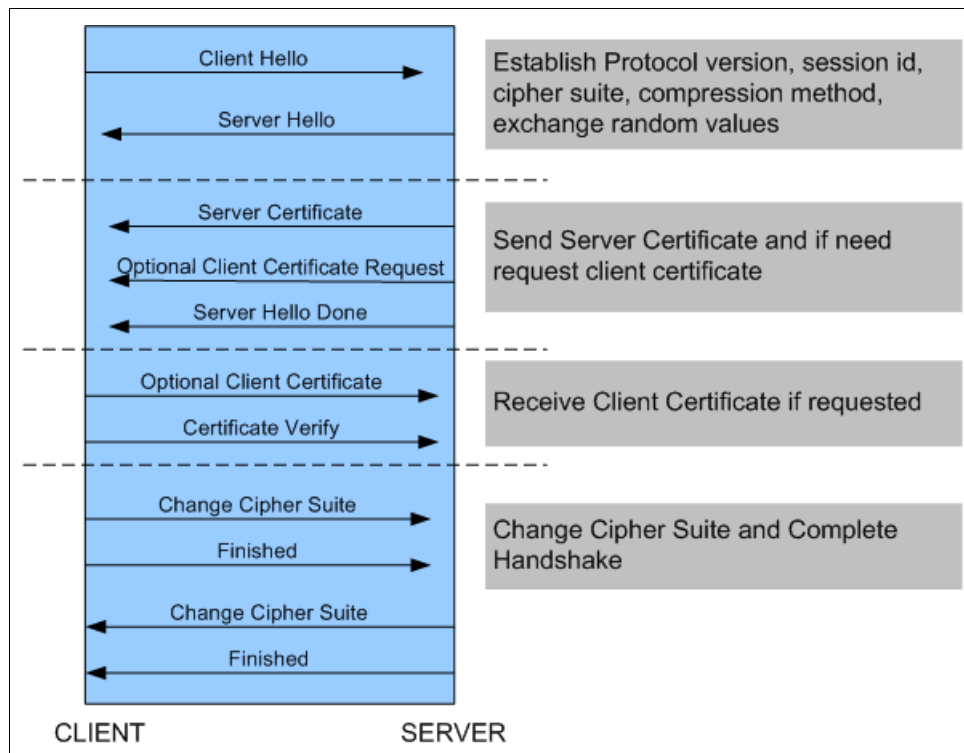


Figure 4-1 Typical SSL negotiation

The cryptographic algorithms that are used for an SSL session are based on the algorithms the server and client negotiate during setup time. During the SSL handshake, the client and server exchange a list of algorithms. The algorithm selected is based on the best match between the client's list and the server's list. The selectable algorithms can be limited by configuring a subset of allowable algorithms at the server. Servers can support encryption using Triple DES or other encryption algorithms (RC2, RC4, DES, and AES). A hardware crypto coprocessor, if available, is used for DES, Triple DES, and AES encryption.

4.5.4 TN3270 SSL support

SSL/TLS support for the z/OS TN3270 server can be implemented either within the server itself, or using an AT-TLS policy with the Policy Agent, which is considered a best practice. In each case, the function is much the same but the definitions are coded differently.

Each type of SSL/TLS support provides two different forms of the SSL/TLS negotiation.

- ▶ Server-side SSL/TLS
- ▶ Client authenticated SSL/TLS

The major difference between them is whether a certificate is required to authenticate the client to the server. In most cases, the certificate distribution for client-side authentication is an administratively heavy process and the normal user ID logon for applications is enough. However, it is a really strong authentication technique and goes a long way towards satisfying security standards such as SOX, PCI DSS, or HIPPA. The server can also dictate the level of encryption and in the case that the server should not allow any encryption below the Triple DES encryption.

Currently, there are three versions of SSL/TLS negotiation:

- ▶ TLS
- ▶ SSLv3
- ▶ SSLv2

Although TLS and SSLv3 are extremely similar, the SSLv2 protocol is much older and not considered as secure. Within the server, there are controls to limit what the client can request relating to the version of SSL negotiation should be. Ensure not to allow SSLv2 negotiation by using the NOSSLV2 option.

An architectural overview of the TN3270 SSL negotiation is depicted in Figure 4-2.

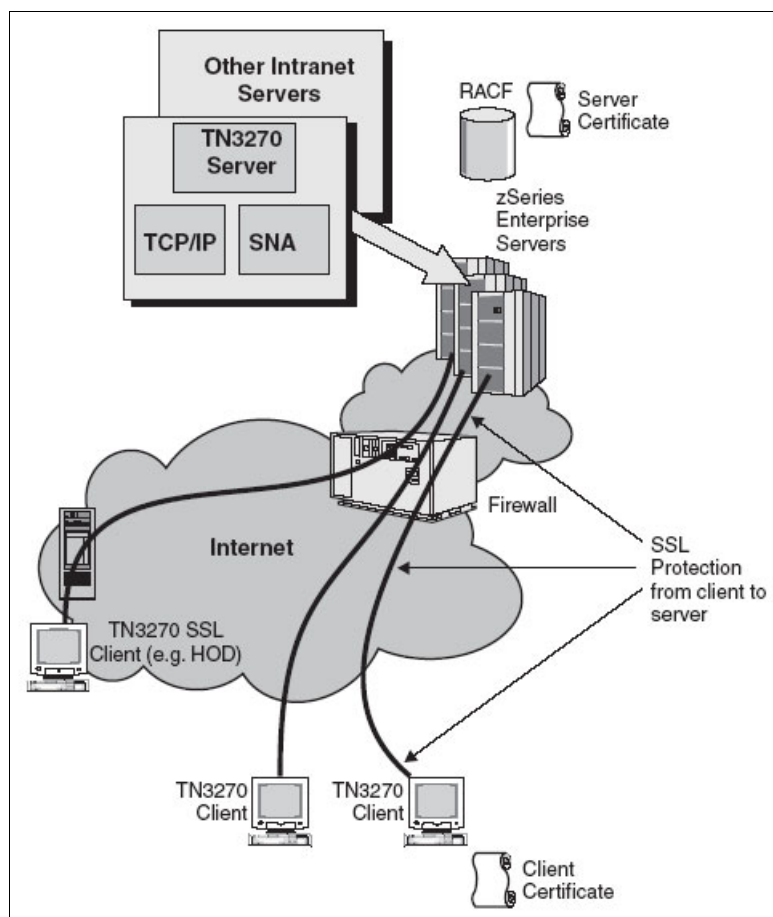


Figure 4-2 TN3270 SSL negotiation

Client access controls

The TN3270 server on z/OS has extra controls to secure access across the SNA network. Using a mixture of LUNAME, IPGROUP, and PORT statements, a system administrator can control which applications a user can access. For example, if you wanted to ensure only a particular set of users can access a sensitive application, you can design your TN3270 server to allow access only to the application via a specific TCP port on the stack. You can add security to the port using the SSL/TLS support mentioned previously to ensure the confidentiality and security of the application.

4.5.5 Securing DLSw connections

Many organizations that still use SNI (FID4 connections) or boundary (peripheral) links to devices might use DLSw to communicate between organizations. Although this is a good way to communicate between two different NetIDs, SNI traffic does flow across a TCP connection. To protect the data as it traverses this link, use either IPSec VPNs or a hardware encrypted channel.

Another possible security exposure can be introduced if a DLSw router is enabled into passive mode; this type of configuration potentially allows an attacker to connect to the DLSw router as a peer. The attacker could gain access into the SNA network without having to authenticate their system to the router. Either all DLSw peers should be pre-coded or IPSec AH or ESP NULL encryption with authentication VPNs should be used to authenticate the peers at the IP level.

4.6 Searching security

Although connecting different business ventures is critical in the current business environment, it also has added significant obstacles in dealing with securing those connections. Within SNA, these obstacles involve controlling searches from going out and coming into the corporate network. In this section we describe techniques that you can use to control the searching behavior for applications within an SNA network.

4.6.1 Basics of searching

Because most SNA networks are a mix of subarea and APPN environments, it is important to understand how VTAM conducts *searches*. Where does a search originate from within the network? Does it come from the subarea portion of a network or does it start from an APPN node? The answer to these questions will determine how the search will be conducted and what options will come into play.

4.6.2 Subarea searches

When a search for a resource comes from a subarea environment, VTAM will use two start options to determine how the search will be conducted - SORDER and SSCPORD. The table in Figure 4-3 describes the relationship of how these two options effect the overall searching for a resource within the subarea environment.

SORDER				
	APPNFRST	APPN	ADJSSCP	SUBAREA
SSCPORD	1. APPN Network 2. Learned Owner 3. Coded Owner 4. Prev. Successes 5. ADJSSCP Table 6. Prev. Failures	1. Learned Owner 2. Coded Owner 3. APPN DS DB 4. Prev. Successes 5. APPN Network 6. ADJSSCP Table 7. Prev. Failures	1. Learned Owner 2. Coded Owner 3. APPN DS DB 4. Prev. Successes 5. ADJSSCP Table 6. Prev. Failures	1. Learned Owner 2. Coded Owner 3. APPN DS DB 4. Prev. Successes 5. ADJSSCP Table 6. Prev. Failures 7. APPN Network
PRIORITY				
DEFINED	1. APPN Network 2. Learned Owner 3. Coded Owner 4. ADJSSCP Table	1. Learned Owner 2. Coded Owner 3. APPN Network 4. ADJSSCP Table	1. Learned Owner 2. Coded Owner 3. APPN DS DB 4. ADJSSCP Table	1. Learned Owner 2. Coded Owner 3. APPN DS DB 4. ADJSSCP Table 5. APPN Network
Prefers APPN ←————→ Prefers Subarea				

Figure 4-3 SSCPORD & SORDER table

Although SORDER and SSCPORD control the order in which certain searches are performed, they do not, for the most part, control the contents of the search list. The contents of the search list are controlled by the adjacent SSCP tables that are defined to VTAM and by the SSCPDYN and DYNASSCP start options. For optimum security, define exactly what nodes are to be searched in the adjacent SSCP tables, and code two start options as N0 to prevent VTAM adding any unwanted nodes into the search order.

In the table show in Figure 4-3, notice that if the search is started in APPN and is now being performed within the subarea environment, all of the APPN searches are ignored.

4.6.3 Searching an APPN network

If a resource needs to be discovered within an APPN environment, a *locate* request is sent into the network and is either answered by the owning network node of the resource by a specialized network node called a central directory server (CDS), or by a network node that is connected to another NetID, called a border node.

In 4.3.2, “Access control start options” on page 133, we explain the importance of using the DYNADJCP option in controlling which nodes will be allowed to create CPCP sessions with VTAM. However, this does not address how you control the searches within an APPN network. Consider writing a Directory Services Management Exit (DSME) for any type of APPN search that can occur in a SNA network. It is rare that one is concerned with the searching that occurs within the native environment. More important is the ability to control the searches that come in from and go out to other organizations’ networks.

4.6.4 Controlling searches of other APPN networks

When searching other networks, a special type of network node called an Extended Border Node (EBN) is used. The EBN can make connections and initiate searches into other NetIDs. In most practical applications of SNA, EBNs are used to connect different enterprises together for some shared business purpose. Ensuring that an enterprise sends appropriate searches to the correct customer networks is an important issue.

You can have a situation as depicted in Figure 4-4 where a company is connected to two different business partners.

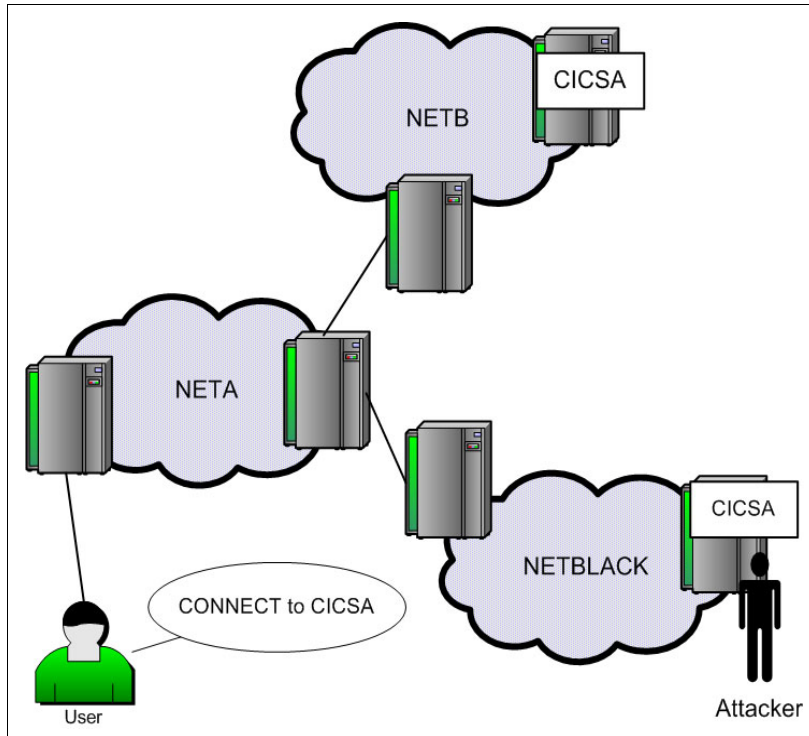


Figure 4-4 Searching example

If searches are not sent carefully, a user within the organization could think that they are connecting to an application on a particular business' network. However, the user is really connecting to an attacker's application that is used to gather data, such as account numbers and passwords. In this section, we explain how to prevent this and other attacks by tuning your VTAM settings.

Tuning options for searching other APPN networks

There are two APPN-specific start options that are used to control cross-network searches in an APPN network; BNDYN and BNORD. BNORD in an APPN network is similar to SSCPORD for SNI connections. Depending on what was coded, VTAM could take previous successful searches and uses this to then reorder the searching sequence. Although this option has a minimal effect when it comes to the overall security of the SNA network, it is recommended that BNORD=DEFINED is coded.

The BNDYN option has a big effect on how your searching will occur and thus must be looked at in a security context. When APPN searches for resources in different NetIDs, VTAM creates a table called an Adjacent Cluster Routing table for each NetID to control the searching. BNDYN defines how and if VTAM will add nodes dynamically to this table. There are three settings for the BNDYN option:

- ▶ When BNDYN=NONE is specified, there will not be any entries in the routing table except those explicitly coded within a predefined adjacent cluster routing list
- ▶ When BNDYN=LIMITED is specified, the Border Node automatically adds routing table entries if one of the two conditions have been met;
 - BNs and non-native network nodes that VTAM learns about, with a NetID that matches the NetID of the desired resource
 - Any node which performed a search into the network that originated from another NetID that matches the DLU's NetID of the current search.
- ▶ When BNDYN=FULL is specified, all active border nodes in the native subnetwork and active border nodes and peripheral network nodes in non-native subnetworks attached to this node are automatically added to the routing list

Your BNDYN setting can have a dramatic effect on where your searches go. For example, having BNDYN=FULL would allow APPN searches to go to every known Border Node that is connected to the corporate NetID. Looking at Figure 4-4 on page 143 again, if the option is set to BNDYN=FULL there is a chance that NETBLACK will be searched before NETB. This could enable a session between the attacker's CICS application and user session to allow a masquerade attack. Set BNDYN to NONE to control searching by using an ADJCLUST table, which is describe in the next section.

4.6.5 ADJCLUST tables

Use an adjacent cluster routing definition list (ADJCLUST table) to customize routing between different APPN NetIDs. When a border node cannot satisfy a search request within its own domain, it will use the ADJCLUST table to determine which cross network domains to search.

Separate adjacent cluster tables can be coded for each individual NetID that an organization is connected to (see the example in Figure 4-5 on page 145). This can include setting up searches to go through intermediate networks which allow two organizations to connect without having a direct connection. Below is a simple example of an organization that uses NETA as its NetID with a border node connected to two different NetIDs (NETB, NETC).

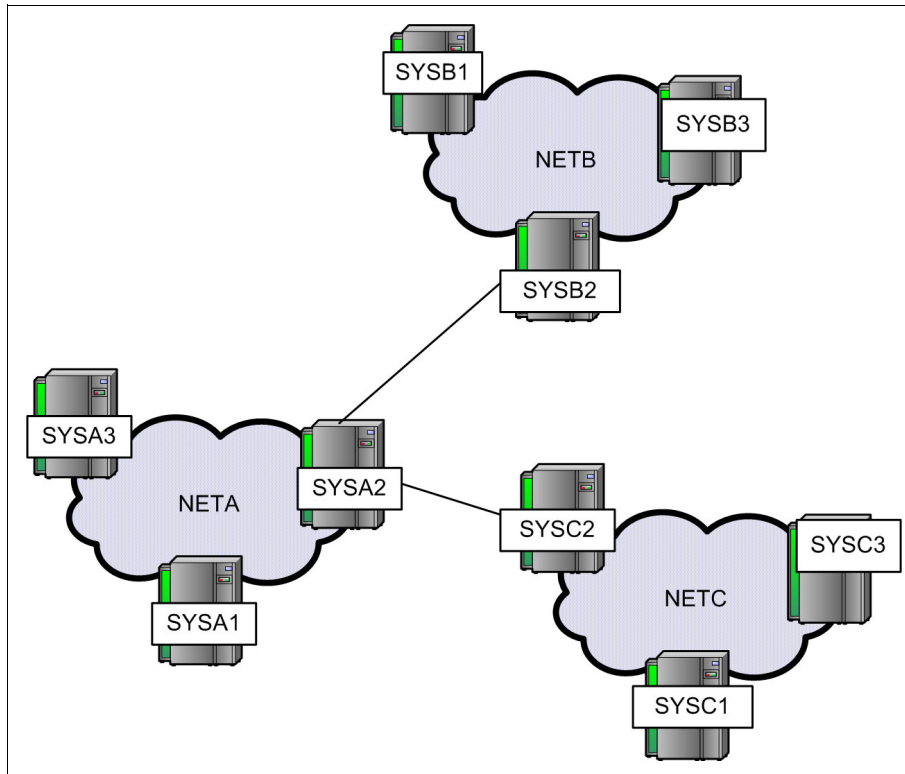


Figure 4-5 Basic network design

Example 4-1 is an example of an ADJCLUST table that could be coded on SYSA2.

Example 4-1 ADJCLUST table

```

*****
SAMADJCL VBUILD TYPE=ADJCLUST
*****
* DEFAULT NETWORK ID
*****
NONET NETWORK SNVC=4, ALLOW DEPTH OF 4 NETWORKS
BNDYN=LIMITED ALLOW LIMITED DYNAMICS
ASYS2 NEXTCP CPNAME=NETA.SYSA2
BSYS2 NEXTCP CPNAME=NETB.SYSB2
CSYS2 NEXTCP CPNAME=NETC.CSYC2
*****
* ROUTING FOR NETID=NETB
*****
NETB NETWORK NETID=NETB,
BNDYN=LIMITED,
SNVC=4 ALLOW DEPTH OF 4 SUBNETS
BSYS2 NEXTCP CPNAME=NETB.SYSB2
*****
* ROUTING FOR NETID=NETC
*****
NETC NETWORK NETID=NETC,
BNDYN=LIMITED,
SNVC=4 ALLOW DEPTH OF 4 SUBNETS
CSYS2 NEXTCP CPNAME=NETC.SYSC2
*****

```

In the ADJCLUST table, you can override the BNORD, BNDYN, and SNVC parameters for each NetID the APPN network is connected. For searches originating in SYSA1 without a qualified NetID, all the nodes attached to SYSA2 will be searched. This could allow a black hat to insert their SNA application into a network and have users unwittingly connect to that application in a network beyond the reach of their business.

Network-qualifying searches

Creating an Adjacent Cluster table for a Boarder Node that prevents unqualified searches from leaving the native network, carries with it a responsibility to ensure that valid searches are qualified with the correct NetID. This has the additional benefit of optimizing the search patterns. There are two ways to achieve this:

- On every node, define a CDRSC with a NetID for every cross-net target (same-net CDRSCs do not need to be defined). This ensures that all cross-net searches originating in this node carry a NetID, and thus fall under the control of the ADJCLUST table for that NetID rather than a default table. For example:

```
VBUILD TYPE=CDRSC
NETWORK NETID=USIBMN92
LU925 CDRSC
LU926 CDRSC
LU996 CDRSC
```

- On the network nodes only, define a CDRSC with a NetID and CP name. This creates an APPN directory entry. The value of NQNMODE also needs to be NAME rather than NQNAME, but this is usually so because of the default start option value which is NAME. This technique ensures that alias (unqualified) searches from served end nodes get their NetID added by the NN server before any further searching is performed, and therefore use the correct ADJCLUST table when they reach a border node. For example:

```
VBUILD TYPE=CDRSC
NETWORK NETID=USIBMN92
LU925 CDRSC CPNAME=EN587
LU926 CDRSC CPNAME=EN594
LU996 CDRSC CPNAME=NN556
```

The CPNAME value does not need to be correct, although you will save an extra search by getting them right. What matters is the creation of an APPN directory entry, because that is the first place the NN server looks when it receives a search request from a served end node. If the coded values are not correct, the search will fail, but the subsequent broadcast will succeed and will update the values in the directory entry.

Which of these two methods you use depends on your network topology and the relative amounts of effort needed to implement each. The first method works for searches originating from this node or received over a subarea connection. The second method works for all searches, including those received over APPN.

4.6.6 Controlling searches entering a network

Although controlling searches originating in your network is important, it is even more important to control searches destined to your network. Each enterprise needs to protect their network from unwanted searches. In the past, this was manually done using one of two exits (SME or DSME) to control the searching. However, in recent releases of z/OS, some of the functions in these exits have been incorporated into z/OS Communications Server.

4.6.7 Session Management Exit

The Session Management Exit (SME) is a multifunction exit routine you can use to control and manage LU-LU session-related functions. You can use the exit to authorize session establishments, obtain session accounting data, and better manage SSCP and GWPATH selection. In addition, VTAM can invoke the SME exit for the following functions:

- ▶ For each session establishment before any cross-domain flows
- ▶ For each session establishment after the destination resource for the session has been determined
- ▶ To determine ordering for the adjacent SSCP tables to try for each cross-network session establishment
- ▶ To translate the name, determine the owning SSCP, or translate CoS and logo mode (some of these functions are for cross-domain session establishments, others are for cross-network session establishments)
- ▶ To determine the appropriate ALS to use as the connection for an independent LU that is the destination LU (DLU) for the session
- ▶ To modify the virtual routes (VRs) and the associated transmission priorities (TPs) that are to be used in session establishment or RTP pipe setup

This exit will be called only during subarea-side function where the session awareness is maintained and only when a session between the OLU and DLU will take place. VTAM can invoke the session management exit at certain points in processing such as:

- ▶ When VTAM initialization has completed
- ▶ When normal VTAM termination occurs
- ▶ During XRF session switch
- ▶ SSCP takeover
- ▶ MNPS recovery

4.6.8 Directory Services Management Exit

The Directory Services Management Exit (DSME) is similar to the SME exit. It is used to authenticate and control APPN searches, whereas the SME exit is used for Subarea searches. It has many of the same base functions that the SME exit has, but for APPN searches, instead. For example:

- ▶ Initial Authorization for an LU search
- ▶ Border Node Selection
- ▶ Re-drive of Authorizations for LU-LU search

The DSME exit can also control which session searches are allowed, depending on the following criteria, among many others:

- ▶ OLU name or NetID
- ▶ Whether the search is fully qualified or not
- ▶ The adjacent CP name or NetID in the OLU direction of the node performing the search

4.6.9 Searches that are not network-qualified

Use the Adjacent control point (ADJCP) major node to control how adjacent CPs can connect to this VTAM. One of the parameters is the ALIASRCH key word. This controls whether an adjacent node from another NetID is allowed to send not fully qualified locates into a border node on your network. If ALIASRCH is set to NO, the adjacent node will not be allowed to send non-qualified searches into the enterprise network. This requires you to create an ADJCP table entry for each of the cross-network connections. However, as a general rule, it can be useful as an aid to keep track of the nodes the enterprise should be connected to.

4.6.10 Authorized Cross-Net searches

The AUTHNET option on the adjacent control point table allows you to limit what NetIDs can be searched through your network.

You must add only the AUTHNET parameter to an ADJCP definition with a list of NetIDs that can be searched for by the adjacent node being defined. This would block the ability of a black hat to use an intermediate network to perform any searches into another network unless they are explicitly allowed. For example:

In Figure 4-6 on page 149, there are three networks connected to NETX for one reason or another. NETBLACK has a hacker that is attempting to access NETA or NETC resources even though there is no reason that a search should be allowed through NETX to either of these networks.

On NETX.SYSX1, the following ADJCP list has been predefined:

```
*****
SAMADJCP VBUILD TYPE=ADJCP
*****
* Allow NETA to search NETC C *
*****
SYSA2 ADJCP NETID=NETA,NN=YES,AUTHNETS=(NETC) X
ALIASRHC=NO
*****
* Allow NETC to search NETC A *
*****
SYSC2 ADJCP NETID=NETC,NN=YES,AUTHNETS=(NETA), X
ALIASRHC=NO
*****
* Do not allow NETBLACK to search any networks *
*****
SYSB2 ADJCP NETID=NETBLACK,NN=YES,AUTHNETS= X
ALIASRHC=NO
```

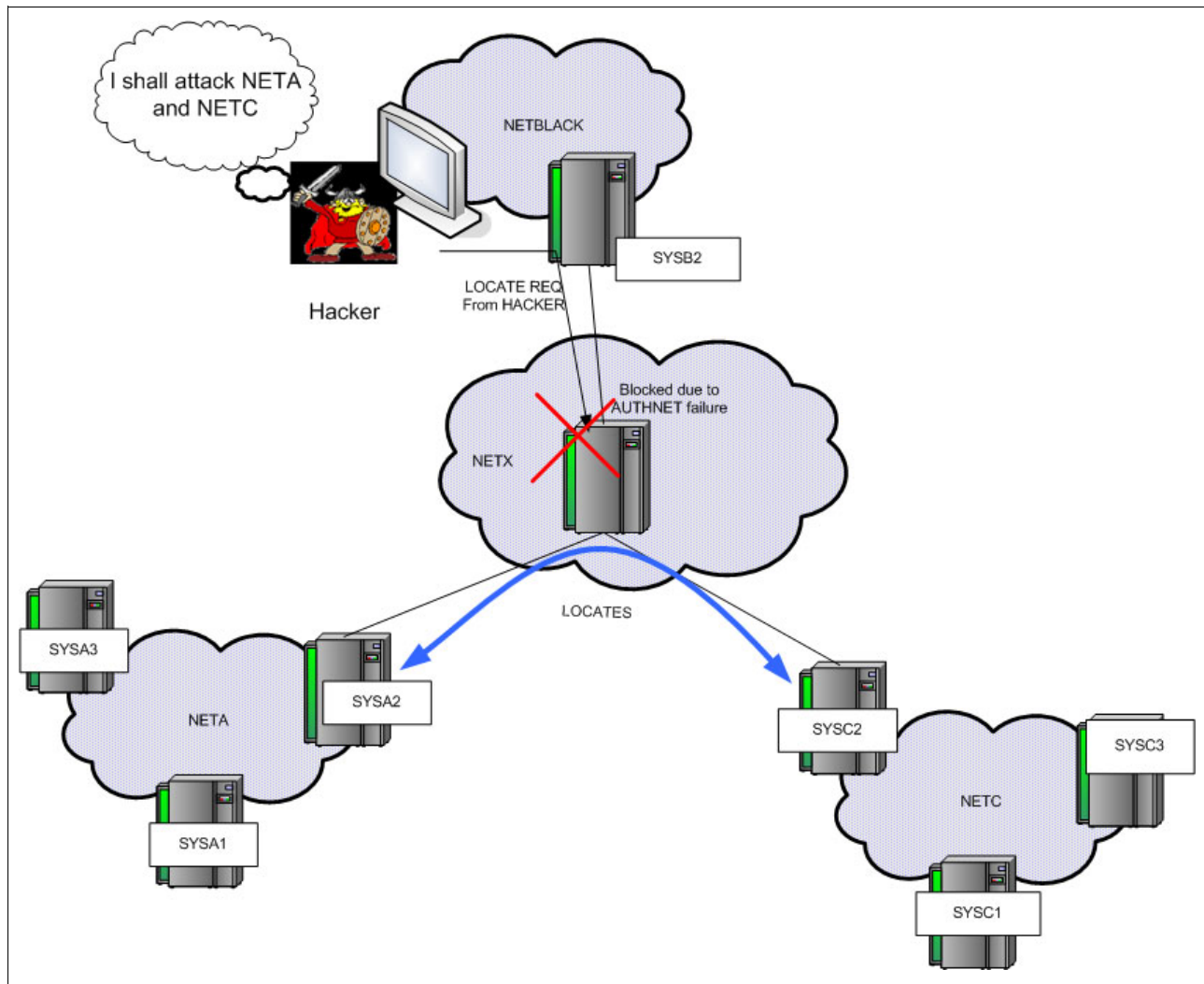



Figure 4-6 Example of Authnets

This ADJCP table prevents any search requests coming from NETBLACK but allows NETA and NETC to perform searches on each other.

4.7 Application security

SNA provides the same basic security functions as IP does, namely data confidentiality, data integrity and partner authentication. The major difference is that SNA has no asymmetric encryption, so the following conditions apply:

- ▶ Key distribution and key management must be done manually
- ▶ All partner authentications must be done via the possession of shared symmetric keys

Data confidentiality and data integrity work in similar fashion whether the session concerned is dependent LU (z/OS to z/OS) or independent LU 6.2 (where one partner does not need to be owned by z/OS). The same is true whether the API used is the *record* API or the APPCCMD API. Authentication is somewhat different in the LU 6.2 environment, where VTAM provides extra facilities for applications using the APPCCMD API.

4.7.1 Session-level encryption for data confidentiality

The shared symmetric key used in data encryption is exchanged securely between partners during the session setup flow. This typical example is the most complex case, namely an SLU-initiated dependent LU session:

- ▶ The administrator installs shared symmetric keys (only DES and Triple DES are supported) on each partner node. These are the master keys that are used to encrypt the session keys, and there should be one key per physical node. Each VTAM node will have installed:
 - Its own key
 - The keys for all partner VTAMs and independent LUs with which it is expected to communicate securely
 - The keys for all of the dependent LUs it serves that are capable of SNA encryption
- ▶ The VTAM definitions determine whether encryption is to be performed on a particular session. If it is, the SLU VTAM performs the following functions:
 - It generates a symmetric key for use in the session.
 - It encrypts this key by using one of the master keys installed by the administrator twice: Once using the key for the SLU node and once using the key for the partner VTAM node
 - It sends both the encrypted keys to the PLU VTAM in the CDINIT or *locate* request.
 - PLU VTAM then decrypts its copy of the session key but sends the other copy with the BIND to the SLU
 - The SLU device decrypts the session key in the BIND so that now both partners have the same key (the session data now flows in encrypted form).

This is the way it works within an up-to-date APPN network if you have installed the appropriate keys (for the SLU node and for the PLU's CP) in the SLU VTAM's key database. The locate or CDINIT requests flow end-to-end without any encryption or decryption on the session path, even if there are subarea hops on the path.

If the partner's key cannot be found, or if VTAM is a subarea node, the session key in the CDINIT must be decrypted at each intermediate node, and re-encrypted using the master key of the next node on the path. Thus, the adjacent node's key is required to be in the SLU VTAM's key database rather than the partner node's key.

Even if the network topology permits end-to-end crypto session setup, an alternative is to let the NN servers of the session endpoints take part in the process. The setup flows are then in three parts, with encryption and decryption being done at each NNS and at each endpoint.

The network configuration determines which session initiation method (end-to-end, host-by-host, or NNS-to-NNS) is used, based on the complexity of the network and the effort required to set up the key databases. End-to-end means less overhead in setting up the session but more key administration. Using host-by-host means more work in setting up the session but fewer keys defined in each host. Letting one or both NN servers take part is a compromise between these two options.

Whether the session initiation is done end-to-end or host-by-host, the actual session data always flows encrypted end to end. The whole idea of the initiation flows is to send the session key securely from SLU to PLU.

This scheme works for both record API and APPCCMD API. With LU 6.2, the actual encryption options are negotiated on the BIND and BIND response.

Setup of session-level encryption (SLE)

By default, VTAM supports SNA session encryption, but you can turn it off, or restrict the cryptographic products to be used, using the ENCRYPTN start option.

As to the session itself, the definitions that determine whether encryption is used are coded in the following places:

- ▶ The APPL definition of the PLU
- ▶ The APPL or LU definition of the SLU
- ▶ The mode table associated with the SLU

The APPL definition has two relevant keywords: ENCR and ENCRTYPE. ENCRTYPE is easy as the options are DES or TDES24 (triple DES). VTAM uses the higher of the SLU's and PLU's values. Triple DES is preferable because DES is no longer considered a secure cipher.

ENCR can have the following values:

COND	Try to secure every session, but set up a clear session if the negotiation fails.
REQD	Sessions must be encrypted. Session setup fails if the negotiation fails.
OPT	We don't care, so let the partner decide whether sessions will be secured or not.
NONE	There will be no encrypted sessions with this application.
SEL	The application will specify (using an appropriate API) which messages are to be encrypted. As with REQD, if the crypto negotiation fails then the session setup fails.

The LU definition has the same values and the same meaning for ENCR and ENCRTYPE, except that the SEL value is meaningless and thus not available. In addition, the LU definition has a CKEYNAME keyword that points to the master key associated with this LU. An application acting as SLU always uses its node's master key. The key database must have a suitable key filed in it under that name. The default for CKEYNAME is the LU name.

Use the mode entry associated with the SLU to override the values in the VTAM definitions. However, security cannot be decreased. For example, if the VTAM definition says Triple DES then you cannot override that with DES in the mode entry.

These are the keywords in the mode entry:

- ▶ ENCR defines a bit string that corresponds to the ENCR value in the LU or APPL definition.
- ▶ ENCRYP has only one valid value, namely TDES24. Omitting ENCRYP defaults to DES.
- ▶ CKEY allows you to use an alternate key name for this session. You can define alternate (master) keys for temporary use while the primary master key is being updated.

Installing keys

All of the previous setup tasks are business as usual for a VTAM systems programmer. What might be unfamiliar is the process of installing those master keys on the VTAM nodes.

If triple DES is to be used (the only serious option), a product implementing the CCA (Common Cryptographic Architecture), such as ICSF, must be installed. VTAM issues ICSF calls to generate, encrypt, and decrypt the session keys.

ICSF generates the master keys for you. The local copies of the keys (both for this node and for partners) are stored in the secure key database. At the same time, clear keys are produced for export. Thus, you can generate keys for all nodes on the same VTAM and simply export the clear versions to the desired places, where they too will be placed in the secure database.

4.7.2 Message authentication for data integrity

SNA message authentication works in almost exactly the same way as data confidentiality (encryption), except that rather than encrypting the message itself, it encrypts a hash value (message authentication code) appended to the message. The same algorithms (DES or triple DES) are available, and the same methods are used to generate the keys.

In fact, VTAM's message authentication offers an alternative to crypto techniques, namely a message digest created by an internal VTAM algorithm, rather than DES or triple DES. This alternative cannot be regarded as truly secure.

To invoke message authentication for a session or sessions, we again have additional keywords on the APPL definition (not LU definition) and the mode entry. On the APPL definition, the following meanings apply:

MAC	Specifies whether message authentication is required (MAC=REQD), preferred (MAC=COND) or not supported (MAC=NONE).
MACTYPE	The method of producing the message digest: MACTYPE=CRC (VTAM's home made code) or MACTYPE=DES (the real thing). If MACTYPE=DES, the value of ENCRTYPE determines whether DES or triple DES is actually used.
MACLNTH	The length of the message digest, with 4, 6, or 8 as the valid values for DES. The longer, the better.

On the mode entry at the SLU end, exactly the same keywords are used to override the MAC specifications for a given session.

4.7.3 LU 6.2 session-level authentication

In addition to the encryption and data integrity features already described, LU 6.2 resources have an extra authentication feature available for them. The partner LUs authenticate with each other by exchanging encrypted random data on the BIND and BIND-response. The encryption is done using DES (*not* triple DES) under a shared symmetric key.

The key itself (referred to as a *password* in the VTAM literature) is held in RACF, and you need to define the appropriate RACF profiles in the APPCLU resource class to hold the passwords. A password is required for each LU-LU pair that will engage in session level authentication.

Because APPN control points are LU 6.2 resources, the same principles apply to CP-CP sessions as for any other LU-LU sessions. This is a particularly useful feature when you need to be sure that nobody has introduced a *rogue* APPN node into your network, which is much easier to do in these days of ubiquitous EE.

Configuring LU 6.2 authentication

To protect a VTAM application program against an impostor partner, you need to:

- Choose an application program that uses the VTAM APPC API, not the record API. Applications that use the record API must code the authentication algorithms. CICS is the major LU 6.2 user of the record API.

- ▶ Code the VERIFY and SECLVL keywords as appropriate on the APPL definition, whether SLU or PLU or both. APPC=YES is a prerequisite.
- ▶ Code the appropriate RACF resource profiles

The VERIFY keyword determines whether authentication is required (VERIFY=REQUIRED), optional (VERIFY=OPTIONAL), or not supported (VERIFY=NONE).

Use SECLVL to indicate either the basic or the enhanced level of authentication is to be used. The choices are LEVEL1 (basic), LEVEL2 (enhanced) or ADAPT (pick the higher one of the two partners' choices). The difference between the basic and enhanced algorithm is that the enhanced algorithm sends an encrypted amalgam of three values (two random numbers and the SLU name) rather than just one random number.

If the sessions to be protected are CP-CP sessions, there are no APPL definitions. The VERIFYCP and SECLVLCP start options do the job of the VERIFY and the SECLVL keywords, respectively.

The name of the RACF profile depends on two factors:

- ▶ Whether this profile is for CP-CP sessions
- ▶ Whether the local LU supports network-qualified names (the ACB has NQNAMES=YES).

In this case, the name of the profile must be in this form:

`localNetID.localLUname.remoteNetID.remoteLUname`

If neither of those conditions is true, the profile has only three parts:

`localNetID.localLUname.remoteLUname`

Therefore, you would code these RACF definitions:

```
RDEFINE APPLCU NETX.VTAM1.NETY.VTAM2 UACC(NONE) +
SESSION(SESSION(X'8B44D208C1AA'))
```

Notice the use of the SESSION optional segment to define the DES key that is used in the authentication. The same key must be installed in the partner node.

Do not forget SETROPTS CLASSACT(APPCLU) to activate the class.

4.7.4 LU 6.2 conversation-level authentication

Session-level authentication is designed to verify the identity of a partner LU before an LU 6.2 session starts. However, an LU 6.2 session typically carries many conversations, each of which might represent a different individual user. Conversation-level security is designed to authenticate the user on any given conversation.

Conversation-level security is completely under the control of the LU 6.2 application. If the application uses the record API, VTAM has no involvement in the process. If the application uses the APPCCMD API, the SECACPT keyword of the APPL definition can be used:

- ▶ SECACPT=NONE means that conversation level security is not supported.
- ▶ SECACPT=CONV means that conversation level security is supported.
- ▶ SECACPT=ALREADYV means that the *already verified* option of conversation level security is allowed in addition to the CONV level. This is used when an application program passes an incoming request on to another program. The first program has already verified the user ID and password, so it tells the second program not to bother.

- ▶ SECACPT=PERSISTV means that the *persistent verification* option is supported. This allows two application programs to have multiple conversations with each other, but performs the verification only once.
- ▶ SECACPT=AVPV means that both ALREADYV and PERSISTV are supported.

Because conversation-level security implements a user ID and password, it is not to be regarded as secure unless the session is encrypted.

4.8 Recap of recommendations

This section summarizes the high-level practice recommendations to use for SNA network security.

- ▶ Policy recommendations:
 - Do not allow mixed security environments (for example, production and test) to be connected via SNA.
 - Be sure to separate system programmer and system operation duties between two or more individuals.
 - Allow only as much access to any resources as is required to perform tasks.
 - Ensure that all resource definitions are purged when equipment is removed from an SNA environment.
- ▶ Network recommendations:
 - Secure SNA links using IP with some type of encryption and authentication (IPSec VPNs or SSL/TLS):
 - For Enterprise Extender, use IPSec to protect the data.
 - For TN3270, use AT-TLS or SSL/TLS to protect the data.
 - Review the TN3270 settings that can control access to VTAM applications
 - Evaluate the use of multiple TN3270 servers to separate those applications that hold sensitive data from those that do not.
 - Do not allow nonqualified searches to enter the SNA environment from other NetIDs.
 - Set BNDYN=NONE and use ADJCLUST tables to secure network searches.
- ▶ Platform recommendations:
 - Set up protections for VTAM data sets by using RACF or an equivalent SAF interface product.
 - Code adjacent CP table definitions for any non-native nodes to restrict access.
 - Code adjacent SSCP tables to control searching.
 - Modify DYNADJCP, DYNASSCP, and SSCPDYN start options to NO.
 - Code ALIASRCH to NO and use the AUTHNET option in the ADJCP definitions for cross-network connections.
- ▶ Application:
 - Evaluate the use of session-level encryption and authentication techniques to secure sessions that carry critical data.



Shared Memory Communications over RDMA

In this chapter we provide information about the security characteristics of the IBM z/OS Shared Memory Communications over RDMA (SMC-R) function of z/OS 2.1. It is based on the assumption that readers have a basic understanding of TCP/IP protocols and the related z/OS implementation of those protocols.

SMC-R is an open protocol that is defined in the informational RFC titled *Shared Memory Communications over RDMA*, which is available on the following web page:

<https://datatracker.ietf.org/doc/draft-fox-tcpm-shared-memory-rdma/>

In this chapter, we focus exclusively on the IBM z/OS implementation of the SMC-R protocol. Our primary purpose is to describe how SMC-R communications are secured and the way that existing z/OS Communications Server network security features apply to SMC-R.

The chapter includes the following sections:

- ▶ 5.1, “Overview” on page 156
- ▶ 5.2, “Security characteristics of SMC-R connections” on page 159
- ▶ 5.3, “z/OS network security features and SMC-R” on page 162

5.1 Overview

This section provides an overview of the Shared Memory Communications over RDMA support that was introduced in z/OS Communications Server 2.1.

Remote Direct Memory Access (RDMA) is a communications technology that enables a host to make a subset of its memory directly available to a remote host. By doing so, data can be transferred between hosts efficiently and without any help from the CPU on the source or target host. Historically, RDMA has been confined to high-performance computing environments where the cost of maintaining RDMA-capable network fabrics such as InfiniBand was justified given the emphasis of performance over cost. However, RDMA is now available on standard Ethernet-based networks by using the industry (InfiniBand Trade Association) standard referred to as RDMA over Converged Ethernet (RoCE). With RoCE, the cost of adopting RDMA is lower because it can flow over the Ethernet fabrics that are already in place to carry IP network communications. Both standard TCP/IP and RDMA traffic can flow over the same physical LAN fabric at the same time, but RDMA network interface cards (RNICs, also referred to as RoCE host channel adapters (HCAs), are required to do so. On System z, the 10Gb RoCE Express adapter serves as the RNIC.

z/OS Communications Server 2.1 introduces a new capability that combines the performance benefits of RDMA with the widely-used TCP/IP sockets programming interface. This function, called Shared Memory Communication - RDMA (SMC-R) allows your TCP sockets applications to benefit from direct, high-speed, low-latency, memory-to-memory (peer-to-peer) communications over RDMA transparently - no changes are required in your application programs.

SMC-R provides an enterprise class of services for RDMA that are designed for enterprise class data center networks. Communicating peers (the z/OS TCP/IP stacks) dynamically learn about the shared memory capability by using traditional TCP/IP connection establishment flows. With this awareness, the TCP/IP stacks can switch from TCP network flows to more efficient direct memory access flows that use RDMA. The application programs are unaware of the switch to shared memory communications.

The remainder of this section will describe relevant characteristics of SMC-R communications in only enough detail to provide a basis for the later security discussion. For a more complete description of the z/OS SMC-R implementation, see Chapter 10 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

5.1.1 SMC-R: A hybrid protocol

Shared Memory Communications over RDMA is a hybrid protocol that uses RDMA technology within an existing IP network topology. SMC-R connections are established and operate transparently to applications within the context of their TCP socket connections. IP communications occur over OSA adapters (as they have in the past) and the associated SMC-R connections are established over the RNICs. As such, the RNICs must be attached to the same network infrastructure as the OSAs.

SMC-R's reliance on existing IP network topology and TCP connection setup preserves critical TCP/IP operational and network management features, including compatibility with transport layer load balancers (for example, Sysplex Distributor), preserving the IP security model (IP filters, VLANs, TLS/SSL, and so forth), and minimal (or zero) topology changes to accommodate the use of RDMA. This reliance on IP topology is a foundational element of the SMC-R security landscape.

5.1.2 SMC-R eligibility

For two nodes to be eligible to communicate with SMC-R, several criteria must be met:

- ▶ Both must be enabled for SMC-R.
- ▶ Both must have direct access to the same physical LAN fabric.
- ▶ Both must have direct access to the same IP subnet and VLAN (if VLANs are defined).
- ▶ The organization does not require IPsec protection of network and transport layer headers on the LAN segment (however, TLS/SSL can be used to protect application data across SMC-R enabled connections). We describe the role of IPsec in more detail later.

The *direct-access* requirements are based on the fact that the underlying RDMA connections are non-routable. This means that SMC-R connections are not routable as well. The direct-access requirements ensure that a direct communication path exists at layer 2 between the SMC-R capable nodes, with no intervening IP router. The additional VLAN requirement further confines the traffic within the physical LAN fabric in cases where VLANs are in use.

The topology requirements are illustrated in Figure 5-1. As you can see, the SMC-R enabled TCP connections between HOST A and HOST B are allowed because both hosts have OSA and RoCE adapters connected to the same physical LAN fabric, VLAN and IP subnet. On the other hand, even though HOST C is attached to the same physical LAN fabric, it cannot establish any IP connections to HOST A or HOST B without an intervening IP router because it is connected to a different VLAN and IP subnet.

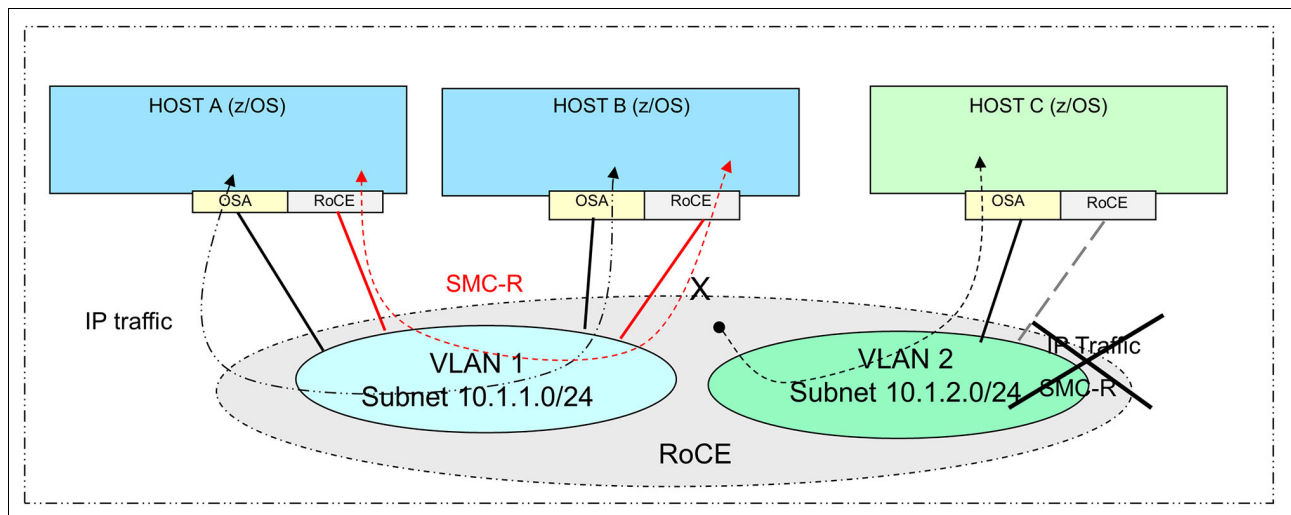


Figure 5-1 Network topology and SMC-R eligibility

Because SMC-R connection processing uses existing IP topology (TCP/IP connection setup) SMC-R connections transparently *inherit* the same VLAN and IP subnet connection eligibility attributes of the associated TCP connection. When VLANs are in use, SMC-R connections then become VLAN qualified.

Note: Because SMC-R's topology and eligibility requirements mimic those of IP, the level of trust that an organization has in its IP network infrastructure should mirror its trust, in that infrastructure for SMC-R purposes.

5.1.3 Enabling SMC-R and connection setup

SMC-R is enabled by specifying the SMCR parameter of the GLOBALCONFIG statement in the TCP/IP profile data set and including one or more Peripheral Component Interconnect Express (PCIe) function ID (PFID) values. Each PFID value represents an RNIC adapter that is configured by using the traditional hardware configuration definition (HCD) tools. TCP/IP activates the RNICs when the first SMC-R capable IP interface is started. IPAQENET or IPAQENET6 interfaces with the OSD channel path ID type can be configured for SMC-R capability.

Any TCP connections that traverse SMC-R capable IP interfaces are eligible for SMC-R communications. The decision about whether an eligible connection uses SMC-R communications is reached during traditional TCP connection establishment. The sequence of flows that determine whether or not to use SMC-R on a given TCP connection is called Rendezvous processing, which is illustrated in Figure 5-2.

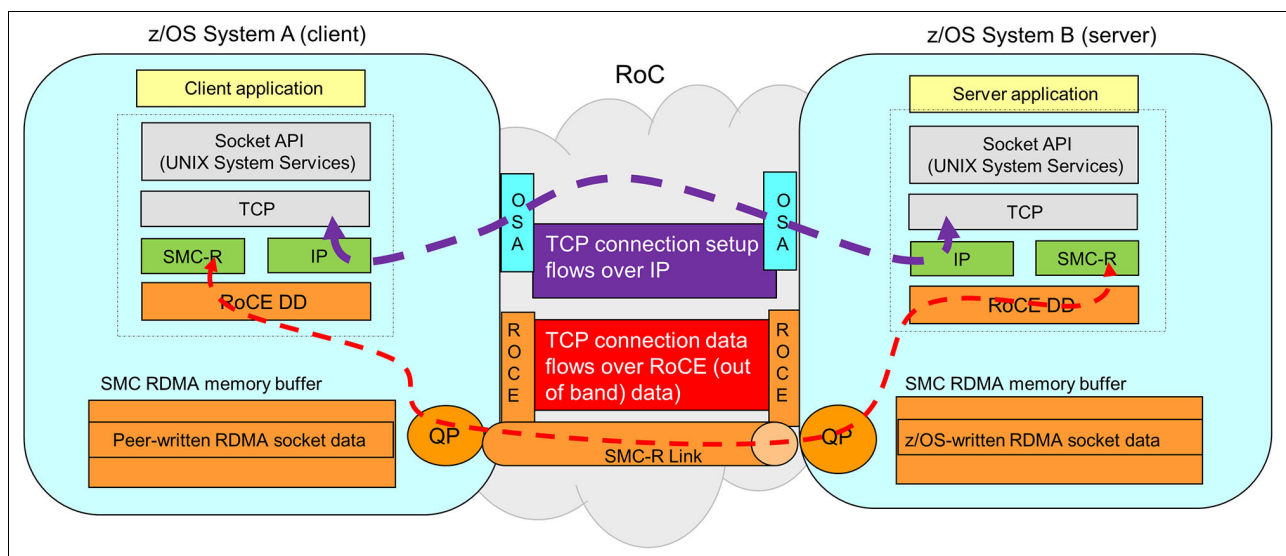


Figure 5-2 Rendezvous processing

The Rendezvous exchange of information occurs in three stages:

1. TCP connection establishment flows

TCP connections are still established using the standard three-way handshake mechanism. When SMC-R communications are enabled, the client adds TCP options settings in the SYN request to indicate that it supports SMC-R protocols. The server, when SMC-R communications are enabled, likewise responds with TCP options settings for SMC-R in the SYN-ACK response. No additional exchange of information is required in this stage of the rendezvous processing.

2. In-band SMC-R Connection Layer Control (CLC) messages

After the TCP three-way handshake succeeds, the client and server negotiate the use of SMC-R for this TCP connection by using SMC-R CLC messages that flow as in-band data over the TCP connection. Conceptually, these flows are similar to the TLS/SSL handshake processing that occurs after the TCP connection is established, but they occur before any data is allowed to flow over the TCP connection (including the TLS/SSL handshake).

The CLC messages exchange the following information:

- Layer 2 addressing information (MACs and GIDs)
- RoCE credentials, consisting of
 - Remote memory buffer access information
 - Queue Pair and related information

3. SMC-R Link Layer Control (LLC) messages

Using the RoCE credentials exchanged in phase 2, an RDMA connection called an SMC-R Link is established between the two peers across Reliable Connected Queue Pairs (RC QPs). SMC-R LLC messages are then exchanged across the SMC-R Link to confirm that the RoCE information is correct and that the RC QPs that comprise the SMC-R link have connectivity. This stage is skipped if an existing RoCE connection is used for this TCP connection.

Because the z/OS TCP/IP stack does not allow the client and server applications to exchange application data before or during rendezvous processing, the TCP connection can revert to IP protocols if there is a failure during the setup of the SMC-R communications. However, after the RoCE connection is confirmed by using the LLC messages, the TCP connection is committed to using SMC-R protocols and cannot fall back to using IP protocols if SMC-R communications encounter an error.

Even though application data is sent out of band from the TCP connection with SMC-R communications, the TCP connection remains active to preserve the connection state for monitoring and management functions, load balancers, and so on, and to support various stack functions, including connection termination processing.

5.2 Security characteristics of SMC-R connections

This section examines SMC-R connections from a security perspective and compares them to IP packets and TCP connections within a single LAN segment, because that is the scope of an SMC-R connection.

5.2.1 Protecting application data

First, let us consider the protection of application-level data as it traverses the LAN. Many organizations need to protect application-level data (transactions, database query results, transferred files, and such) as it crosses any network between two nodes. When those applications use TCP-based protocols (FTP, HTTP, CICS transactions, and so on), Transport Layer Security (TLS) protocols can be used to provide endpoint authentication, data authentication, data integrity and data privacy (encryption) protections for the application data. Because this protection is done above the transport layer, it can be used for SMC-R enabled connections just as effectively as it can for TCP connections over a regular IP network.

5.2.2 Protecting network protocol headers

We have explained that application data can be protected over SMC-R enabled links by using the TLS, just as it can be over regular TCP/IP. However, if you're wondering about protecting the TCP and IP headers that contain the application data, this section gives you a closer look at the types of attacks that can be directed at the lower-level networking headers and then discusses protection scenarios as they apply to regular IP networks and those carrying RDMA traffic.

TCP/IP, RDMA, and the underlying Ethernet-based link layer protocols must all carry enough information in each transmission unit to ensure the data is properly handled as it traverses the network and after it arrives at its destination. Specifically, such a protocol provides the information that you need for the following purposes:

- ▶ Identify where it came from (source identity or location).
- ▶ Identify who its intended target is (destination identity or location).
- ▶ Identify where the data is supposed to go within that target (upper-layer tagging).
- ▶ Convey other types of control information to ensure proper handling of transmitted data.

Ethernet provides the source and destination identity/location information in the form of MAC addresses. Along with the MAC addresses, Ethernet uses an associated upper-layer addressing value (an IP address in the case of IP or a Global ID (GID) in the case of RDMA) to assist in the routing of a given frame through the LAN and to the proper device driver after it arrives at its destination. Other control information is included in each Ethernet frame header.

IP provides the source and destination identity/location information through IP addresses and the control information through various flags or headers. In the IP cases relevant to our discussion, TCP provides the upper-layer tagging in the form of TCP port information and also provides its own control information. Other control information is included in each IP packet header.

When an IP packet traverses an Ethernet-based LAN, it is susceptible to different kinds of attack techniques, including the following types:

- ▶ MAC and IP address spoofing, where a LAN-attached device generates or modifies an Ethernet frame containing an IP packet using another device's MAC address or IP address as the source address information to impersonate that device. This technique is usually used in combination with packet injection or man-in-the-middle (MITM) attacks as described here.
- ▶ Packet injection, where a LAN-attached device assembles a new Ethernet frame containing an assembled IP packet using spoofed source MAC and IP addresses and sends it to a destination with the intent that it will be consumed as though it originated at the spoofed address. Note that this requires pre-knowledge of other pieces of control information as well (sequence numbers, and so on).
- ▶ Man-in-the-middle (MITM) attacks, where a LAN-attached device captures an Ethernet frame carrying an IP packet that was sent by a legitimate IP node on the LAN, modifies some of the contents, and then forwards it on to the destination node with the intent that the destination node will consume the modified data as though it came from the original sender.
- ▶ Denial of Service (DoS) attacks, where a LAN-attached device takes some sort of action to disrupt a connection that already exists between two legitimate IP nodes on that LAN segment or else to consume the resources of one of the nodes such that other nodes cannot use its services. An example of a DoS attack on a TCP connection is that of injecting a packet into an existing TCP connection with incorrect sequence information to force the receiver of the injected packet to close the TCP connection in response to an error condition.

There are a few of ways to mitigate the risks involved in these types of IP-based attacks:

- ▶ For LANs that are easily accessed (for example, wireless LANs, those with easily accessible wired ports, those accessible from less secure portions of the network through Layer 3 routing, and so on), IPSec can be used to protect the IP packets, including the IP headers.
- ▶ For more secure LANs, such as those connecting an enterprise's high-performance (multi-tiered) clusters, closely controlled physical access along with firewall control for external access might be enough to ensure that only authorized IP nodes have access to the LAN.
- ▶ Regardless of LAN accessibility, z/OS Communications Server's TCP Traffic Regulation support (part of integrated Intrusion Detection Services) can be used to limit the number of TCP connections that any single client can establish against a given TCP port. This can be useful in mitigating DoS risks.

SMC-R connections are expected to be established on the second type of LAN, where the physical access is controlled tightly enough to obviate the need for cryptographic protection of network and transport layer headers.

Because SMC-R is based on RoCE, which in turn is based on RDMA, it relies on the RoCE and RDMA protocols that use the Ethernet MAC and the RDMA GID for their source and destination identity/location. The RC QPs that comprise the SMC Link provide the *session* level information (analogous to a TCP connection). The RC QPs are identified by source and destination queue pairs. An SMC Link is further bound to a specific area of memory in each partner called a remote memory buffer which is identified by a special token and key, called RToken and RKey, respectively. All of this information is carried in the various session establishment flows over IP and RoCE as described and also over the RNICs during the underlying RDMA connection establishment flows.

Although the SMC-R sessions can be attacked by using techniques similar to those described for IP, these connections are no more or less vulnerable to such attacks than regular TCP/IP connections. However, because there is no RDMA analog to IPSec, SMC-R connections should be enabled only on LAN segments that are physically secure.

If an organization does not trust in the physical security of a LAN segment enough to flow data without IPSec protection, then that segment should not be enabled for SMC-R.

Although the preceding point should be used as a general guideline regarding the appropriateness of SMC-R traffic over any given LAN segment, bear in mind that the use of IPSec to protect a TCP session actually disables the use of SMC-R on z/OS, as described in subsequent sections.

5.2.3 Firewalls and Deep Packet Inspection (DPI) devices

One other important consideration in the use of SMC-R is its effect on firewalls and deep packet inspection (DPI) devices, such as *bump-in-the-wire* intrusion prevention devices. These devices process Ethernet frames, IP packets, and other well-known protocols built upon IP, and they are usually deployed at network zone boundaries along with a Layer 3 (IP) router.

Because SMC-R traffic is not routable (due to its RDMA roots), it is not expected to reach traditional Layer 3 firewalls. Likewise, it would be unusual for SMC-R traffic to reach a non-traditional firewall or a DPI device.

However, such devices might be incompatible with RDMA traffic, so it is important to read product documentation for the specific device to learn how the device behaves if it encounters SMC-R (RDMA) traffic.

5.3 z/OS network security features and SMC-R

This section provides an overview of the key security features of z/OS Communications Server and how each relates to SMC-R. Note that this section discusses only the subset of security features that are specifically relevant to the SMC-R functions in the TCP/IP stack. It is not a comprehensive survey of all the Communications Server security functions.

5.3.1 Interface-based SMC-R enablement

The TCPIP profile data set's INTERFACE statement allows you to specify whether or not a given IPAQENET or IPAQENET6 interface (with the OSD channel path ID type) is enabled for SMC-R by specifying the SMC or NOSMC parameter. If not specified, the default is to enable SMC-R for these types of interfaces.

5.3.2 Port-based SMC-R exclusion

You can use the TCPIP profile data set's PORT and PORTRANGE statements to exclude a port or set of ports from SMC-R eligibility by specifying the NOSMC parameter. This can be a useful feature for disabling SMC-R for a specific z/OS TCP server application.

5.3.3 SAF-based network access controls

z/OS Communications Server provides two types of SAF resources that allow system administrators to control access of z/OS user IDs to network resources. One controls access to different network zones (hosts, subnets, and networks), and the other controls access to local TCP ports.

EZB.NETACCESS.sysname.tcpname.zonename resources control which z/OS user IDs are allowed to access different network zones. User ID with permission to a NETACCESS resource are allowed to send and receive data to and from the network zone represented by the zone name. Network access control provides an additional layer of security on top of any authentication and authorization mechanisms that are used in the network or at the peer system by preventing unauthorized users from communicating with the peer network resource.

EZB.PORTACCESS.sysname.tcpname.portname resources determine which z/OS user IDs are allowed to bind to specific TCP ports. User IDs with permission to a PORTACCESS resource are allowed to bind to the TCP port represented by the resource.

Because both NETACCESS and PORTACCESS controls are enforced at the socket layer, they apply fully to SMC-R enabled connections.

For more information about NETACCESS and PORTACCESS resources, see Chapter 3 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

5.3.4 IP filter rules

z/OS Communications Server's IP security function includes IP packet filtering that controls the flow of IP packets into and out of the TCP/IP stack. An administrator can define IP security policy to permit or deny any type of IP traffic from entering/exiting the TCP/IP stack. Both local and routed traffic are supported and filter rules can be defined using a wide variety of criteria.

Basic permit/deny filters for local traffic are honored for TCP connections that use SMC-R. A permit filter allows TCP connections to be established while a deny filter prevents them. This is true regardless of the use of SMC-R. After a TCP connection is successfully established, any change in IP filtering configuration that installs a deny filter for the connection will immediately cause the connection to be dropped for both IP and SMC-R traffic.

Given that RoCE (and hence, SMC-R) traffic is not routable, IP filters that apply to routed traffic do not apply to SMC-R traffic.

For more information about IP filtering, see Chapter 19 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

5.3.5 IPSec

In addition to IP filtering, Communications Server's IP security function provides a complete IPSec implementation. IPSec is a cryptography-based set of technologies that allow participating peers to establish a wide variety of VPN tunnels. As described earlier, IPSec protects entire IP packets (not just the application data within those packets) and is completely transparent to application programs. z/OS applies IPSec protection under the direction of IP filter rules that specify an action of *protect with IPSec*.

Because IPSec protocols are bound tightly to IP packet formats, they are simply not compatible with SMC-R. As such, when any TCP connection is protected by an IPSec filter rule, z/OS makes that connection ineligible for SMC-R. Furthermore, if an SMC-R enabled TCP connection is already established when such an IPSec filter rule is installed, then that connection will be terminated.

For more information about IPSec, see Chapter 19 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

5.3.6 SSL/TLS, including Application Transparent TLS (AT-TLS)

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL) are cryptography-based technologies that are applied just above the transport layer, typically under direction of the application program that is generating the traffic to be protected. z/OS offers two RFC-compliant TLS/SSL implementations:

- ▶ System SSL, a component of the z/OS Cryptographic Services element. System SSL can either be invoked directly through its own set of APIs or it can be invoked transparently on the application's behalf based on Communications Server Application Transparent TLS (AT-TLS) policies. As its name implies, the latter approach protects application traffic without requiring application code changes.
- ▶ Java Secure Socket Extension (JSSE), a 100% Java implementation for applications based on WebSphere and other pure Java applications. JSSE2 (the most current version of JSSE) is provided as part of the z/OS Java Runtime Environment.

Because TLS (and SSL) protection is applied to application data just above the transport layer (before any TCP segmentation, which is true even for AT-TLS), it is completely

compatible with SMC-R. The TCP/IP stack, including the SMC-R function sees it all simply as application data.

For more information about System SSL, see *z/OS Cryptographic Services System Secure Sockets Layer Programming Version 2 Release 1*, SC41-7495. For more information about AT-TLS, see Chapter 22 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

To download a copy of the documentation for the IBMJSSE2 provider, see the Security documentation on IBM developerWorks®. In addition to this cross platform information, z/OS specific information for JSSE2 can be found in the *z/OS JSSE Reference Guide*.

5.3.7 SSH

The Secure Shell (SSH) protocol is widely-used in to cryptographically protect traffic between UNIX environments (including z/OS UNIX). SSH operates at above the transport layer, so it applies to SMC-R enabled TCP connections in the same manner as TLS/SSL.

The Ported Tools for z/OS offering includes a version of OpenSSH for z/OS. For more information about this OpenSSH implementation, see the *IBM Ported Tools for z/OS: OpenSSH User's Guide Version 1 Release 2*, SA23-2246.

5.3.8 Application layer security protocols and features

Many security protocols exist at the application layer and some applications have their own security features. For example, Web Services Security (WS-Security), Domain Name System Security (DNS-SEC), security features within SNMP and so forth. As with TLS/SSL and SSH, the fact that these are applied above the transport layer means that it all just looks like application data to the TCP/IP stack and to SMC-R. Because of this, application layer security functions will continue to work over SMC-R enabled connection.

5.3.9 Integrated Intrusion Detection Services (IDS)

z/OS Communications Server Intrusion Detection Services (IDS) provide reporting and, in some cases, defensive protections, for a wide variety of intrusion-related events that fall into the following categories:

- ▶ Scan detection and reporting
- ▶ Traffic regulation
- ▶ Attack detection, reporting and prevention

Scan detection and traffic regulation are both enforced during TCP connection setup and therefore complete before SMC-R enablement is ever considered for a given connection.

Attack detection includes a wide range of checks. Because most of the TCP-related checks apply to inbound TCP packets, they are not relevant to the direct memory transfers employed by SMC-R communications. However, two attack types do apply to SMC-R enabled connections. Let's examine each of these in more detail.

TCP queue size events

You can configure IDS policy to detect when a TCP connection's send or receive queues become constrained due to the amount or age of the data on the queue. When a queue becomes constrained, the IDS rule can either reset the TCP connection or continue to

monitor the condition until the queue is no longer constrained. Send or receive queues for SMC-R enabled TCP connections are considered to be constrained in the following circumstances:

- ▶ The send queue is considered to be constrained when available outbound data cannot be sent because the peer's RMB element has no available space for a prolonged period of time (30 seconds). This condition typically identifies a problem with the peer application not consuming the available data in a timely manner.
- ▶ The receive queue is considered to be constrained when inbound data that has been stored into the local memory buffer is not received by the application for more than 30 seconds.
- ▶ When either queue becomes constrained, the TCP connections are monitored or stopped according to the relevant IDS policy rule.

Global TCP stall events

You can configure IDS policy to detect attacks that are designed to consume system resources by creating many TCP connections and causing them to stall, making them unable to send data. A global stall condition is in effect when at least 50% of the active TCP connections are stalled and at least 1000 TCP connections are active. A global TCP stall IDS rule can either reset stalled connections or continue to monitor the condition. TCP connections that traverse SMC-R links are considered during global TCP stall detection. Such a connection is considered to be stalled when the TCB is write-blocked.

For more information about IDS functions, see Chapter 18 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

5.3.10 Multilevel Security (MLS)

Multilevel security is an enhanced security environment in which the z/OS security server and trusted resource managers enforce mandatory access control policies in addition to the usual discretionary access control policies. To participate in an MLS environment, the user IDs associated with tasks trying to access z/OS resources and those resource profiles in the SERVAUTH class need to have security labels defined.

MLS is supported for SMC-R enabled TCP connections with one exception. If a TCP connection requires MLS packet tagging (that is, if the source and destination zones are both SYSMULTI), then SMC-R will be disabled for that connection. If an SMC-R enabled TCP connection is already established and a dynamic configuration update introduces a requirement for MLS packet tagging on that connection, it will be dropped.

For more information about MLS, see Chapter 4 in the *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650.

Related publications

The publications listed in this section are useful for more information about topics in this book.

IBM Redbooks

Some publications referenced in this list might be available in softcopy only:

- ▶ *Security on the IBM Mainframe: Volume 1 A Holistic Approach to Reduce Risk and Improve Security*, SG24-7803
- ▶ *Reduce Risk and Improve Security on IBM Mainframes: Volume 3 Mainframe Subsystem and Application Security*, SG24-8196
- ▶ *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 1 Base Functions, Connectivity, and Routing*, SG24-7996
- ▶ *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 2 Standard Applications*, SG24-7997
- ▶ *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 3 High Availability, Scalability, and Performance*, SG24-7998
- ▶ *IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking*, SG24-7999
- ▶ *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, drafts, and additional materials on the IBM Redbooks website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS Communications Server IP Configuration Guide Version 2 Release 1*, SC27-3650
- ▶ *z/OS Cryptographic Services System Secure Sockets Layer Programming Version 2 Release 1*, SC41-7495
- ▶ *IBM Ported Tools for z/OS: OpenSSH User's Guide Version 1 Release 2*, SA23-2246

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



SG24-8195-00

ISBN 0738440949

Printed in U.S.A.

Get connected

