

Reduce Risk and Improve Security on IBM Mainframes: Volume 1 Architecture and Platform Security

Axel Buecker

Boudhayan Chakrabarty

Lennie Dymoke-Bradshaw

Cesar Goldkorn

Brian Hugenbruch

Madhukar Reddy Nali

Vinodkumar Ramalingam

Botrous Thalouth

Jan Thielmann



 **Security**

z Systems



International Technical Support Organization

**Reduce Risk and Improve Security on IBM Mainframes:
Volume 1 Architecture and Platform Security**

December 2014

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

First Edition (December 2014)

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

© Copyright International Business Machines Corporation 2014. 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	xi
Trademarks	xii
IBM Redbooks promotions	xiii
Preface	xv
Authors	xv
Now you can become a published author, too!	xvii
Comments welcome	xvii
Stay connected to IBM Redbooks	xvii
Part 1. Direction and architecture	1
Chapter 1. Introduction: Why these books are being written	3
1.1 Scope of these books	5
1.1.1 In scope	5
1.1.2 Out of scope	5
1.2 Enterprise security	5
1.3 IBM Security Framework and IBM Security Blueprint	6
1.4 Horizontal functions	7
1.5 Structure of preferred practices	8
1.6 Risk-based approach	9
1.7 Structure of this IBM Redbooks series	9
1.8 Conclusion	9
Chapter 2. Foundation of a holistic security architecture	11
2.1 IBM Security Framework	12
2.1.1 Advanced Security and Threat Research	13
2.1.2 People	14
2.1.3 Data	16
2.1.4 Applications	17
2.1.5 Infrastructure	18
2.1.6 Security Intelligence and Analytics	20
2.1.7 Security Maturity Model	21
2.2 IBM Security Blueprint	22
2.2.1 Foundational Security Management	24
2.2.2 Security Services and Infrastructure	25
2.2.3 Architectural principles	26
2.3 Conclusion	28
Chapter 3. Mainframe security architecture in the enterprise	29
3.1 History of the mainframe	30
3.1.1 Late 1960s	30
3.1.2 Early 1970s	31
3.1.3 Late 1970s	32
3.1.4 Early 1980s	32
3.1.5 Late 1980s	33
3.1.6 Early 1990s	33
3.1.7 Late 1990s	34

3.1.8	Early 2000s	34
3.1.9	Late 2000s	35
3.1.10	The mainframe today: 2010 onwards	36
3.1.11	Strengths of the mainframe	37
3.2	Workloads on the mainframe	41
3.2.1	Batch processing	41
3.2.2	Online transaction processing	42
3.2.3	Modern day classification	43
3.3	Virtualization	47
3.3.1	Virtualization on System z	48
3.3.2	Goals and benefits of virtualization	50
3.4	Overview of mainframe security architecture	51
3.4.1	Hardware and microcode design	52
3.4.2	Software design	57
3.5	Role of the mainframe within an enterprise security architecture	66
3.5.1	Risk and Compliance Assessment	66
3.5.2	Command and Control Management	68
3.5.3	Security Policy Management	70
3.5.4	Identity, Access, and Entitlement Management	71
3.5.5	Data and Information Protection Management	73
3.5.6	Software, System, and Service Assurance	74
3.5.7	Threat and Vulnerability Management	75
3.6	Statement of Integrity and certification levels	76
3.6.1	Evaluation Assurance Levels	78
3.6.2	Certification levels for the mainframe	79
3.6.3	Statement of integrity	79
3.7	Conclusion	80
Chapter 4. Hardware components		83
4.1	System components	84
4.1.1	The complete picture	84
4.1.2	Microcode	85
4.1.3	Processor	86
4.1.4	Storage	89
4.1.5	Coupling Facility	90
4.2	Devices	90
4.2.1	Channel connectivity	90
4.2.2	Adapters	91
4.2.3	Controllers	93
4.2.4	External storage	95
4.2.5	Terminals	97
4.2.6	Printers	97
4.3	Conclusion	98
Chapter 5. Software components		99
5.1	Operating systems	100
5.1.1	z/OS	100
5.1.2	z/VM	102
5.1.3	Linux on z	104
5.1.4	z/VSE	104
5.1.5	z/TPF	105
5.2	z/OS functions	105
5.2.1	Initial program load	105

5.2.2	Master scheduler	106
5.2.3	The link pack area	107
5.2.4	System parameters	107
5.2.5	Linklist	108
5.2.6	Job entry subsystem	108
5.2.7	Job Control Language	109
5.2.8	Procedures and tasks	109
5.2.9	Time Sharing Option	110
5.2.10	ISPF	110
5.2.11	Data sets	111
5.2.12	Catalog	113
5.2.13	Input/output processing	113
5.2.14	Storage management	114
5.2.15	SDSF	115
5.2.16	System Management Facility	115
5.2.17	Resource Measurement Facility	116
5.2.18	System logger	117
5.2.19	UNIX System Services	117
5.2.20	Software delivery	118
5.2.21	SMP/E	119
5.2.22	Cross-system coupling facility	120
5.2.23	z/OS Security Server	120
5.2.24	Language Environment	121
5.3	Network functions	122
5.4	Application software	123
5.5	ISV/OEM software	123
5.6	Conclusion	124
Chapter 6. Security solutions for IBM System z		125
6.1	IBM InfoSphere Guardium for z/OS	126
6.2	IBM Security zSecure Suite	128
6.2.1	Administration management with Security zSecure Suite	129
6.2.2	Security audit and compliance with Security zSecure	133
6.3	IBM Security QRadar	136
6.3.1	IBM Security QRadar Log Manager	138
6.3.2	IBM Security QRadar SIEM	138
6.3.3	IBM Security QRadar Risk Manager	139
6.3.4	IBM Security QRadar QFlow for network and application activity monitoring	140
6.3.5	IBM Security QRadar VFlow for virtual activity monitoring	140
6.3.6	IBM Security QRadar Vulnerability Manager	141
6.3.7	IBM Security QRadar Network Anomaly Detection	141
6.3.8	Integration of QRadar SIEM with z/OS	142
6.4	IBM Security Key Lifecycle Manager	142
6.4.1	Security Key Lifecycle Manager components and resources	143
6.4.2	Encryption-enabled 3592 and LTO tape drives	145
6.4.3	Enterprise storage: IBM System Storage DS8000 (2107, 242x)	145
6.4.4	Managing encryption	145
6.5	IBM Enterprise Key Management Foundation	147
6.5.1	The solution architecture	147
6.5.2	Centralized key management using the IBM Enterprise Key Management Foundation	148
6.5.3	Certificate management	151
6.6	Encryption Facility for z/OS	152

6.6.1 Encryption Services Feature	152
6.6.2 IBM Encryption Facility for z/OS Client	152
6.6.3 DFSMSdss Encryption	152
6.6.4 Encryption Services batch programs CSDFILEN and CSDFILDE	153
6.6.5 Encryption Facility for OpenPGP	153
6.7 IBM Security AppScan	153
6.7.1 Main editions of IBM Security AppScan	154
6.8 IBM Security Access Manager	157
6.8.1 Security Access Manager architecture	157
6.8.2 IBM Security Access Manager for Enterprise Single Sign-On	159
6.9 IBM Security Identity Manager	160
6.9.1 IBM Security Identity Manager entities	160
6.9.2 Security Identity Manager key functions and logical architecture	161
6.9.3 The Security Identity Manager logical architecture	162
6.9.4 The Security Identity Manager RACF adapter	163
6.10 Federated Identity Management	165
6.11 Conclusion	165

Part 2. Guiding principles for IBM System z security 167

Chapter 7. Organizing for security	169
7.1 The mainframe infrastructure does not exist alone	170
7.1.1 Defense in depth: The castle approach	170
7.1.2 The analogy	171
7.2 Security policy definition and implementation	171
7.3 Security design	171
7.3.1 Threats change over time	172
7.4 Continuous improvement	172
7.5 System z audits	173
7.6 Monitoring, alerting, and forensics	173
7.7 Access creep	174
7.8 The purpose of security	174
7.8.1 Maintaining the integrity of the operating system	174
7.8.2 Maintaining the security of application data	175
7.8.3 Maintaining the separation of applications, virtual machines, and networks.	175
7.9 Types of security controls	176
7.9.1 Preventive controls	176
7.9.2 Monitoring controls	176
7.9.3 Detective controls	177
7.9.4 Forensic controls	177
7.9.5 All controls	177
7.10 Standards-based approach	177
7.10.1 Need for a standards-based approach	178
7.10.2 Challenges with a standards-based approach	178
7.10.3 “Standards-plus” approach	178
7.11 Security engineering	179
7.11.1 Business and application development	180
7.11.2 Audit and external compliance	180
7.11.3 Security management and operations	180
7.11.4 Technology and infrastructure capabilities	180
7.11.5 Other responsibilities	181
7.11.6 Security engineering and data ownership	181
7.12 Conflicts of interest	181

7.12.1	Conflict of availability versus security	182
7.12.2	Conflict of cost versus security	182
7.13	Proving that security controls work	183
7.13.1	Testing individual controls	183
7.13.2	Regular security control scans	183
7.13.3	Penetration testing	184
7.14	IBM services to help you	184
7.14.1	Preventive maintenance	184
7.14.2	The IBM System z Security Portal	184
7.14.3	IBM Emergency Response Services	184
7.15	Decision time	185
7.16	Conclusion	185
Chapter 8. IBM System z hardware		187
8.1	Physical access controls	188
8.2	Hardware Management Console and Support Element	189
8.2.1	Considerations for multiple Hardware Management Consoles	189
8.2.2	HMC guiding principles	190
8.3	Input and output configuration	192
8.4	Terminals and printers	194
8.5	Storage devices	195
8.6	Tape libraries and removable media	195
8.7	Network adapters	196
8.8	Other peripheral devices	197
8.9	Logging and auditing for hardware	197
8.10	Conclusion	198
Chapter 9. IBM z/OS security		199
9.1	Introduction	200
9.1.1	Finding security gaps	200
9.1.2	The magic ingredient	201
9.1.3	Third-party software	202
9.2	z/OS settings	202
9.2.1	Consoles	202
9.2.2	System Management Facility	204
9.2.3	MVS commands	204
9.2.4	Program Properties Table	205
9.2.5	Subsystem definitions	208
9.2.6	PROG00	209
9.3	z/OS system routines	212
9.3.1	Supervisor calls	212
9.3.2	Program Call routines	214
9.3.3	System exits	216
9.3.4	I/O appendages	217
9.4	z/OS component protection	217
9.4.1	UACC	217
9.4.2	RACF database	219
9.4.3	Master catalog	219
9.4.4	SYS1.PARMLIB	220
9.4.5	System data sets	220
9.4.6	System page data sets	224
9.4.7	System dump data sets	224
9.4.8	SYS1.STGINDEX	225

9.4.9	SYS1.LOGREC	225
9.4.10	SMF data sets	225
9.4.11	System linklist	226
9.4.12	System LPALIST	227
9.4.13	APF-authorized libraries	227
9.4.14	System log	228
9.5	The IBM Health Checker for z/OS	228
9.5.1	Managing your health checks	230
9.6	RACF settings	232
9.6.1	INITSTATS	233
9.6.2	SAUDIT	233
9.6.3	CMDVIOL	233
9.6.4	OPERAUDIT	233
9.6.5	AUDIT classes	233
9.6.6	GLOBAL classes	234
9.6.7	Enhanced Generic Naming	234
9.6.8	BATCHALLRACF	235
9.6.9	PROTECTALL	235
9.6.10	Tape data set protection	235
9.6.11	Erase on scratch	236
9.6.12	Inactive user ID revocation	236
9.6.13	Password management	236
9.6.14	GENERICOWNER	238
9.6.15	Multi-level security options	238
9.7	JCL procedures	238
9.8	UNIX System Services	239
9.8.1	z/OS UNIX security	240
9.8.2	UID 0	240
9.8.3	BPX profiles	244
9.8.4	Sharing IDs	249
9.8.5	UNIX file system security	249
9.8.6	Access control lists	252
9.8.7	APF and other privileged bits in the FSP	252
9.8.8	Sanction lists	253
9.9	DFSMS	253
9.9.1	Placing data sets	254
9.9.2	Naming standards for data sets	254
9.9.3	Managing storage volumes	254
9.9.4	Backup and recovery processes for data sets	256
9.9.5	Removing unwanted data sets	256
9.9.6	Migrating data sets on a use frequency basis	256
9.9.7	Installing storage devices and migrating data sets to those devices	257
9.9.8	Disposing old storage devices	257
9.10	JES and SDSF	257
9.10.1	JES command security	257
9.10.2	NODES profiles	258
9.10.3	JESINPUT profiles	258
9.10.4	JESJOBS profiles	259
9.10.5	SURROGAT profiles	260
9.10.6	WRITER profiles	260
9.10.7	SDSF protection	260
9.11	TSO/E	261
9.11.1	Regulating access to the system	261

9.11.2	Limiting the use of TSO/E commands.	262
9.11.3	Limiting access to data sets	262
9.11.4	Summary of TSO/E resources that are protected by using RACF	263
9.11.5	IKJTSO00	263
9.12	Integrated Cryptographic Service Facility	264
9.12.1	Protecting ICSF keys and APIs.	265
9.12.2	Protecting key data sets	266
9.12.3	Keystore Policy	267
9.12.4	Trusted Key Entry Workstation	268
9.12.5	Access Control Points.	268
9.12.6	Disabling SAF checks for improved performance.	268
9.12.7	Special Secure Mode	269
9.13	Started procedures	269
9.13.1	Assigning RACF user IDs to started procedures	270
9.13.2	Authorizing access to resources	271
9.13.3	Setting up the STARTED class.	271
9.13.4	Using the started procedures table (ICHRIN03)	273
9.13.5	Started procedure considerations	274
9.14	Special procedures for privileged users	275
9.15	Multiple LPARs and shared DASD	276
9.16	Conclusion	278
Chapter 10.	IBM z/VM security	279
10.1	Overview of z/VM security.	280
10.1.1	The principles of guest isolation	281
10.1.2	The principles of hypervisor security.	281
10.2	Securing the virtual machine.	281
10.2.1	Privilege classes	282
10.2.2	Assigning a new privilege class to a VM.	282
10.2.3	LOGONBY	282
10.2.4	The COMMAND directory statement	283
10.3	Hypervisor security	283
10.3.1	Default settings	283
10.3.2	Command restructuring.	283
10.3.3	Observers and secondary users	284
10.3.4	Virtualized hardware cryptographic features.	284
10.4	Secure connectivity	285
10.4.1	TCP/IP	286
10.4.2	Guest LANs and Virtual Switches	286
10.4.3	HiperSockets.	287
10.5	External Security Managers	287
10.5.1	Reasons to use an ESM for z/VM.	288
10.5.2	Control of privileged VM commands.	288
10.5.3	Auditing of security operations	288
10.5.4	Security zones and multi-tenancy	289
10.6	Features and extensions to z/VM	289
10.6.1	DirMaint.	289
10.6.2	Single System Image clustering	291
10.6.3	Systems Management APIs	291
10.7	Preferred practices in z/VM security	292
10.7.1	Defining and deploying a security policy.	292
10.7.2	Examining audit trails periodically.	292
10.7.3	Applying recommended services	293

10.7.4 Data integrity	293
10.7.5 Installing and deploying an ESM.	294
10.8 Conclusion	295
Chapter 11. Linux on System z security	297
11.1 Security checklist	298
11.2 Securing the logical access to z/VM	298
11.2.1 z/VM user passwords	298
11.2.2 Choosing the z/VM privilege class	298
11.2.3 z/VM network connection	299
11.3 Securing the data	299
11.3.1 Securing your minidisks	299
11.3.2 Reducing the intrusion points with shared disks	300
11.3.3 Protecting the data with encrypted file systems	301
11.4 Securing the network	301
11.5 Access control.	301
11.6 Authentication	302
11.6.1 Improving security for SSH key pair	303
11.7 User management.	303
11.7.1 Centralized user repository	303
11.7.2 Securing connections to the user information repository	304
11.8 Audit	305
11.9 Separation of duties	305
11.10 Conclusion	306
Related publications	307
IBM Redbooks publications	307
Other publications	307
How to get Redbooks publications.	308
Help from IBM	308

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 about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

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

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	HiperSockets™	RMF™
AppScan®	IBM®	S-TAP®
BookManager®	IMS™	S/390®
Build Forge®	InfoSphere®	ServicePac®
CICS®	Insight™	System p®
ClearQuest®	Language Environment®	System Storage®
DataPower®	Lotus®	System z®
DB2®	MVS™	System z10®
developerWorks®	NetView®	System/390®
DirMaint™	OS/390®	Tivoli®
Domino®	Parallel Sysplex®	VTAM®
DRDA®	POWER®	WebSphere®
DS8000®	POWER7®	z/Architecture®
Enterprise Storage Server®	PR/SM™	z/OS®
eServer™	ProductPac®	z/VM®
FICON®	QRadar®	z/VSE®
FlashCopy®	RACF®	z10™
GDPS®	Rational®	z9®
Geographically Dispersed Parallel Sysplex™	Redbooks®	zEnterprise®
Global Technology Services®	Redbooks (logo)  ®	zSecure™
Guardium®	Resource Measurement Facility™	
	RETAIN®	

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

LTO, Ultrium, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

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

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

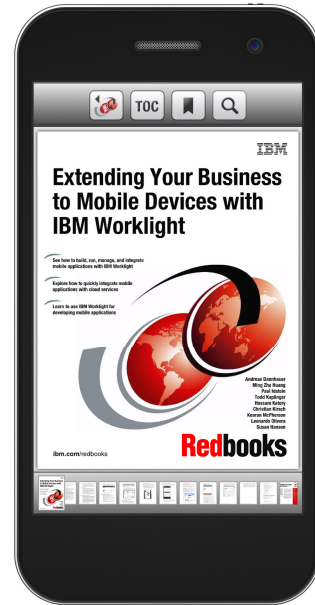
UNIX is a registered trademark of The Open Group in the United States and other countries.

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



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 System z® hardware and software. In an age of increasing security consciousness, IBM System z provides the capabilities to address the needs of today's business security challenges. This publication explores how System z hardware is designed to provide integrity, process isolation, and cryptographic capability to help address security requirements. This book highlights the features of IBM z/OS® and other operating systems, which offer various customizable security elements under the Security Server and Communication Server components. This book describes z/OS and other operating systems and additional software that leverage the building blocks of System z hardware to provide solutions to business security needs.

This publication's intended audience is technical architects, planners, and managers who are interested in exploring how the security design and features of System z, 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 the International Technical Support Organization, Austin Center.

Axel Buecker is a Certified Consulting Software IT Specialist at the ITSO, Austin Center. He writes extensively and teaches IBM classes worldwide on areas of software security architecture and network computing technologies. He holds a degree in Computer Science from the University of Bremen, Germany. He has 27 years of experience in various areas that are related to workstation and systems management, network computing, and e-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.

Boudhayan Chakrabarty is a Security Intelligence and Compliance Specialist at IBM India Software Labs, supporting worldwide customers with a specialization in security products. He is an IBM Certified Solution Advisor and holds a Bachelor of Technology degree in Computer Science from College of Engineering, Bhubaneswar (India). He has been involved in numerous white papers and IBM developerWorks® articles and has been conducting training, webcasts, and enablement courses on IT Service Management, and the IBM Security Intelligence and Compliance Portfolio. He is a regular speaker at external forums and his talks on Security Intelligence are published in the public domain.

Lennie Dymoke-Bradshaw is a long-term IBM MVS™ practitioner. He has worked with S/370 since 1975 when he started programming a 370/145 running DOS/VS in PL/I and assembly language. Over the years, he has maintained his interest in programming while performing various roles in applications programming, systems design, technical support, systems programming, software support, and whatever other technical roles customers have asked of him. During this time, he has worked with every level of IBM RACF® since RACF 1.3, and he maintains an interest in computer security, especially as applied to System z. Lennie joined IBM in 1996 and now works with System z cryptography as well. Lennie currently works in IBM System z software pre-sales in the UK, and publicizes the enormous benefits of System z from a security standpoint.

Cesar Goldkorn is an IBM IT Specialist that is based in Brazil, working with the Storage Management team in Strategic Outsourcing since 2005. Previously, Cesar worked at a Brazilian insurance company as system programmer, technical support manager, and enterprise architecture manager. He has over 40 years of experience with IBM large system hardware and software, where he started working with IBM 7044, IBM 1401, IBM 1130, and IBM /370-165 II running OS/MVT, VM/370, and OS/VS2 SVS. His areas of expertise include z/OS support, RACF, SMS and Storage Management, Performance Management and Capacity Planning, Enterprise Architecture, Total Quality Management, assembly language, PL/I, and APL.

Botrous Thalouth, CISSP, is a System z Virtualization Security Architect for the IBM z/VM® Development team in Endicott, NY. He has 14 years of experience working with System z and z/VM, and has developed or tested code for nearly every layer of the z/VM virtualization product. His primary focus is z/VM Security Development, including virtual machine isolation, SSL and TLS, LDAP, virtualization of hardware cryptography, and RACF for z/VM. He has written security-related articles, and also presents on IBM System z security at conferences around the world.

Brian Hugenbruch is a technical team lead providing day-to-day guidance and advice on the IBM RACF security management system. He has a Master's degree in Computer Application from Sri Venkateswara University in India. He has eight years of experience in mainframe RACF and specializes in z/OS security, particularly RACF and associated products. He is engaged in mainframe security projects in Bangalore, India.

Madhukar Reddy Nali is a Certified Senior IT Specialist on IBM System z. He works on architecture and technical solution development for Mainframe Speciality Services Area in IBM Global Technology Services® (GTS) Delivery Technology and Engineering team. He has a M.Phil degree in Computer Science, Master's degree in Information Technology, and an MBA. Vinod has over 14 years of experience on System z, and he held various roles in IBM since 2004. He also holds a post-graduate diploma in Cyber Law.

Botrous Thalouth is an IBM S/390® Development Consultant with the Cairo Technology Development Center (CTDC). He provides worldwide 2nd and 3rd level support for SM/370, Text Search Engine and Text extenders, z/OS, and DB/DC. He has over 35 years of experience with IBM mainframes starting with S/360 as an application and system programmer. He also worked in the IBM Scientific Center. While he was there, he designed and implemented an Arabic morphology system, the algorithm of which is used by many IT companies. Botrous provides support for and teaches many courses for IBM clients in the Arabian Gulf area for z/OS, z/VM, IBM z/VSE®, Linux for System z, IBM CICS®, IBM DB2®, and application programming. Botrous has a Bachelor of Engineering degree from Cairo University.

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

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

Richard Conway

International Technical Support Organization, Austin Center

Stacy Anderson, Malcolm Beattie, John Dayka, Manfred Gnirss, Michael Kasper, Mark Nelson, John Petreshock, Karl Schmitz, Peter Spera, Mike Spiegel, Jan Thielmann, Maria Tzortzatos, Bruce Wells

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts>

- ▶ 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>



Part 1

Direction and architecture

This part positions the current landscape for mainframe security in a holistic concept and tells you what you can expect in this IBM Redbooks publication series.

This part introduces you to the IBM Security Framework and IBM Security Blueprint as a product-neutral means to understand business-driven security. This part look at the history of security on the IBM mainframe and describes its hardware and software components in a security context.

The final sections of this part of the book explain different security functions, which are based on the IBM Security Framework, and security threats and risks.



Introduction: Why these books are being written

In 2010, IBM instituted a project to write an IBM Redbooks publication focusing on security for the IBM mainframe. This book looked at the IBM Security Framework and related it to the IBM System z and its operating systems. It was entitled *Security on the IBM Mainframe*, SG24-7803. The project produced a deliverable that was of real value to many who learned the capabilities of the widely used System z mainframe.

The project team produced a book that explained much of the System z security architecture and showed how the server components, such as the Hardware Management Console (HMC) and IBM Process Resource/System Manager (IBM PR/SM™), contributed to making System z one of the most highly respected computing platforms in the world. It showed the breadth of the security capabilities of the operating systems that run on System z. Although it had a major focus on z/OS, it also addressed IBM z/VM and Linux and touched on the other operating systems that are available, such as IBM z/VSE and z/TPF.

At that time, however, it did not tackle the question of how to configure and securely organize a mainframe (that is, an IBM System z server and its operating systems) within the larger organization. It did not address the risks that are found in today's IT environment. It was *theoretical*, rather than *practical*.

This book is the first volume of an expanded set of three books looking at the same subject but covering more aspects of that subject.

In 2013 and 2014, System z has moved on. Several new System z servers are available: the z114, the zEC12, and the zBC12. These servers are more powerful than their predecessors. They offer new features for enhancing the System z traditional qualities of service. They provide new security capabilities as well.

Also, the world has moved on. Cyber security is a large concern to many organizations, whether they are commercial, governmental, military, and so on. There are unsolicited access attempts to the data on those mainframes through illegitimate acts of cybercrime.

Cybercrime is a sophisticated activity. It is no longer a playing field for “script-kiddies” trying to get access to systems and servers for fun, and it is not about quick hacks to get in and get out quickly. It is now about real commercial, political, or even military advantages. *Hacking* (quite a vague term in itself) is only part of the process that is used criminally to gain access to data that is held within organizations and enterprises. If access to one part of an enterprise is gained, then this access can be used to gain access to another part of the same enterprise. The breach of the security controls is used to create or obtain another breach. Data is leaked carefully over a long period. Breaches are used and controlled by attackers so that detection is difficult.

This level of activity is no longer associated only with those smaller distributed servers, which frequently run Linux, UNIX, or Windows based software. The goal in many of these attacks is to access the data that is held on the mainframe. Why would the mainframe be under attack? Well, many years ago, the infamous American bank robber John Dillinger was asked why he robbed banks. He replied, “Because that is where the money is.”¹ So, the answer to why the mainframe is now under attack is that this is the server where many organizations store and process their most valuable data.

There have been reports in the press recently of large systems data breaches, and it is apparent that some of this is associated with attempts to access mainframe data. In consequence, it is of real value for us to again consider the security capabilities of the mainframe. This time, however, we want to ensure that our clients know how to configure these machines so that they are highly resistant to attacks. If *resistance* is not possible or practical, you must understand where *detective controls* can be used. If detective controls are not possible, then you must understand what *forensic capabilities* are possible.

Some of the attacks are social in nature. For example, tricking an employee into divulging authentication credentials is a relatively simple way in which to attack any system. There is nothing mainframe-specific in such an attack. The processes can be used to gain credentials for any sort of system. However, it is also apparent that the skills and knowledge that are required to manage and operate a sophisticated IBM System z mainframe are different from those that are used by many people who run an Intel-based Microsoft Windows or Apple Macintosh system at home. It is also different from the skills of those professionals who maintain banks of Linux, UNIX, or Windows servers for commercial organizations.

The complexity of TSO access, JCL, started tasks, and the ways in which various organizations use System z over many years, means that there are now fewer people with the knowledge and skills to even attempt trying to break into a mainframe system. However, as criminal organizations realize the benefits of gaining access to mainframe data, so the efforts to gain access increase.

So, the need to secure mainframes and their valuable data exists within an enterprise in which there are many other types of processors. It exists within a network of firewalls, filters, DMZs, and application gateways, all of which can make up layers of defense for the enterprise.

It has been stated that z/OS running on System z is the most secure commercial operating system available. We (the authors) beg to disagree. If configured properly, it is as secure as claimed; however, z/OS can also be configured and run in a highly insecure manner. It can be run so that all users have access to all data, which is not highly secure. However, it is still *highly securable*; the controls in z/OS can be altered so that high security restrictions can be put in place.

So, we might more accurately claim that z/OS is the most securable commercial operating system available.

¹ Also attributed to Willie Sutton.

It is the difference between *secure* and *securable* that is really the reason for existence for this series of books. We, the authors, aim to give the security professional, or the enterprise security architect, an understanding of System z security. We also aim to give guidance about how that security can be established and maintained. Some of this is process and procedure; some is confined to influencing *wetware*² to behave in a *security conscious* manner. However, there are also the technical aspects of configuration, the need to choose one option over another, and the understanding of the benefits of method A over method B. It is this last matter that we are aiming to consider here.

1.1 Scope of these books

These books must cover the use of the mainframe as it is used in organizations, which means that the mainframe might be used with other platforms that might normally be considered *less secure*. Nearly all organizations use distributed servers. These servers handle tasks that are not normally considered appropriate for the mainframe.

1.1.1 In scope

The System z mainframe server can be any of those in use today. Thus, any System z or IBM zSeries server can be the target. However, we might, along the way, emphasize the security capabilities of later servers in preference to those earlier ones, such as the z900 and z800.

From the operating system or the software perspective, we cover z/OS, the most widely used operating system on the platform, and Linux on System z and the z/VM virtualization technology. Also, we point our examples and references to IBM RACF and other commonly used IBM software products. Those installations with other third-party products available in the market should try to relate the concepts that we explain here to the architecture of those products and their equivalents to the products referred in these books.

1.1.2 Out of scope

In 2010, the System z environment was enhanced with the addition of zBX frames and the IBM DB2 Analytics Accelerator (IDAA) frame. A new server architecture was introduced that allows the new zBX frame to closely integrate with the System z server using integrated networks. The zBX frame can be attached to the latest mainframes and can host Intel blades running Linux or Windows, IBM System p® blades running AIX®, or IBM WebSphere® DataPower® appliances. The configuration, management, and network access to all of these zBX servers and devices is managed by a large extension of the microcode of the System z and the HMC. This extension is known as the Unified Resource Manager (sometimes known as the zManager). The collective name for this extension to System z is IBM zEnterprise®.

1.2 Enterprise security

We examine the use of the mainframe together with various other servers that might act as application gateways, web servers, and so on. In a large environment, there are firewalls and other network security devices. There is an email system, and thousands of client systems running Windows or Linux-based operating systems within an organization.

² Humans; you and me; employees.

The System z mainframe might be running z/VM to host Linux servers. There might be one or more z/OS LPARs in a sysplex, even with z/OS under z/VM for testing or software maintenance purposes.

So, there are many aspects to the security of such an environment. To secure such an environment, you must consider the following subjects:

- ▶ Identity management
- ▶ Access to data and other resources
- ▶ Logging and auditing
- ▶ Security intelligence
- ▶ Application security (and its development)
- ▶ Processes and procedures
- ▶ Physical security of the various servers and network components

1.3 IBM Security Framework and IBM Security Blueprint

These matters of business-driven enterprise security are encapsulated in a concept that is known as the IBM Security Framework. The IBM Security Framework provides a business view of the security posture of an enterprise. It is a high-level view, but it incorporates all that is necessary for consideration. Figure 1-1 shows the IBM Security Framework.

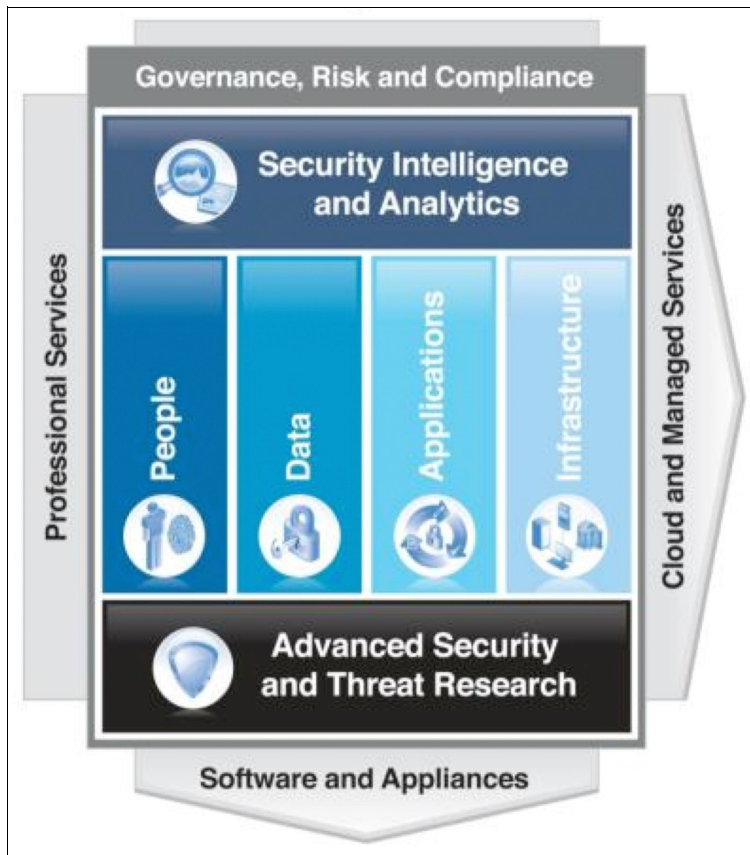


Figure 1-1 IBM Security Framework

It shows the four vertical security domains of People, Data, Applications, and Infrastructure, and the two horizontal security domains of Security Intelligence and Analytics, and Advanced Security and Threat Research. Those two horizontal domains apply to all four of the vertical domains. The underlying concepts of Governance, Risk, and Compliance drive and control all of the security domain activities within an organization. This whole structure is then overlaid on a base of professional and managed services, and the software and appliances on which these components run.

This framework tells you much about what security must be applied, and what types of questions you must ask about security matters.

The IBM Security Framework is described in depth in *Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security*, SG24-8100.

IBM has extended its security view to include the IBM Security Blueprint. The IBM Security Blueprint looks deeper into the matters that are raised by the IBM Security Framework, and provides groups of Foundational Components and Subcomponents so that a logical view of the security measures that are needed for designing a secure solution can be established.

This new series of three books about the IBM System z mainframe security does not address all of the security aspects that are described in these books. However, you should be familiar with the concepts of the IBM Security Framework and also the IBM Security Blueprint to understand how those concepts are connected to the preferred practices of mainframe security that are explained in these books.

1.4 Horizontal functions

The new books attempt to cover security functions that were not addressed in the first edition of this book. The functions include the following ones:

- ▶ Identity management

Although the first edition of this book touched on identity management, there was no description of identity management within and across an entire enterprise.

- ▶ Access management

Access management in the first edition of this book was restricted to descriptions about RACF and access controls, which need more explanation.

- ▶ Security intelligence

The first edition of this book contained little or no material about security intelligence, monitoring, and forensics. This book corrects this situation and provides some insight into the role and capabilities of security intelligence solutions.

Security intelligence should not, of course, be applied solely to the mainframe; it must be applied across an enterprise to apprehend potentially malicious activity as early as possible. However, with any multi-layered defense mechanism, there is a need to perform monitoring and real-time analysis of security events at several different points within the IT infrastructure.

Security intelligence solutions can be built on the integrity of the controls within the operating systems.

- ▶ Security compliance

The first edition of this book did not relate many of the security controls to compliance issues. Indeed, there was not much about compliance. This book will rectify this situation and ensure that there is some description of compliance matters, especially regarding some of the more common standards and requirements, such as PCI-DSS, which affect so many organizations today.

1.5 Structure of preferred practices

Although there are many different subjects and components to be tackled in terms of security capabilities, you need to have a structure for the recommendations that are made in this series of books.

Each component of a typical System z environment has a host of settings that can provide for different needs. At the same time, each installation has a different set of security needs. Do not assume that the security needs of a supermarket are the same as those of a bank, unless the supermarket also has banking services within the same infrastructure. If so, then it is likely that the infrastructure might need upgrading from *supermarket status* to *bank status*.

If there are security measures that can be taken, then they are there to counter a risk. It is not always possible to identify the exact set of circumstances, but it is reasonable that we should try to explain the risk, and so give some insight into why a setting is preferred over some other setting. It might also be reasonable to explain whether there is a downside to a setting. Many times, there are conflicts between performance and security, and in times past, organizations have often chosen performance over security. We, the authors, must call out these situations and make sure that you understand where compromises might have been taken.

At least the following items must be considered for each recommendation:

- ▶ What is the threat that this control counters or reduces?
- ▶ How does this control work to reduce risk?
- ▶ Is it a detective control or a preventive control?
- ▶ What are the alternatives to using this control or setting?
- ▶ What prerequisites are there for the implementation of this control?
- ▶ How robust is this control? For example, what types of scenario can prevent this control from being effective? How can an attacker³ attempt to compromise or disable this control?
- ▶ What are the costs (downsides) if any, to use this type of control?
- ▶ What implementation plan might need to be associated with implementing a change to put this control in place?
- ▶ What are the managements considerations that are associated with the change?
- ▶ How does this control align to the IBM Security Framework?

In addition, each organization must consider the applicability of the control to their environment. This can be judged in the light of the organization's tolerance for risk. However, this is not something we can assess.

³ An attacker is the person or persons attempting to gain unauthorized access to a system or data.

In addition to these assessments of security controls, there is a need to examine other decisions regarding software or hardware configuration. Frequently, decisions are made without reference to any security consultation simply because it has not been recognized that there is a security aspect to the decision. One such example in the z/OS environment relates to DASD sharing, which is done to provide a recovery capability. However, such decisions must be made in consultation with a security officer or architects. There is a real danger of decisions being assumed to have no security context and hence providing weaknesses for an attacker to use.

1.6 Risk-based approach

Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security, SG24-8100 defines risk management in terms of Risk Identification, Risk Analysis, Risk Controlling, and Risk Reporting. We use these terms in as they are used in that book.

1.7 Structure of this IBM Redbooks series

This book is intended to be the first of three volumes that gives a holistic view of security on the IBM System z mainframe.

The first volume (this book) gives an overall introduction to the mainframe security architecture. It also includes the basic technical overview of the mainframe components, including its hardware, operating systems, and software functions. This first half of this book presents some of the contents from the previous edition of *Security on the IBM Mainframe*, SG24-7803. Part two of this book covers guiding principles for security of the System z server and the operating systems that it hosts.

The second volume looks primarily at networking and communications server security. It examines the architecture of the networking security capabilities, and then moves on to describe the preferred practices for all the components of networking security, including IBM VTAM® configuration, TCP/IP configuration, Intrusion Detection Services, AT-TLS, IPsec, FTP, and similar components. These subjects must be covered for all applicable operating systems of the System z platform.

The third volume moves away from the operating systems and looks at some of the major software packages. Most of these software packages run on z/OS. For example, DB2, IBM CICS, IBM IMS™, WebSphere MQ, WebSphere Application Server, and so on, are major components that are found on many deployed systems. However, there are other infrastructure and automation components must be addressed, such as IBM Tivoli® Workload Scheduler, Tivoli NetView® for z/OS, System Automation for z/OS, and so on. In each case, we show the security architecture of the software component, and then show what preferred practices there are for the component.

Keep your eyes open for Volumes 2 and 3, which will be released at a further date.

1.8 Conclusion

We conclude here by recommending this first book and its two companion books to you, and wishing you success in reducing the risks in your organization and securing it all against those *bad guys*.



Foundation of a holistic security architecture

This chapter introduces the IBM Security Blueprint and its relevance to System z. It sets out a high-level, business-oriented view of the security landscape.

The Security Framework is technology-neutral, providing a guide to *what* solutions are needed rather than how they are delivered. Solutions for problems in a domain tend to share characteristics, and can be evaluated using common criteria.

The next step after the IBM Security Framework's business view is the *IBM Security Blueprint*, a map of security concerns and responses for technical architects. Designed primarily for use by technical stakeholders, the Security Blueprint goes beyond a strategy view to add a more detailed operational and technical view. Although the Security Blueprint is still product-neutral and platform-neutral, it provides a guide to the architecture of security systems.

The Security Blueprint provides system views, component catalogs, and solution patterns that can be used to build and analyze secure IT systems.

Much of the descriptions in this chapter are taken from *Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security*, SG24-8100. That book gives a more detailed description of both the Security Framework and the Security Blueprint and shows how they can be used in designing secure systems and applications.

The aim here is to introduce the IBM Security Framework and the IBM Security Blueprint as concepts that are as relevant to the System z environment as to any other environment. For more information about on these two valuable concepts, see *Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security*, SG24-8100.

This chapter covers the following topics:

- ▶ IBM Security Framework
- ▶ IBM Security Blueprint
- ▶ Conclusion

2.1 IBM Security Framework

Today, any business initiative inside an organization is guided by the principles of *Governance, Risk, and Compliance*, which are often seen as broad terms that have different meanings to different stakeholders across an organization. Each CxO is trying to manage risk for their domain, and therefore has different priorities and points of view when it comes to handling these risks:

- CRO** The *Chief Risk Officer* looks at the organization's overall risk profile and where they are most vulnerable to unexpected loss.
- CFO** The *Chief Financial Officer* must ensure that the necessary controls are in place to have accurate financial statements.
- CISO** The *Chief Information Security Officer* must ensure that the IT Infrastructure supports the overall business drivers of the organization. The CISO must minimize the risk of the IT environment and assess and communicate the impact of this environment on the overall organization from a Governance, Risk, and Compliance perspective.

Regardless of the organizational perspective of risk management, both process and IT controls must be established to get a complete picture of the organization's *risk posture*. Establishing IT security controls, monitoring these controls, mitigating the risk observed through those controls, and reporting and communicating risk posture are critical capabilities for an IT security organization.

IBM created the IBM Security Framework to help ensure that every necessary IT security aspect can be properly addressed when you use a holistic approach to business-driven security.

Figure 2-1 shows the IBM Security Framework.

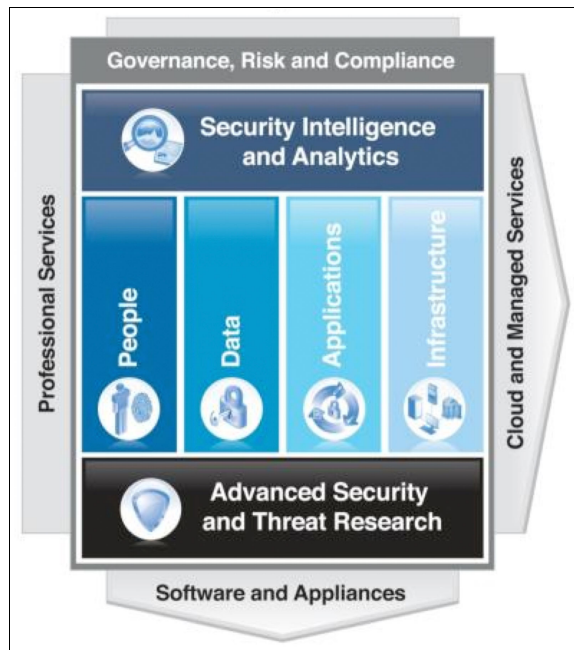


Figure 2-1 IBM Security Framework

The capabilities that are described by the IBM Security Framework are based on a foundation of the Advanced Security and Threat Research infrastructure. The solutions are provided within the security domains, and additional layers can be delivered through software, hardware (including appliances), and as services, whether managed, professional, or cloud-based.

IBM provides the full breadth and depth of solutions and services that can enable organizations to align this business-driven, secure by design approach to security with the IBM Security Framework.

The following sections take a closer look in to the layered model of the IBM Security Framework in the following areas:

- ▶ Advanced Security and Threat Research
- ▶ People
- ▶ Data
- ▶ Applications
- ▶ Infrastructure
- ▶ Security Intelligence and Analytics
- ▶ Security Maturity Model

2.1.1 Advanced Security and Threat Research

The threat landscape continues to evolve, and attacks continue to grow in number and complexity, as does the resulting business loss. It is critical to ensure that an advanced research team is being used to *stay ahead of the threat*. More than ever, ongoing research and development are imperative in ensuring that security solutions remain effective.

Advanced Security and Threat Research addresses the needs of the security market and can provide a foundation for understanding threats, their sources, and how to effectively respond to these attacks. Organizations are facing an explosion of data that must be incorporated into this research. For example, social media, custom malware, geo-location, advanced analytics, and the resulting targeted advanced persistent threats can dramatically increase the range and depth of the data that must be considered in Advanced Security and Threat Research.

It can be challenging for a typical organization to attempt to perform even a small portion of this overall research. There are many organizations that specialize in this type of research, including IBM.

These organizations research and monitor the latest Internet threat trends, develop security content for use in security products, and help advise organizations and the general public about how to respond to emerging and critical threats.

To be effective, such research groups can benefit from access to live customer data (for example, live monitoring of managed security services traffic). They need a global view where both global event monitoring and a global reach are important aspects. It is important to participate in industry consortiums and work with the appropriate government entities to stay ahead of the latest trends.

Figure 2-2 shows a summary of (and additional aspects to be addressed within) the Advanced Security and Threat Research domain.

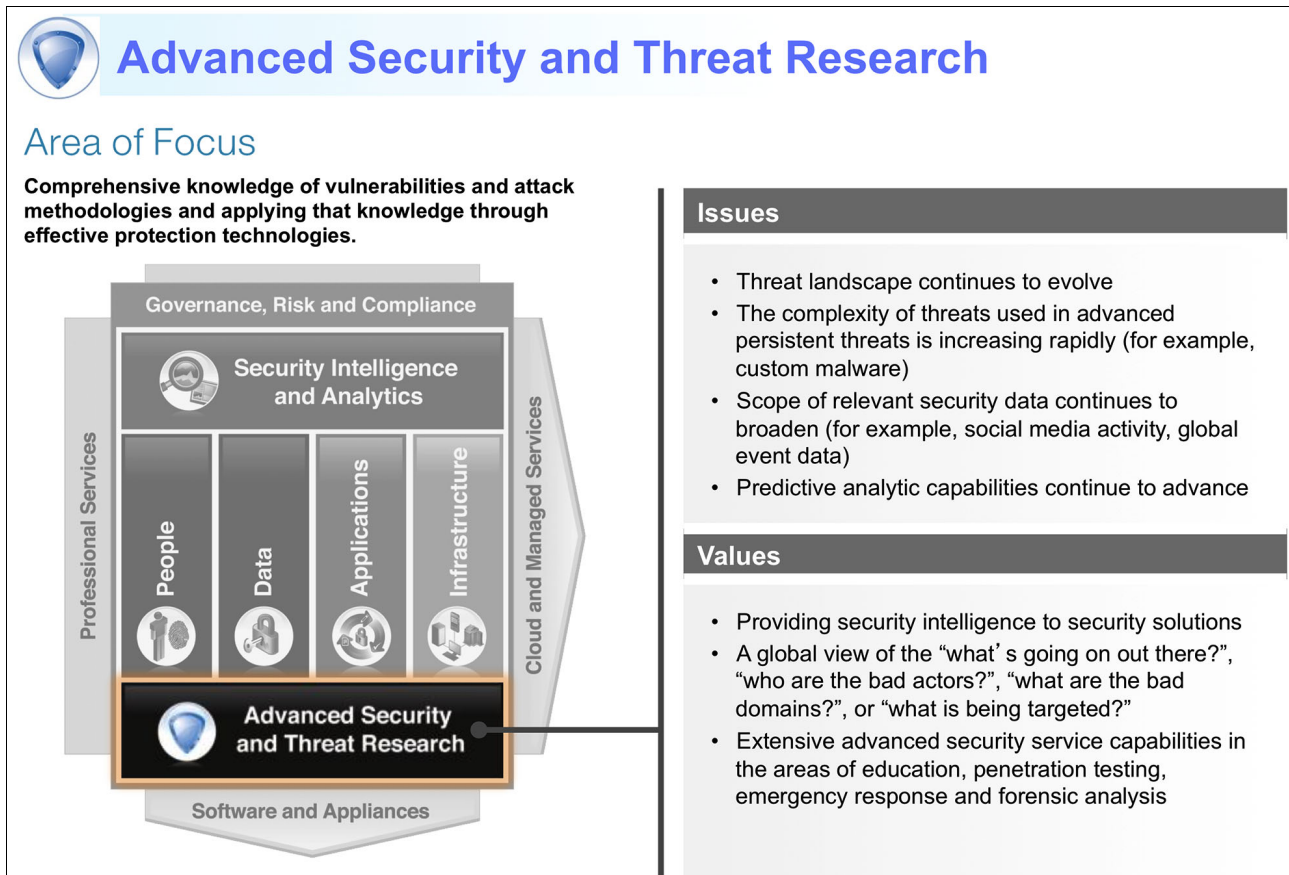


Figure 2-2 Advanced Security and Threat Research domain

2.1.2 People

Organizations must protect the assets and services that serve the business and support the business operations. One aspect of protection is provided by *access control*. The ability to provide effective access control services is based on the ability to manage people and identity. Enterprises define access control models, which are relationships between people and identities that are expressed in terms of *role*, *rights*, *business policies*, and *rules*.

Operationally, people acting in authorized roles in an organization or as part of an extended relationship are granted access (or *rights*) to infrastructure, data, information, and services. At the same time, people acting in unauthorized roles are denied access if they are acting outside of the business policies and agreements.

Identity systems must be integrated with appropriate sets of access controls. Unless multi-platform identity systems are available to manage user roles, rights, and privileges across the IT infrastructure, the presence of multiple technological architectures require multiple identity and access control systems to ensure that users have access to the correct assets and services.

Within an identity system, people can be issued a *credential* to prove their identity to IT systems. A credential can take several forms, including a physical identity card or logical token. The *trustworthiness* or *strength* of the credential is an important aspect of business policy and risk management. The ability to effectively manage the lifecycle of an identity, that is, the creation, removal, and role changes for dynamic populations of workforce, customer, or user communities, is important. The lifecycle of identities and credentials can be influenced by, for example, business cycles, employment cycles, and customer relationships.

Often, identity systems must manage user roles, rights, and privileges across a heterogeneous environment that consists of multiple architectures and technology implementations within an IT infrastructure. Identity systems must be integrated with the appropriate sets of access controls. Especially in these complex situations, Identity Management systems can help mitigate the risk of dormant, expired, or shared IDs.

Compliance in an identity and access context is often externally motivated. For example, legislated privacy and evidence recording is a driver for the implementation of comprehensive user provisioning and identity-related record keeping.

Every organization must maintain privileged user access activity to ensure that no high-level access is being misused for malicious intent. In this context, it becomes more important to closely monitor privileged access and attribute every action to real people in the organization.

Figure 2-3 shows a summary of (and additional aspects to be addressed within) the People domain.

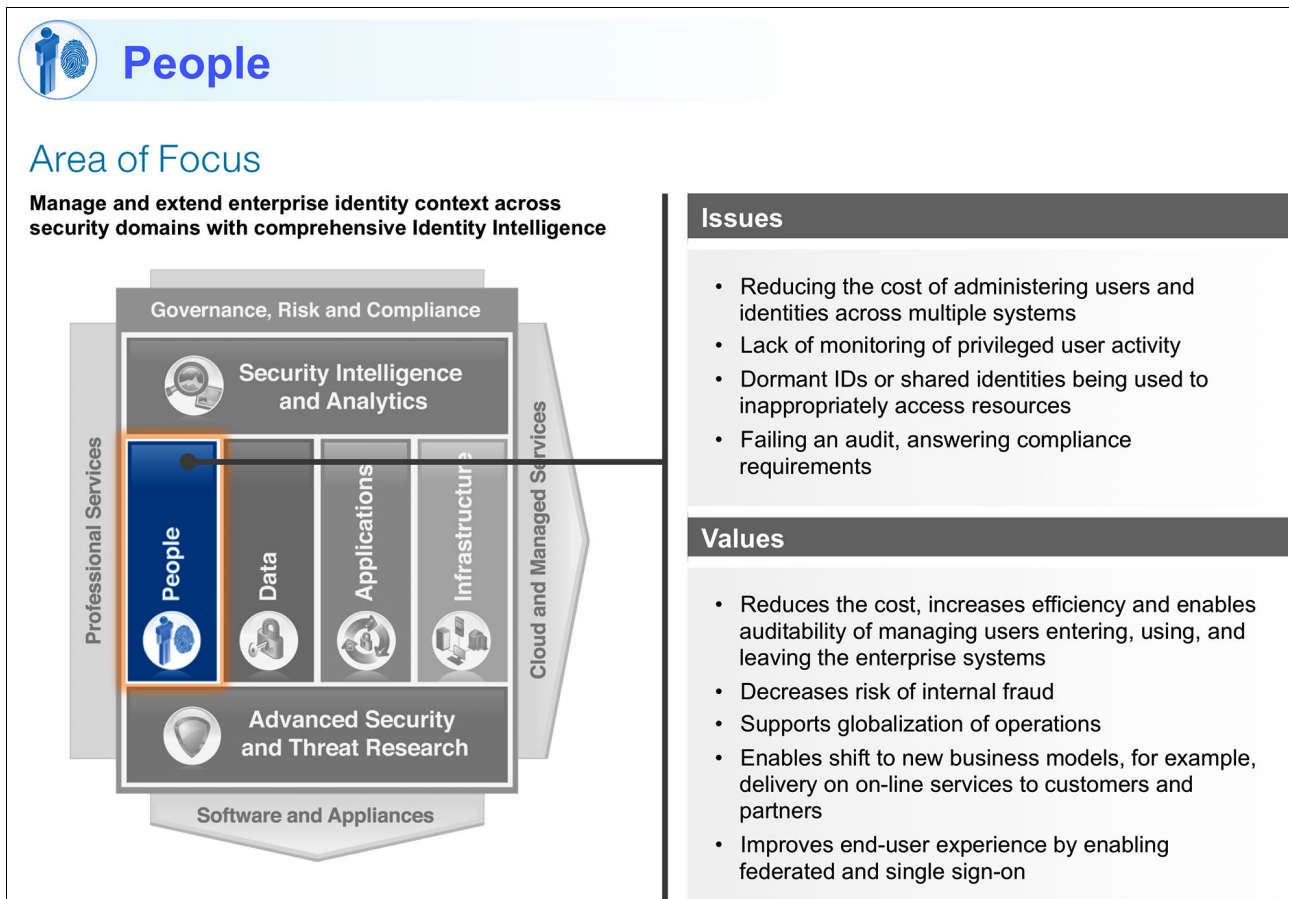


Figure 2-3 People domain

2.1.3 Data

Organizations must protect both the *raw data* and *contextualized information* that is within its span of control. The organization must provide guidance to IT about the classification and value of data and information, and how the risks to data and information must be managed. An effective plan for data and information protection includes maintaining a catalog or inventory of these assets, along with attributes, policies, enforcement mechanisms, and services that govern the access, transformation, movement, and disposition of data and information.

This data and information protection plan can be applied to business processes, business transactions, or business and infrastructure support processes. The protection of data and information covers the full information lifecycle, from creation to destruction across its various states, locations, and instantiations, and data that is stored (*at rest*) and data that is being physically or electronically transported (*in motion*).

The term *data* can be applied to a wide range of electronically encoded assets. These assets include software and firmware, which must be protected against technical risks (for example, to ensure that malicious code is not introduced) and business risks (to ensure that licensing terms are not violated). It also includes other types of intellectual property, such as plans and designs, and both structured and unstructured forms. Monitoring data access and protecting data from unwanted leakage or loss (*data loss prevention*) is made even more complex in today's changing environment of cloud delivery and mobile computing platforms.

Measuring and reporting on an organization's compliance regarding protection of data and information is a tangible metric of the effectiveness of the enterprise security plan. A *data and information compliance report* reflects the strength or weakness of controls, services, and mechanisms in all domains.

Figure 2-4 on page 17 shows a summary of (and additional aspects to be addressed within) the Data domain.

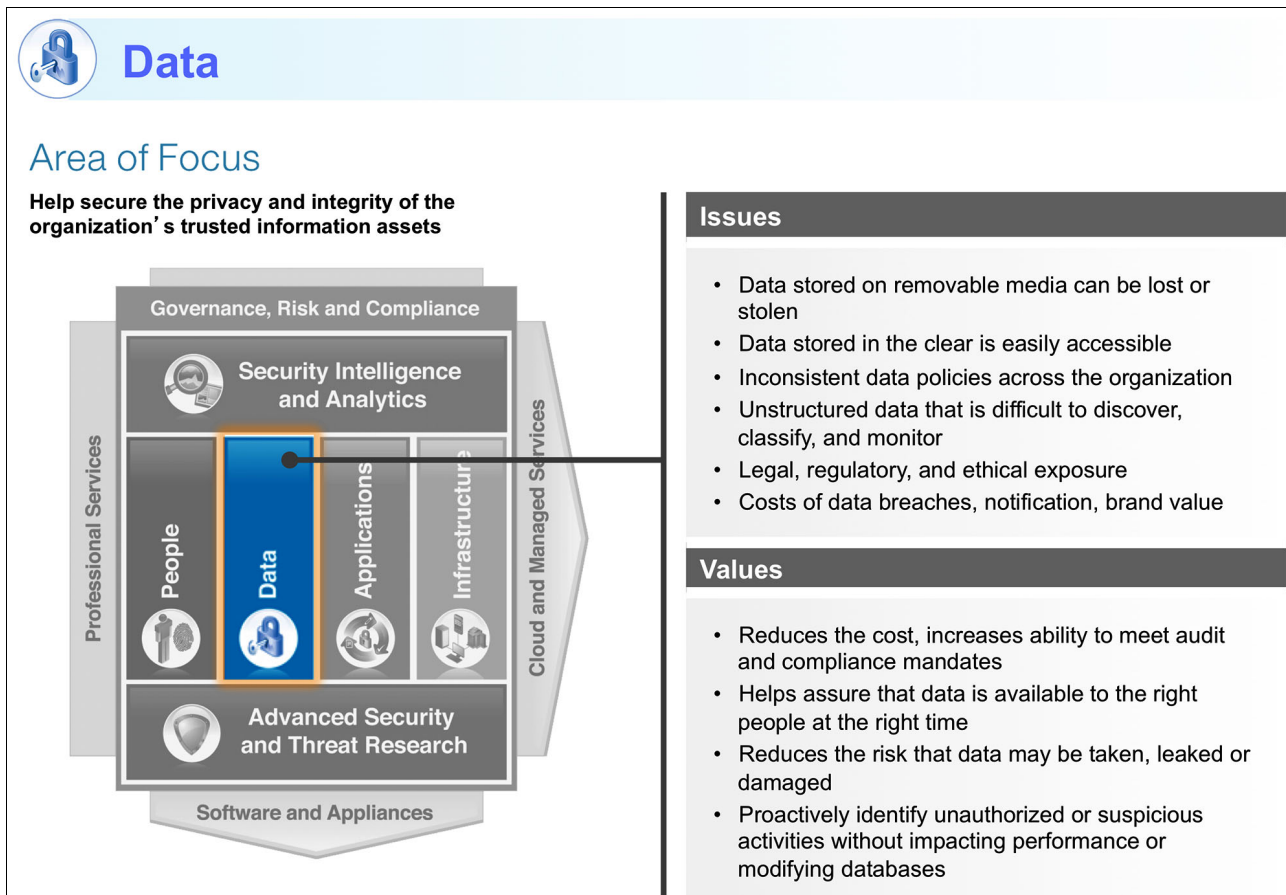


Figure 2-4 Data domain

2.1.4 Applications

It is imperative to have a layered approach to security that ensures that all security domains are appropriately addressed in an IT security infrastructure. One of the most effective ways to avoid a breach is to make sure that the applications your users are accessing are designed and implemented securely.

Organizations must proactively protect their *business-critical applications* from external and internal threats throughout their entire lifecycle, from design to development, test, and production. For example, whether an application is internally focused, such as a customer relationship management (CRM) system, or is an externally facing application, such as a new customer portal, clearly defined security policies and processes are critical to ensure that the application enables the business rather than introducing more risk.¹

Automatic scanning of application source code and vulnerability testing of operational systems can help to identify weaknesses that might be used by an attacker before a security incident occurs.

¹ To learn more about how IBM looks at the secure development lifecycle, download *Security in Development: The IBM Secure Engineering Framework*, REDP-4641, found at: <http://www.redbooks.ibm.com/abstracts/redp4641.html?Open>

Figure 2-5 shows a summary of (and additional aspects to be addressed within) the Application domain.

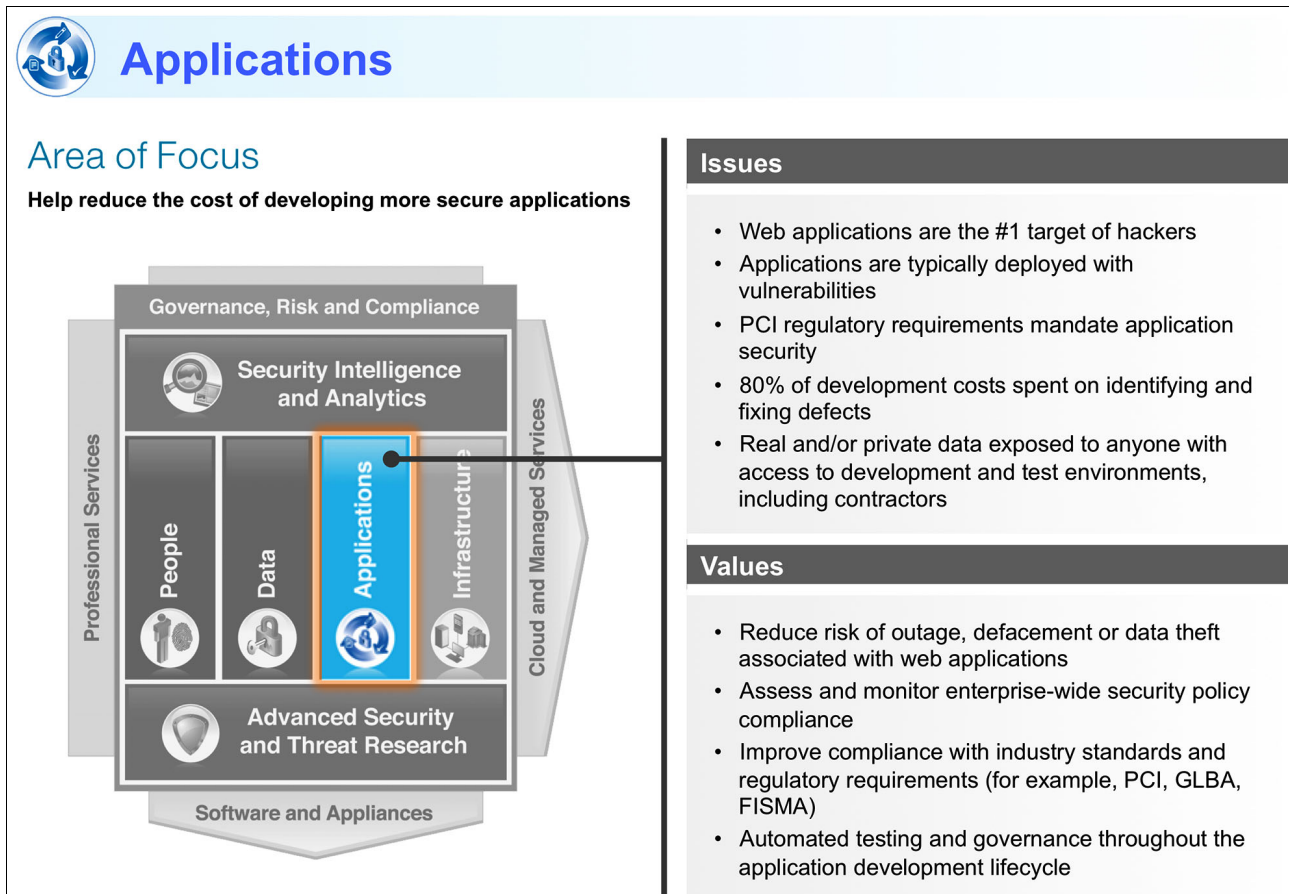


Figure 2-5 Application domain

2.1.5 Infrastructure

Organizations must *preemptively* and *proactively monitor* the operation of the business and the IT infrastructure for *threats* and *vulnerabilities* to avoid or reduce breaches.

Security monitoring and management of an organization's network, servers, and endpoints is critical to staying ahead of emerging threats that can adversely affect system components and the people and business processes that they support. The need to identify and protect the infrastructure against emerging threats has increased with the rise in organized and financially motivated attacks. Although no technology is perfect, the focus and intensity of security, monitoring, and management can be affected by the type of network, servers, and endpoints that are deployed in the IT infrastructure and how those components are built, integrated, tested, and maintained.

Organizations are increasingly using *virtualization technology* to support their goals of delivering services in less time, with greater agility, and at lower cost. By building a structure of security controls within this environment, organizations can reap the goals of virtualization, such as improved physical resource use, improved hardware efficiency, and reduction of power costs, while they ensure that the virtual systems are secured with the same rigor as the physical systems.

In networks today, organizations also are faced with hundreds of new web and non-web applications that are available to their users. Social media applications, peer-to-peer file transfer applications, Voice over Internet Protocol (VoIP), web-based email, cloud data storage, and many others are all readily available. The ease and speed at which these new, and often weak, applications can be installed or simply accessed can reduce the effectiveness of a perimeter-based security architecture and provides many new types of risks. A more integrated infrastructure security solution must take these new threats into account.

Securing an organization’s overall infrastructure can mean taking precautions against a failure or loss of physical infrastructure assets that might impact business continuity. This security can involve protection from indirect threats and vulnerabilities, such as the impact of the loss of a utility service, a breach in physical access control, or a loss of critical physical assets. Effective physical security requires a centralized management system that allows for correlation of inputs from various sources, including property, employees, customers, the general public, and local and regional conditions. Although securing physical infrastructure assets is as important as securing IT infrastructure assets, the remainder of this book focuses on the IT aspects of this security domain.

Figure 2-6 shows a summary of (and additional aspects to be addressed within) the Infrastructure domain.

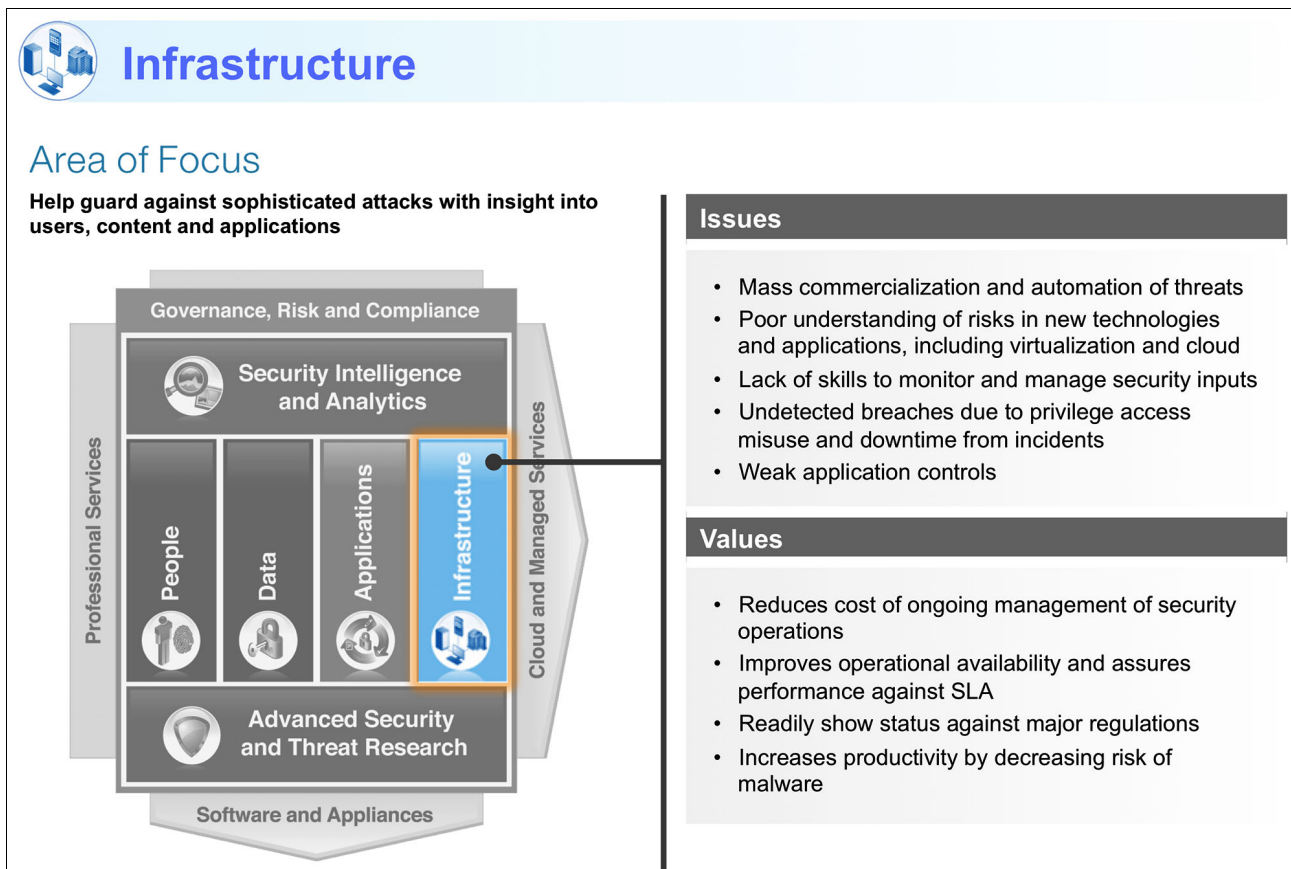


Figure 2-6 Infrastructure domain

2.1.6 Security Intelligence and Analytics

Security Intelligence and Analytics provide a layer of discovery and reporting on top of the security domains, that is, a control center for logging, viewing, analyzing, alerting, and reporting on events across, rather than within, domains. It provides a unified collection, aggregation, and analysis architecture for application logs, security events, vulnerability data, identity and Access Management data, configuration files, and network flow telemetry from security applications and devices throughout the security domains. In addition, the Security Intelligence and Analytics layer provides a common platform for all searching, filtering, rule writing, and reporting functions, and a user interface for log management, risk modeling, vulnerability prioritization, incident detection, and impact analysis tasks.

The comprehensive nature of the security intelligence architecture implies the collection of vast amounts of both real-time and historic data for alerting and forensic analysis. The reduction of potentially billions of items of security data into a manageable set of actionable items, and the identification of patterns of behavior that fall outside the norm, are the province of the analytics within this layer. This action requires the ability to correlate data and scale across multiple data types and sources beyond the ones that are found in the individual security domains.

Figure 2-7 shows the Security Intelligence and Analytics layer in the IBM Security Framework.

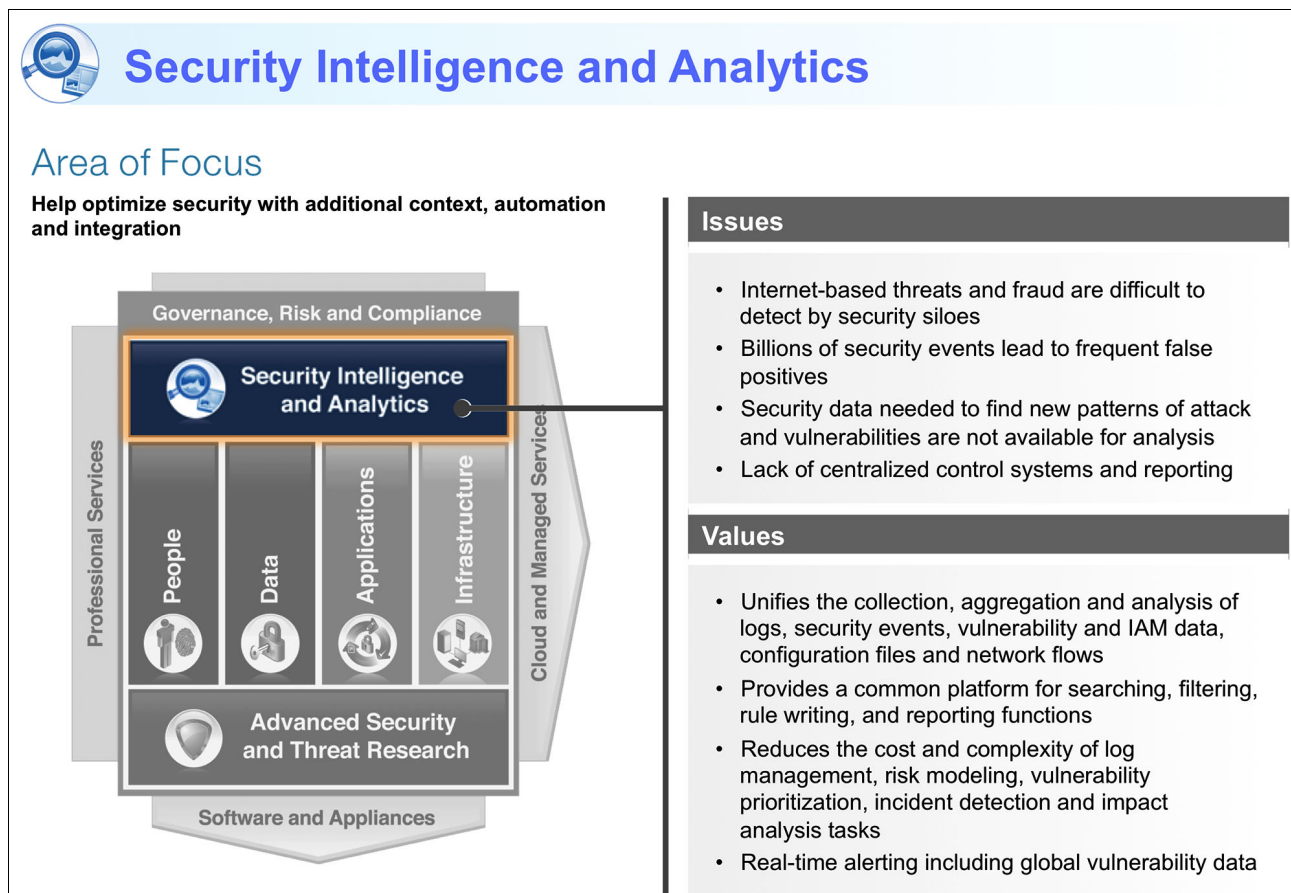


Figure 2-7 Security Intelligence and Analytics layer

2.1.7 Security Maturity Model

The IBM Security Framework provides a reference model for the security domains by which an organization can protect its people, data, applications, and infrastructure. Within each domain, there can be varying degrees of protection, from minimal, or basic, through proficient to optimized.

Organizations vary in their ability or intent to deploy security solutions depending on their budgets, experience, skill sets, and risk appetite. They might have many years of experience with user access controls, for example, with advanced, automated technology that is deployed at the optimized level in the People domain. At the same time, they might be in their first implementation of application scanning, so they are likely to be at a basic level of security in that domain.

To help clarify the security controls that are representative of these maturity levels within each domain, IBM developed the Security Maturity Model that is shown in Figure 2-8.

Security Intelligence ↑	Security Intelligence: Flow analytics / predictive analytics Security information and event management Log management			
	Optimized Identity governance Fine-grained entitlements Privileged user management	Data flow analytics Data governance Encryption key management	Fraud detection Vulnerability correlation Hybrid scanning	Multi-faceted network protection Anomaly detection Hardened systems
	Proficient User provisioning Access management Strong authentication	Data masking / redaction Database activity monitoring Data loss prevention	Web application protection Source code scanning	Virtualization security Asset management Endpoint / network security management
	Basic Directory management	Encryption Database access control	Application scanning	Perimeter security Anti-virus
	People	Data	Applications	Infrastructure

Figure 2-8 The Security Maturity Model

Encryption and database access control are representative of a basic level of data protection, for example. To have an optimized level of data security, the organization must deploy security solutions that provide controls at the basic, proficient, and optimized level that is shown in Figure 2-8. The more optimized the security level, generally the more automated and proactive the security solution is, and the more integrated it becomes with the Security Intelligence and Analytics layer that sits above and across all of the domains.

2.2 IBM Security Blueprint

The IBM Security Framework divides the area of business-oriented IT security into four major security domains and three support layers. The next step is to break down these domains and layers into further detail to work toward a common set of core *security capabilities* that are needed to help an organization securely achieve its business goals. These core security capabilities are called the *IBM Security Blueprint*.

The IBM Security Blueprint uses a product- and solution-neutral approach to categorize and define security capabilities and services that are required to answer the business concerns in the IBM Security Framework.

The IBM Security Blueprint was created after research into many customer-related scenarios that were focused on how to build IT solutions. The intention of the blueprint is to support and help design and deploy security solutions in your organization.

Building a specific solution requires a specific architecture, design, and implementation. The IBM Security Blueprint can help you evaluate these items, but does not replace them. Using the IBM Security Blueprint in this way can provide a solid approach to considering the security capabilities in a particular architecture or solution.

IBM chose to use a high-level service-oriented perspective for the Security Blueprint that is based on the IBM service-oriented architecture (SOA) approach. Services use and refine other services (for example, policy and access control components affect almost every other infrastructure component).

To better position and understand the IBM Security Blueprint, see Figure 2-9.

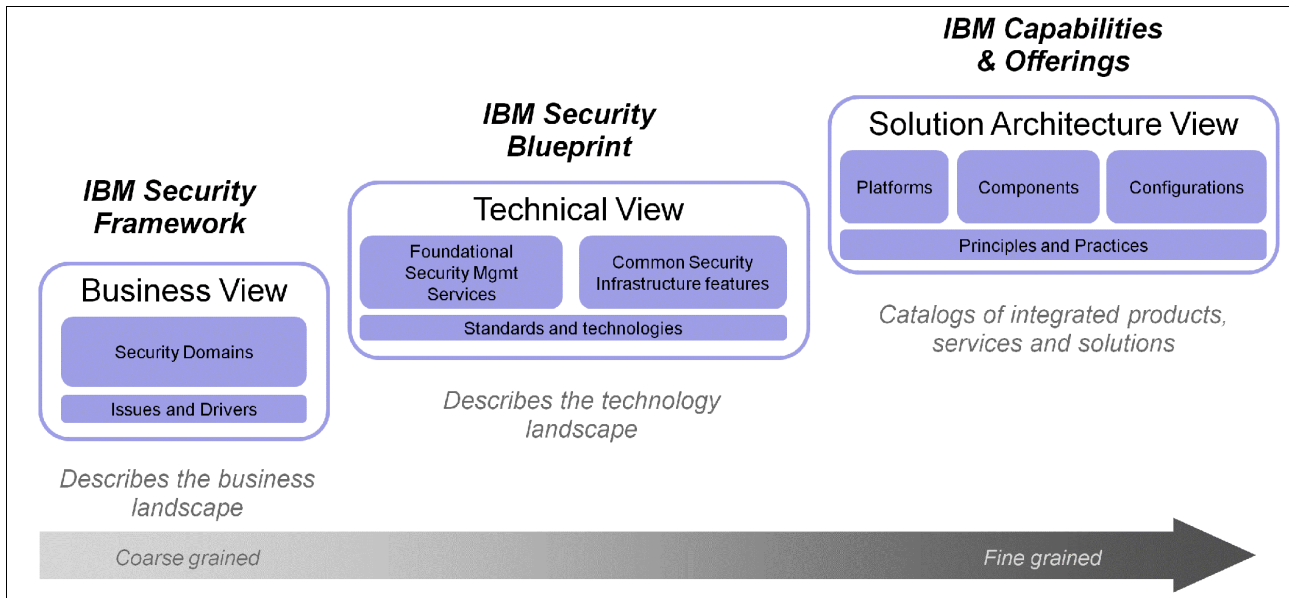


Figure 2-9 IBM Security Blueprint positioning

The left portion of Figure 2-9 represents the IBM Security Framework, which describes and defines the security domains from a business perspective. It is covered in 2.1, "IBM Security Framework" on page 12.

The middle portion in Figure 2-9 on page 22 represents the IBM Security Blueprint, which describes the IT security management and IT security infrastructure capabilities that are needed in an organization. The IBM Security Blueprint describes these capabilities in product- and vendor-neutral terms.

The right portion of Figure 2-9 on page 22 represents the solution architecture views, which describe specific deployment guidance particular to an IT environment and the current maturity of the organization within the respective security domains. Solution architecture views provide details about specific products, solutions, and their interactions.

Figure 2-10² shows the complete IBM Security Blueprint, and each layer and component is described in the following sections.

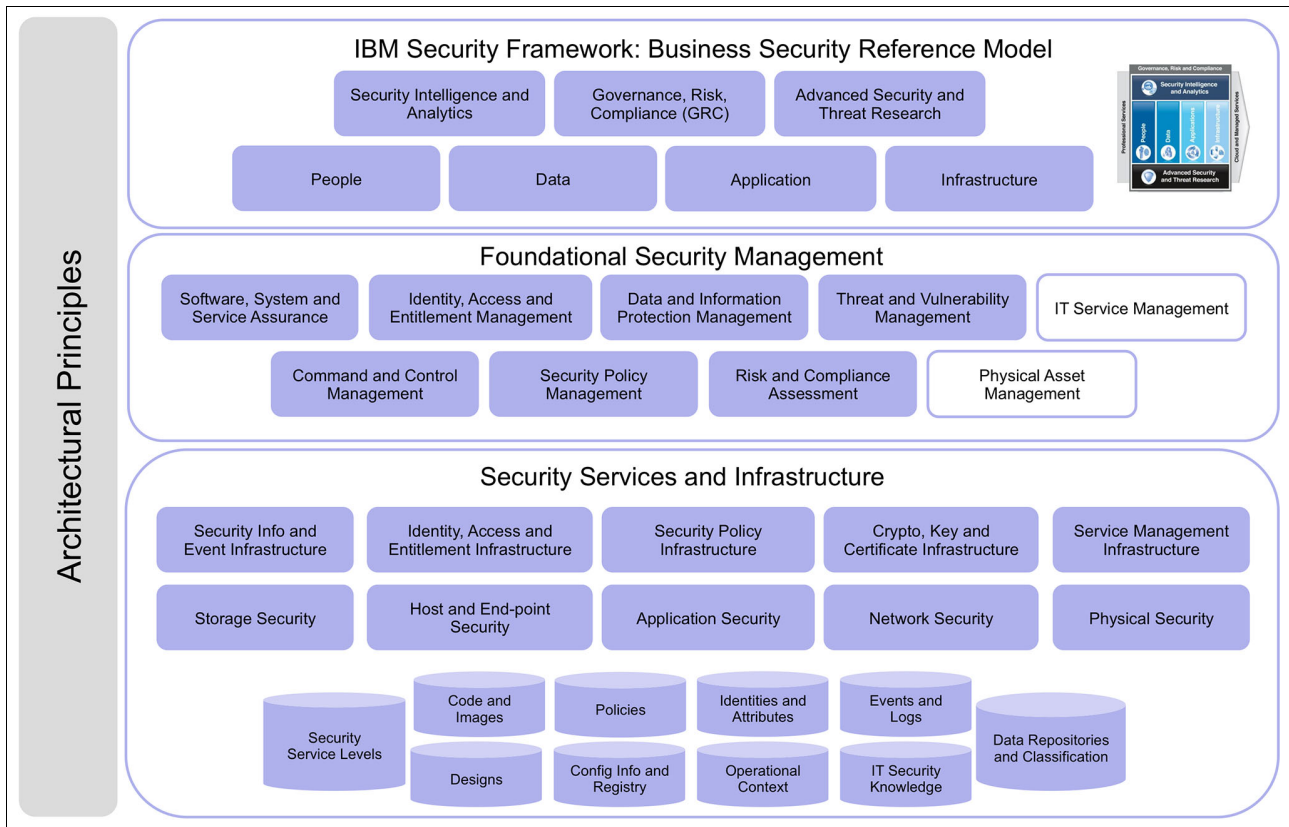


Figure 2-10 The IBM Security Blueprint

² White boxes in Figure 2-10 on page 23 and other diagrams represent services or components that are not solely security-related, but might be connected with other IT service areas as well.

2.2.1 Foundational Security Management

The Foundational Security Management layer contains the top-level components that are used to direct and control IT security from a policy-based, risk management perspective.

Here is a closer look at each Foundational Security Management component:

- ▶ *Risk and Compliance Assessment* enables the IT organization to collect, analyze, and report security information and security events to identify, quantify, assess, and report on IT-related risks that can contribute to the organization's operational risk. Security dashboards provide situational awareness to allow day-to-day risk management. This component covers *risk aggregation and reporting, IT security risk processes, business controls management, resiliency and continuity management, compliance reporting, and legal discovery services*.

- ▶ *Command and Control Management* provides the command center for *security management* and the *operational security capabilities* for IT and non-IT assets and services to ensure protection, response, continuity, and recovery.

Command and Control Management fulfills both a strategic and a tactical role. The strategic role involves defining security policies. The tactical role involves coordination of the security operations. It covers these topics:

- Ensuring that physical and operational security is maintained for locations, assets, humans, environment, and utilities
- Providing surveillance and monitoring of locations, perimeters, and areas
- Providing top-level incidents that are delivered by security intelligence and analytics for further investigation
- Enforcing entry controls
- Providing for positioning, tracking, and identification of humans and assets
- Providing a focal point for continuity and recovery operations

- ▶ *Security Policy Management* provides all the capabilities and repositories to author, discover, analyze, transform, distribute, evaluate, and enforce security policies.

Security Policy Management involves defining the security policies that are aligned with the business goals about how to reach compliance levels and mitigate risks to an acceptable level. It deals with setting up a governance framework to define and enforce policies and measure their effectiveness, reporting back to Governance, Risk, and Compliance.

- ▶ *Identity, Access, and Entitlement Management* provides capabilities that are related to roles and identities, access rights, and entitlements. The correct use of these capabilities can ensure that access to resources is given to the correct identities, at the correct time, and for the correct purpose. These services can ensure that access to resources is monitored and audited for unauthorized or unacceptable use.
- ▶ *Data and Information Protection Management* provides capabilities that protect unstructured and structured data from unauthorized access and data loss, according to the nature and business value of information. It also provides use and access monitoring and audit services.
- ▶ *Software, System, and Service Assurance* addresses how software, systems, and services are designed, developed, tested, operated, and maintained throughout the software lifecycle to create predictably secure software. This component covers the following items:
 - Structured design
 - Threat modeling

- Software risk assessment
 - Design reviews for security
 - Source code reviews and analysis
 - Dynamic application analysis
 - Source code control and access monitoring
 - Code and package signing and verification
 - Quality assurance testing
 - Supplier and third-party code validation
- ▶ *IT Service Management* provides the process automation and work flow foundation for security management. In particular, change and release management processes play a significant role in security management.
 - ▶ *Threat and Vulnerability Management* provides capabilities that identify vulnerabilities in deployed systems and receive reports of vulnerabilities from outside sources, determine the appropriate response, and make proactive changes to deployed systems to maintain the security of the deployed system. Other capabilities collect security events and information from a wide range of sources to gain insight and detect possible threats through event correlation and security intelligence and analytics.
 - ▶ *Physical Asset Management* provides awareness of the location and status of physical assets and awareness of physical security controls, and coordinates the security information for physical systems with the IT security controls.

2.2.2 Security Services and Infrastructure

The Security Services and Infrastructure layer contains components and subcomponents that are used by the Foundational Security Management components in their respective contexts:

- ▶ *Security Information and Event Infrastructure* provides the infrastructure to automate log aggregation, correlation, and analysis. It also enables an organization to recognize, investigate, and respond to incidents automatically, and streamline incident tracking and handling, with the goal of improving security operations and information risk management.
- ▶ *Identity, Access, and Entitlement Infrastructure* provides services to manage user provisioning, passwords, single sign-on, access control, and synchronization of user information across directories.
- ▶ *Security Policy Infrastructure* provides services to manage the development implementation of security policies in a consistent manner and automate the deployment of those policies to IT systems.
- ▶ *Cryptography, Key, and Certificate Infrastructure* provides services to perform cryptographic operations efficiently and provides operational processes and capabilities to manage cryptographic keys.
- ▶ *Network Security* consists of multi-layered network security to provide defense in-depth, deep inspection, and analysis of protocols, application level payloads, and user content to protect at all levels of the network stack. It extends to virtual networks for security in modern and heavily virtualized environments.
- ▶ *Storage Security* provides data-centric security capabilities for protecting data in use, in transit, and at rest through isolation and encryption capabilities. It also provides services to catalog and classify storage assets and associate control policies with them.

- ▶ *Host and End-point Security* provides protection for servers and user devices, such as mobile phones, desktop computers, and mobile computers using both host and network-based technologies. This protection integrates into the virtualization infrastructure to provide security for virtual environments. It includes hardware-based attestation of host operating systems (OSes) and system resources to protect against malicious attacks.
- ▶ *Application Security* provides the infrastructure for testing, monitoring, and auditing deployed applications.
- ▶ *Service Management Infrastructure* consists of the infrastructure services to handle service management processes, such as incident, problem, change, and configuration management. Process automation is generic framework-based services to automate IT actions, including security-related activities.
- ▶ *Physical Security* is an IT infrastructure service to create awareness of physical security and coordinate it with IT security. This service can include employee badges, RFID readers, surveillance systems, and associated technology or assets. Physical Security can include automation that is related to surveillance, motion detection, object and human identification and tracking, entry control, environmental system monitoring, perimeter control, and power and utility system monitoring.

2.2.3 Architectural principles

IBM security architects defined the following *Architectural Principles* that accompany the service decomposition. These principles can be applied to all levels of the framework, blueprint, and solution designs, and are also guidelines for IBM products and solutions.

- ▶ Openness

Openness is of primary importance in an enterprise environment. Openness includes support for all major platforms, run times, and languages, support for major industry standards, published interfaces, and algorithms, avoiding *security by obscurity*, documented trust and threat models and support for Common Criteria, and similar formal security validation programs.

- ▶ Security by default

Security must not be an afterthought in IT solutions; security policies must be secure immediately. This situation is helped by a consistent definition and management of configurations, a consistent set of security roles and persona across products, and a consistent security management user interface.

- ▶ Design for accountability

With many requirements in the compliance area, it is important that all security-relevant actions can be logged and audited, the audit infrastructure is scalable to handle these events, and audit information is immutable and non-reputable.

- ▶ Design for regulations

Regulations drive many requirements in IT security projects, and regulations change over time. Handling this situation requires flexible support for the constraints that are set by government regulations and industry standards and traceability between regulations, standards, and business policies, and the security policies that are used to implement them.

- ▶ Design for privacy

In the current age of data sharing, privacy becomes increasingly important. Solutions must highlight the use of personally identifiable information (PII) and corresponding data protection mechanisms and enable the principles of notice, choice, and access.

- ▶ Design for extensibility

Good solutions are component-based and separate the management of mechanisms from the mechanisms themselves to support various mechanisms under the same framework. Deployed systems must allow for the addition and extension of new mechanisms within the existing management framework.
- ▶ Design for sharing

Multiple solutions can share a single IT environment, such as in a shared service center. To achieve this goal, security services and management must be able to span multiple domains, each domain potentially providing its own and independently set security policy, identity, models, and so on. Architectures must explicitly document the assumptions and limitations that are made in terms of span of control.
- ▶ Design for consumability

All security services must be easily used by various audiences. These audiences include programmers who develop and integrate applications with the security services, management systems that create, update, and manage security policies and other security artifacts, and people who manage security activities, audit security activities, and request access to protected resources.
- ▶ Multiple levels of protection

Defense in depth is a general principle that can be achieved by multiple levels of enforcement and detection. Resources must be designed to protect themselves as a first layer of defense. Intrusions can be contained through *isolation* and *zoning*. Multiple levels also minimize the attack surface to the outer-most accessible layer. *Least privilege* is a similar fundamental principle. Finally, the system should incorporate fail-safe principles.
- ▶ Separation of security management, enforcement, and accountability

Security management services (identity, authorization, audit, and so on) are provided through a dedicated and shared security infrastructure, enabling consistent monitoring and enforcement. The enforcement itself (through cryptography, policy enforcement, or physical isolation) is distributed and kept close to the resources.
- ▶ Security-critical resources and awareness of their security context

Resources and actors are aware of their environment (including physical location and logical collocation) and their security status and context.
- ▶ Model-drive security

Models are reflective of the operating environment, common models, and consistent formats for identity and trust, data, policy, applications, security information and events, and cryptographic keys. Models are consistently interpreted across the stack (for example, network identities are linked to application-level identities) and across units (for example, policies and trust are negotiated and understood within a federation). Models are consistently validated against reality (feedback from policy and model discovery).
- ▶ Consistency in approaches, mechanisms, and software components

Two independent layers of protection for one resource might improve security. But using two different mechanisms for the same purpose for two resources increases the chances that at least one of them fails (plus they increase management impact).

The IBM Security Blueprint lists the preferred standards and mechanisms.

2.3 Conclusion

The IBM Security Framework and the IBM Security Blue are tools to enable the security architect to understand the components and facilities that are needed to architect a secure solution. These tools are useful for all platforms and can be applied to systems hosted on System z and to applications that span multiple heterogeneous hardware and software platforms.



Mainframe security architecture in the enterprise

This chapter provides a detailed overview of mainframe security in the enterprise. It starts with introducing the IBM mainframe in a historical perspective from the late 1960s to the present day. This chapter provides an emphasis on the security issues that were faced at each point during this period. This chapter also describes the different workloads that run on the mainframe and their specific security considerations. This chapter explains the virtualization concepts on the mainframe and describes how they are associated with the security.

This chapter then defines the role of mainframe and its components in overall enterprise security architecture. You will see how the mainframe security architecture fits in the enterprise security architecture.

This chapter provides details about the qualities of service of the mainframe, the statement of integrity, and then moves on to demonstrate how the confidence that many have in the IBM mainframe is borne out by the following items:

- ▶ IBM confidence in itself over 50 years
- ▶ The longevity of the platform
- ▶ The concepts of certification as achieved by various mainframe components

This chapter covers the following topics:

- ▶ History of the mainframe
- ▶ Workloads on the mainframe
- ▶ Virtualization
- ▶ Overview of mainframe security architecture
- ▶ Role of the mainframe within an enterprise security architecture
- ▶ Statement of Integrity and certification levels

3.1 History of the mainframe

Many histories of the mainframe have been provided over its long existence. Each has given a different definition of what is meant by the term *mainframe* and has presented a different view.

The history that is presented here serves a specific purpose. It shows that the IBM mainframe hardware and software have grown in terms of their security capability from a relatively simple batch processing machine to handling vast amounts of data and transactions. In a mainframe, these workloads can be owned and run by different identities, each separately, and can produce predictable results. A mainframe also ensures reliability and an unprecedented level of operational integrity that became the foundation for building strong security features.

However, much of the security was built into the hardware in the early systems. For example, the concepts of *storage key separation* and *privileged operational states*, which are explained later in Chapter 5, “Software components” on page 99, existed since the first IBM System/360 machines. The advances in programming over this period have made extensive use of those hardware capabilities to provide the highly configurable security controls that you use today. This demonstrates a remarkable level of foresight by the architects and designers of those early mainframe systems.

In this book, the scope is limited to IBM mainframes, and to the series that commenced with the System/360 series. This chapter looks at the mainframe evolution over five-year periods and describes the typical systems in each era. This chapter shows the application of hardware and software, and highlights the security landscape and threats of that era.

Let us start our journey with the System/360 mainframe of the late 1960s.

3.1.1 Late 1960s

The System/360 was announced in April 1964 and represented the first basic reorganization of the electronic computer by IBM since the development of the IBM 701 systems in 1952. These System/360 mainframes were becoming popular with many clients during this period. They enabled organizations to integrate all of their data processing applications into a single management information system. It had virtually unlimited storage and instant retrieval capabilities that provided management with up-to-the-minute decision-making information.

The System/360 ran the IBM Tape Operating System/360, the IBM Disk Operating System/360, and the IBM Operating System/360 itself, which came in several flavors. They used punch cards for input and controls, and printers for output. Each system had a console with an operator. The only type of processing was batch, and these systems used the Job Control Language (JCL). There were no internetworking capabilities, other than locally attached terminals.

It was during this time that a research and development lab at IBM Cambridge Scientific Center pioneered virtualization with the IBM CP/CMS. CP/67, the Control Program, was the original hypervisor component that created multiple independent virtual machines (VMs), each of them as a full virtualization of the underlying hardware. CMS, which stood for Cambridge Monitor System then, was a light-weight single-user operating system (OS). Later renamed Conversational Monitor System, CMS continues to be a critical component in the z/VM even today.

However, in IBM laboratories, intense development work was taking place to produce the next generation of mainframes. Time-sharing systems were being developed that would free the mainframe from its batch-only restrictions. Virtual storage systems were under development, which would free the machine from the limitations of small memory sizes and enable extensive resource sharing, marking the major breakthrough in mainframes. Even in those early days, extended addressing capabilities of the hardware and software were under investigation. Some of this early work, such as addressing capabilities beyond the original 24-bit limitations, came to fruition only after two decades.¹

Security landscape and threats

At that time, there were no real threats to the data processing environment. This system had a single user (the operator). All access controls were related to physical access.

3.1.2 Early 1970s

IBM System/370 models largely replaced the System/360 mainframes. They had virtual storage capabilities, although true virtual storage memory separation capabilities were still in their infancy compared with today's commercial systems.

The OSs were IBM DOS/VS², IBM OS/VS1, and IBM OS/VS2. OS/VS2 later became IBM MVS, which introduced address space memory separation. IBM CICS was available on all these OSs. DOS/VS used POWER as a spooling system, while OS/VS1 used JES. OS/VS2 used HASP and ASP.

As OS/VS2 release 3 became available, it was named MVS, and, introduced address spaces. IBM issued the *Statement of Integrity*³ for MVS, which was the first of its kind in the industry. IBM VM/370 became formally available for those wanting a VM environment for running multiple levels of OSs.

TCAM and BTAM networking systems were used in some deployments. BTAM was used for IBM IMS, a hierarchical database system, which was available on OS/VS2. TCAM was available to support TSO.

Security landscape and threats

Little software was available to provide any concept of user identification.

TSO was available on a few machines. This provided an early registry of users, named the User Attributes Data Set (UADS), but applied only to TSO and had simple password controls. The concept of access control to resources were still under development.

Some data sets were password protected using the PASSWORD data set, but that was not common. As processing was still predominantly batch, this was not a major issue then.

¹ Interested readers might want to investigate the 32-bit addressing capabilities of the S/360 Model 67, which was made available in 1966.

² Abbreviations: DOS/VS - Disk Operating System/Virtual Storage, OS/VS - Operating System/Virtual Storage, HASP - Houston Automatic Spooling Priority (or Program), ASP - Attached Support Processor, IBM POWER® - Priority Output Writers, Execution processors and input Readers, CICS - Customer Information Control System, VM/370 - Virtual Machine/370, TCAM - Telecommunications Access Method, BTAM - Basic Telecommunications Access Method, IMS - Information Management System, DOS/VSE - Disk Operating System/Virtual Storage Extended, MVS/SE - Multiple Virtual Storage/System Extension, MVS/SP - Multiple Virtual Storage/System Product, MIPS - Million Instructions Per Second, TSO - Time Sharing Option, ISPF - Interactive System Productivity Facility, JES - Job Entry Subsystem

³ Read the z/OS Statement of Integrity at the following website:

http://www-03.ibm.com/systems/z/os/zos/features/racf/zos_integrity_statement.html

3.1.3 Late 1970s

System/370 model 168 was available with about 2.7 MIPS of computing power. The IBM 303x series of processors became available a few years later.

DOS/VS became IBM DOS/VSE. MVS ran JES2 and JES3 spooling systems. MVS 3.7 and 3.8 became available and proved to be the stable versions of that time, and were also widely adopted. MVS later changed to IBM MVS/SE, and then to IBM MVS/SP.

There was growth in use of CICS and IMS for transaction-based systems. TSO, and later ISPF, were widely used for systems programming and application development.

The IBM Systems Network Architecture (SNA) grew in popularity along with the IBM 3705 communications processors. VTAM became an important component of MVS.

Security landscape and threats

The security manager, Resource Access Control Facility (RACF), was introduced in 1976, followed by some competitor products. The early RACF provided optional user authentication using a changeable 8-byte password, together with controls on access to individually selected data sets. Passwords could be supplied at TSO logon and on a PASSWORD statement on a batch job. The TSO submit process could store the password in read-protected storage for inserting it into the submitted jobs using an exit. If this was not done, then the JCL had hardcoded passwords.

Threats were related to access within companies. There was a need to protect computer resources from unauthorized access in each organization over the network.

3.1.4 Early 1980s

MVS/SP was installed on IBM 308x processors in many installations. These were single, dyadic, or quadratic processors, with a possible real storage allocation of up to 64 MB. MVS morphed into IBM MVS/XA with vastly extended memory addressability and new I/O instructions and capabilities. DOS/VSE became IBM VSE/SP, but still with only 24-bit addressability. VM/SP also debuted in this time frame.

IBM DB2 was introduced as the IBM relational database. CICS introduced the capability of a Multi-Region Option (MRO).

IBM 3725 processors replaced 3705 processors and larger SNA networks were created. Many organizations provided individual terminals to each of their developers, rather than having to share.

Security landscape and threats

Security products, such as RACF, rapidly changed from a means to protect a selected set of critical data sets to a means to protect all data sets and multiple types of other resources. The ability to map users to groups helped improve ease of use for managing resource access control. The support for generic profiles provided performance advantages over the use of discrete profiles.

Threat profiles started to grow, with governments starting to face security and privacy issues, mainly because of the expansion of inter-networked systems.

3.1.5 Late 1980s

The IBM 309x processors were made available during this period. The IBM Process Resource/System Manager (PR/SM) added multi-system capability on single processors. IBM MVS/ESA (Enterprise Systems Architecture) was introduced, providing access to data spaces and larger amounts of memory. Advanced Address Space Facility (AASF) provided simplified cross-memory services capability through access registers.

Virtual Storage Extended (VSE) still used 24-bit memory. CICS gained new architecture improvements with multiple task control blocks (TCB). SNA was extended to have interconnection capabilities, providing links between and among multiple organizations, using linked IBM 3745 communication controllers.

The use of VM for more than just virtualization increased as various applications were created for it. Some customers started to use VM for as a development and deployment platform, not only a hypervisor, but also an OS. Middleware products and databases became prevalent for VM when used as an OS.

Security landscape and threats

Security identification (user ID and password) was used for all mainframe access. Access to other applications was secured using products such as RACF, which extended their control over multiple classes of resources. Most installations protected all data sets by default, making it mandatory to log on using a valid user ID and password. Rudimentary capability for security propagation was available.

3.1.6 Early 1990s

Some industry observers were declaring the impending death of the mainframe. For example, one such analyst wrote in the March 1991 issue of *InfoWorld*, “I predict that the last mainframe will be unplugged on March 15, 1996.”⁴

But IBM, along with many of its customers, believed that this way of computing always would be in demand. IBM continued funding research and improving the architecture of the mainframe.

Typical systems included MVS/ESA running on IBM ES9000 processors with 128 MB of storage. IBM MVS/ESA OpenEdition became available, extending UNIX capabilities to a mainframe. This opened the gateway for the mainframe to the growing network of personal and midrange computers over TCP/IP and other networking protocols, such as Novell NetWare and Netbeui. Most large organizations still had SNA network interconnections for their mainframes.

The three members of the VM family (IBM VM/SP (System Product), IBM VM/SP HPO (High Performance Option), and IBM VM/XA (Extended Architecture)) converged into a single product that is called IBM VM/ESA, with both 31-bit real and virtual addressing.

MVS/ESA V3.1.3 provided massive improvements in security capabilities through products such as RACF, by its ability to secure the consoles, and MVS commands. JES was enabled to use security products such as RACF, and security propagation from one address space to another was made possible. Many other improvements were introduced to the security products during this period. CICS also started using security products such as RACF for transaction security.

⁴ Stewart Alsop in *Info World*, March 15, 1996

Growth of smaller decentralized computing systems was on the rise, with multiple logons required, confusing the computing identity landscape.

Security landscape and threats

Newer releases of security products such as RACF promised greater security when functioning together with the modern MVS OSs of the time. Support for security labels and multi-level security were made possible.

Large organizations demanded identity programs for their staff, which proved more difficult in the face of a growing number of computing systems and platforms.

3.1.7 Late 1990s

With the IBM System/390® (IBM S/390), IBM reinvented the strength of mainframe from the inside by infusing it with an entirely new technology core, reducing its price, and building support for open standards and operating environments such as Linux. IBM dropped bipolar processing technology in favor of cheaper CMOS-based⁵ processors. These processors rapidly progressed from Generation-3 to Generation-6, with large performance gains. Sysplex capabilities were available to provide clustering for MVS systems.

Hardware encryption capabilities were introduced into processors. The IBM Integrated Cryptographic Service Facility (ICSF) was delivered to manage the encryption functions and to provide Common Cryptographic Architecture (CCA) based application programming interfaces (APIs).⁶

MVS morphed into IBM OS/390®, which solved some of the software licensing problems that were faced by many organizations. Sysplexes grew in use, providing better reliability, availability, and serviceability (RAS) characteristics. CICS and DB2 made extensive use of sysplex functions.

Extensive research was going on to port Linux to IBM mainframes. SUSE began working with IBM, primarily at IBM Boeblingen Lab in Germany and Marist College in 1999.

Security landscape and threats

Hierarchical file systems became more standard as products started to use the UNIX capabilities in OS/390. WebSphere products, which are based on Java and inter-networking, made extensive use of the UNIX environment. IBM introduced security services as an offering.

Distributed systems and desktop systems started to merge into the computing landscape, as organizations standardized their choice of platform.

Growth of e-business models and the security threats over the Internet led to contradictory choices that were available for many organizations in promoting their business.

3.1.8 Early 2000s

One of the main drivers of mainframe acceptance and growth during this time was the proliferation in network computing and e-business users and applications. To help customers better harness the power of this pervasive medium, IBM unveiled, in October 2000, the IBM eServer™ zSeries 900, the first mainframe that was built from scratch with e-business as its primary function.

⁵ Complementary Metal-oxide Silicon

⁶ Some cryptographic capabilities were introduced as early as 1990. However, they were atypical.

zSeries processors took a large leap in processing capabilities by increasing the number of faster engines. PR/SM capabilities continued to grow. Hardware encryption capabilities became more pervasive.

OS/390 morphed once more into IBM z/OS. z/OS had 64-bit capability and ran on the zSeries architecture. Potential virtual memory sizes were immense, with up to 16 EBs.

There was an explosion in TCP/IP growth that overwhelmed other networking protocols. Many organizations withdrew from using SNA except on internal networks. Firewalls were used extensively to protect companies from the hosts of malware on the Internet.

The reinvented mainframe handled the unpredictable demands of e-business, allowing thousands of servers to operate within one machine. The first in a new class of e-business servers, the z900, which worked hand-in-hand with z/OS (the z900 flagship OS), was designed for high-speed connectivity to the network and to data storage systems, scalability in the face of unpredictable spikes in workload or traffic, and near zero downtime when clustered. The z900 allowed customers to push performance and connectivity to the outer limits without any concessions to reliability and security.

IBM and SUSE together debuted the official release of Linux on System z in 2000. Within a short period, the wave of enthusiasm for Linux on System z that came with the ability to consolidate thousands of distributed servers led to heightened interest in VM/ESA in the beginning of the new millennium. IBM z/Architecture® support was added to VM/ESA and it was renamed z/VM. The first 64-bit based z/VM was generally available in 2001.

Indeed, that same *InfoWorld* analyst, who in 1991 predicted the death of the mainframe, wrote in February 2002, “*It’s clear that corporate customers still like to have centrally controlled, very predictable, reliable computing systems -- exactly the kind of systems that IBM specializes in.*”⁷

Security landscape and threats

Mainframe security products started to become the core Identity Management solution for some of the organizations. They also acted to fit in with existing solutions in other organizations.

Products such as RACF could work with multiple sysplexes and propagate updates between different sysplexes. They also worked with digital certificates, which became prevalent on the Internet as a mechanism for Identity Management, and a mechanism for client-server authentication using encryption protocols such as Secure Sockets Layer (SSL).

Complexity was added by a multitude of security standards and government regulations on data privacy and integrity. Concepts of availability and business continuity gained critical importance and were often debated over security and performance.

3.1.9 Late 2000s

At that time, zSeries became the IBM System z. Mainframe hardware changed from the z900 to IBM z990, and to IBM z9®, and later to IBM z10™. These systems grew in power and capabilities over each level of advancement. Hardware cryptographic capabilities increased and were used by financial organizations for PIN management, ATM management, and other financial transactions.

Cryptography was integrated into tape and disk systems with centralized key lifecycle management systems.

⁷ Stewart Alsop in *Info World*, February 2002

TCP/IP was everywhere, with a need for organizations to move to IPv6. SNA shrank in use, but applications that were still SNA-dependant used TCP/IP to encapsulate and transmit SNA data over an IP network.

With more stronger and stable commercial releases of Linux on System z made available, it also became a consolidation platform in many organizations.

Security landscape and threats

Public views of security became a major issue for organizations in their choice of security policies, as multiple data loss incidents occurred. *Encryption* became a concept of necessity and importance.

Centralized identity and Access Management became a reality in most large organizations. Methods and processes governing those systems were still in need of work. Most of the organizations had security policies mapping all their IT systems and data.

3.1.10 The mainframe today: 2010 onwards

The mainframe continued to surpass its own power and strengths with the latest hardware models, OS enhancements, and software functions. With the introduction of IBM zEnterprise System (zEnterprise), which replaced the System z machines, the mainframe platform once again promised to offer a proven hybrid computing design that can help organizations manage and integrate workloads on multiple architectures with the simplicity of a single system. Organizations can also drive intelligent insights, in near real time, to improve business performance and reduce risk. use of the zEnterprise IBM BladeCenter Extension (zBX) allowed integration of IBM POWER7® and System x blades, enabling the distributed operating platforms to use mainframe execution speeds and power.

The zEnterprise EC12 system, which succeeded the zEnterprise 196 and the zEnterprise 114, delivered new levels of performance and capacity for large-scale consolidation and growth, support for next generation of digital signature security support, cutting edge pattern recognition analytics for smart monitoring of system health, and proven hybrid computing design for mainframe and distributed workloads. The zEnterprise EC12 was powered by about 120 of the world's most powerful microprocessors running at 5.5 GHz, capable of running more than 78000 millions of instructions per second (MIPS).

Cryptography was built in to the zEnterprise hardware, providing exceptional performance and advanced functions using Crypto Express4S cryptographic coprocessors and accelerators that are individually specialized to address various encryption needs.

The new version of OS, z/OS Version 2 Release 1, marked a new era of z/OS. It set the groundwork for the next tier of mainframe computing, enabling you to pursue the innovation to drive highly scalable workloads, including private clouds, support for mobile and social applications, and more. Its unrivaled security infrastructure was designed to help secure vast amounts of data. Its highly optimized availability helped to deliver new data analytics solutions, and its continued improvements in management were targeted to help automate the operations of zEnterprise systems.

z/OS V2.1 can help deploy the mainframe as a secured enterprise service delivery hub and as an enterprise cryptographic hub. The new crypto-as-a-service function that was designed for Linux clients is intended to make z/OS-based secure key encryption accessible to Linux applications while providing hardware protection for keys. The security fabric of z/OS improves audit readiness, helps secure data and IP, and supports industry requirement standards.

On IBM zEC12 or zBC12 systems, z/OS V2.1 can deliver 100-way symmetric multiprocessing (SMP) support in a single LPAR and can support 2 GB pages. z/OS design supports an architectural limit of 4 TB of real memory per LPAR to reduce memory management impact and improve overall system performance by enabling middleware to use 2 GB pages.

z/VM Version 6 Release 3 was released in 2013. It now supports hypervisor partitions of up to 1 TB for Linux on System z workload. In addition, the Single System Image (SSI) feature was introduced in z/VM 6.2, allowing for greater horizontal scalability of workload.

3.1.11 Strengths of the mainframe

This section describes the role and characteristics of the IBM mainframe in use today. It is the quality of service that the mainframe provides that makes it such an attractive proposition for many organizations. You will see how a typical mainframe environment is managed and how its strengths contribute to its long-lasting confidence. To begin, let us look at the personnel and roles in a typical mainframe installation.

Personnel and roles

Unlike certain other computer platforms, the administration of the IBM mainframe is frequently distributed. This has advantages, as more rigor is required when planning cooperative events. Table 3-1 describes the typical roles that are prevalent.

Table 3-1 Typical roles and their responsibilities

Role	Responsibilities
Hardware planning	Installation for new hardware, maintenance of exiting hardware, and positioning of processors. Hardware includes the mainframe processor complex, storage devices, and other peripheral devices.
Systems programming	OSs software installation, maintenance, debugging of software problems, and configuration of LPARs and OSs. This role also includes ISV/OEM software installation and maintenance in smaller organizations.
Database management	Database software installation, maintenance, debugging, and housekeeping. There might be different groups for physical and logical database management in some organizations.
Middleware management	Middleware software installation, maintenance, debugging of software problems, and configuration. Middleware products include transaction servers, messaging products, application servers, and web-enablement. There might be separate groups for each of these items in large installations.
Security administration	Maintenance of Identity Management, security policy and implementation, and access controls.
Storage administration	Provision of storage and data availability on both DASD and tape-based storage, and configuration and maintenance of storage policies and rules.
Capacity planning	Performance monitoring and analysis, capacity planning and monitoring, and prediction of capacity growth.
System operator	Day-to-day, hour-to-hour operation of the system, performing routine system management tasks, and incident/alert monitoring and reporting.

Role	Responsibilities
Job Scheduling	Manages the schedules of batch operations, monitors the job failures, and resolves dependencies. This role might be performed by System operators in some small environments.
Change management	Management of problems and changes in the entire System z environment. Plans and coordinates changes across the platform to minimize the impact.

Note: The list in Table 3-1 is not expected to be treated as exhaustive or to define what is necessary in every installation. Each installation has its own set of standards and responsibilities. This list is merely an example.

Perhaps the most important of these roles is *change management*. System z environments frequently operate within a strict change management process. It is in the nature of the disciplines that have grown up around the mainframe that makes change management a serious role. The mainframe is often the security hub and the central data server in many large organizations, and the impact of downtime can be unacceptable.

Role of the modern mainframe

In today's environment, the mainframe is likely to be at the center of the operation of large organizations. It can fulfill a multitude of roles:

- ▶ As a central data server hosting databases of enormous size and providing back-end data processing for data requests. For example, DB2, which is one of the most successful database systems, when running on z/OS, can provide high availability, high throughput, and high performance. These capabilities are enhanced when running in a z/OS sysplex. Mainframes are now versatile, as DB2 also runs on Linux on System z.
- ▶ As a central management point for identity by hosting an Identity Management platform, but also providing distributed access through LDAP to other servers and platforms for authentication. For example, RACF can be used as the central management point for Identity Management within an enterprise. It can store information that relates to multiple other systems by using the custom segments facility. LDAP capabilities allow RACF to be used for authentication over secured network links. RACF also can be used for central security access control by using a pluggable authentication module (PAM) for Linux. A single user can have multiple identities on multiple Linux servers.
- ▶ As a central network policy server for enterprise network management. For example, the IBM z/OS Communications Server policy agent provides this capability. Centralized or distributed policy services are provided over connections between a server and clients that are secured by the Application Transparent - Transport Layer Security (AT-TLS) feature. The distributed policy services provide a security mechanism that is based upon client login passwords or pass tickets. The mainframe's ability to host IP security rules and a firewall within the mainframe platform has extended its role to be a centralized network management platform.
- ▶ As a host for web services, the mainframe can provide access to heritage services. For example, SOA services within IBM WebSphere products can be used to provide access to services developed over previous years that encapsulate business rules in various forms, such as COBOL routines, CICS transactions, and so on. Most financial organizations use a mainframe as their middleware and messaging hub that consolidates the messages from various platforms and performs business processing for their varied transactions.

- ▶ As a platform for high-volume transaction processing. For example, the mainframe is still a large host for volume transactions that are frequently run through CICS, IMS, or any other third-party transaction systems. Typically for an organization, their online day might commence in the early hours of the morning and finish in the early evening when the available processing capacity is turned over to batch processing.
- ▶ As a hub of data processing through batch. The mainframe still holds its place as the main platform for large-scale batch processing. The capability to process large volumes of data at a high speed is one of the mainframe strengths that has never changed over years. Many organizations have significant volumes of batch processing during their off-shift hours, when online transactions are at a low ebb. It is this ability to be fully used over the entire 24-hour day that makes the mainframe computing model so efficient and successful.

Thus, it is evident that the present day mainframe serves as a strong centralized platform for large organizations in managing their information technology components and functions.

Key characteristics of the mainframe

It is imperative to realize some of the key concepts that makes the mainframe so special. As the saying goes, the roads of the information super-highway often leads to a mainframe!

- ▶ *RAS*: The RAS of a computer system are important factors in data processing. When a particular computer system “exhibits RAS characteristics”, which means that its design places a high priority on the system remaining in service always. Ideally, RAS is a central design feature of all aspects of a computer system, including the applications. RAS has been a unique selling point for the mainframe over decades.
- ▶ *Security*: One of a firm's most valuable resources is its data, such as customer lists, accounting data, and employee information. This critical data must be securely managed and controlled and simultaneously made available to those users that are authorized to see it. The mainframe computer has extensive capabilities to simultaneously share and protect the firm's data among multiple users.
- ▶ *Scalability*: In business, positive results can often trigger a growth in IT infrastructure to cope with increased demand. The degree to which the IT organization can add capacity without disruption to normal business processes or without incurring excessive impact is largely determined by the scalability of the particular computing platform. Scalability of the mainframe platform is incomparable to any other computing platform that is available.
- ▶ *Continuing compatibility*: Mainframe customers tend to have a large financial investment in their applications and data. Some applications were developed and refined over decades. Some applications were written many years ago, and others might have been written recently. The ability of an application to work in the system or its ability to work with other devices or programs is called *compatibility*. Few organizations have active mainframe applications that are untouched or with few modifications, surviving over different architectures and advancements of mainframe and still using benefits such as performance improvements.

Software maintenance philosophy

Mainframes are remarkable in terms of their availability at all levels of the mainframe architecture, including both hardware and software. There always is a compromise between the system availability versus the maintenance. The mainframe maintains this balance through its maintenance philosophies and mechanisms.

z/OS systems are complex in terms of the relationships between software elements and components. Thus, the z/OS has a sophisticated mechanism for managing the updating of software elements, such as during an installation, maintenance, or fall-back. This mechanism handles the complex dependencies between all the installed software components and their elements over the entire lifecycle and ensures strong compatibility and balance between them.

This mechanism also ensures that the software fixes that are developed by the product vendor or home-grown customization within an organization is managed within its structure. Thus, any change to the particular software or its element is tracked and resolved for dependencies to avoid any conflicts during run time. The technical details of the actual components, such as Sysplex and SMP/E, that deliver this mechanism, are explained later in this book.

Although providing fixes to the problem, making them easily available, and ensuring that they are reliably installed is important, it is also important that problems that are discovered are resolved early, and that the knowledge that is gained in their analysis is captured and retained. Since the 1970s, IBM maintains a large database that is called IBM RETAIN®. RETAIN holds information about problems that are reported on mainframe systems going back to the 1970s. This database enables service personnel to track problems, understand use, and view development issues over all the incarnations of the various OSs. It is an invaluable source of information for both problem-solvers and developers.

Change control and continuous availability

Because of the sheer size of the operations of many mainframe installations, the role of change control is a major factor in the running of the mainframe. Change control introduces disciplines that provide their own benefit.

z/OS and z/VM in particular lend themselves to the type of controlled operation that can be change managed. Backout capabilities for changes are built into the structure of the configuration files.

However, also built into the System z software and hardware are a multitude of capabilities to make changes in a nondisruptive manner. Frequently these capabilities require planning in the configuration of processors, LPARs, and applications, but the aim is to enable applications to continue running while changes are made.

As applications can be configured to run on z/OS sysplexes that straddle LPARs and processors, and can be configured to allow work to flow freely among those LPARs, it becomes possible to remove LPARs and processors from sysplex configurations and still allow applications to continue processing.

It is this property of continuous availability while change continues that characterizes System z and distinguishes it from other computing platforms. Here are examples of the continuous availability concurrent change features:

- ▶ Ability to add or remove hardware in flight. System z makes advances in this area at each new hardware announcement. Even processor books containing processors and memory can be added and removed while other work continues on the same System z mainframe.
- ▶ LPAR and PR/SM reconfigurations can be made by dynamic reconfigurations while some LPARs remain active. Reconfigurations can include changing channel path definitions and device definitions.
- ▶ z/OS Sysplex reconfiguration allows moving of coupling facility structures while applications continue. This feature allows the movement of CFs to new processors.

- ▶ z/OS Sysplex reconfigurations allow the seamless movement of work from one LPAR to the remaining members of the sysplex.
- ▶ z/OS allows the dynamic addition and removal of system libraries and system exits and control points. This feature allows new releases of applications to be installed while the OS remains active, even if newauthorized program facility (APF) and other system libraries are required.
- ▶ z/VM recognizes new hardware in-flight and can use it for new and existing guests.
- ▶ z/VM SSI allows for the mobility of Linux guests from one z/VM LPAR to another LPAR (even on another CPC) without an interruption in workload processing. This feature allows for servers to continue even during planned outages of a z/VM LPAR.

With this type of capability, it is not practical to simply assume that these things are always possible. Careful planning is always required, and planning for future continuous operations and availability in disaster situations is not optional. Nevertheless, it is feasible for applications to stay active for years without outage, and during that time, the following actions can occur:

- ▶ Introduce new storage and move databases.
- ▶ Decommission old storage.
- ▶ Introduce new processors and move processing from LPARs on existing processors.
- ▶ Replace OS versions in a staggered fashion.
- ▶ Replace applications and middleware in a staggered fashion.

It is perfectly feasible to have a user application running on a *z/OS sysplex on a particular set* of System z hardware and then to implement a series of changes to new processing hardware, new DASD storage, a new OS level, a new middleware version, and even a new user application version, and achieve all of the above without any outage. At the end of the exercise, the entire hardware and software is replaced, but the application has suffered no outage.

3.2 Workloads on the mainframe

Traditionally, only two types of workloads are used in mainframes: the *online* and the *batch* workloads. Online workload refers to the transactional workload that run in real time, mostly initiated by the users. Batch workloads are the scheduled running of programs with defined input using batch jobs with minimal or no human interaction. Online transactions run core functions of mission-critical applications and take less than few seconds or minutes. Batch programs are mostly scheduled during off-shift hours to run long-running business processes, such as reporting.

3.2.1 Batch processing

Batch applications are those applications that are processed on the mainframe without user interaction. A batch job is submitted on the computer, the job reads and processes data in bulk, and produces output, such as customer billing statements. An equivalent concept can be found in a UNIX script file or a Windows command file, but a mainframe (or z/OS in this specific case) batch job might process millions of records and are much ore efficient than UNIX or Windows processing.

Although batch processing is possible on distributed systems, it is not as common as it is on mainframes because distributed systems often lack the following features:

- ▶ Sufficient data storage
- ▶ Available processor capacity, or cycles
- ▶ Sysplex-wide management of system resources and job scheduling

Mainframe OSs are typically equipped with sophisticated job scheduling software that allows data center staff to submit, manage, and track the running and output of batch jobs.

Batch processes typically have the following characteristics:

- ▶ Large amounts of input data are processed and stored (perhaps terabytes or more), large numbers of records are accessed, and a large volume of output is produced.
- ▶ Immediate response time is not a requirement. However, batch jobs often must complete within a batch window, which is a period of less-intensive online activity, as prescribed by a service level agreement (SLA).
- ▶ Information is generated about many users or data entities (for example, customer orders or a retailer's stock on hand).
- ▶ A scheduled batch process can consist of the running of hundreds or thousands of jobs in a pre-established sequence.

During batch processing, multiple types of work can be generated. Consolidated information, such as profitability of investment funds, scheduled database backups, processing of daily orders, and updating of inventories, are common examples.

3.2.2 Online transaction processing

Transaction processing that occurs interactively with the user is referred to as *online transaction processing* (OLTP).

One of the main characteristics of a transaction system is that the interactions between the user and the system are short. The user performs a complete business transaction through short interactions, with an immediate response time that is required for each interaction. These systems are supporting mission-critical applications, so continuous availability, high performance, and data protection and integrity are required.

Here are some industry uses of mainframe-based online systems:

- ▶ Banks: ATMs and teller systems for customer service
- ▶ Insurance: Agent systems for policy management and claims processing
- ▶ Travel and transport: Airline reservation systems
- ▶ Manufacturing: Inventory control and production scheduling
- ▶ Government: Tax processing, and license issuance and management

Online transactions often have the following characteristics:

- ▶ A small amount of input data, a few stored records that are accessed and processed, and a small amount of data as output
- ▶ Immediate response time, less than one second
- ▶ Many users who are involved in many transactions
- ▶ Round-the-clock availability of the transactional interface to the user
- ▶ Assurance of security for transactions and user data

3.2.3 Modern day classification

Although this classification is still good for many purposes today, mainframe workloads also can be classified based on the nature of the request and the application. The mainframe's ability to run mixed workloads provides an in-depth classification of workloads that run on it. This classification provides a better view of different requests that are handled by a modern-day mainframe processor. Although some of these requests also are applicable to other platforms, a few of them are more specific to mainframe environments only.

Operating system workloads

The architecture of the mainframe enables different OSs to operate in parallel through its PR/SM technology on the same processor. Because the mainframe processors can host varied operating platforms, a workload classification can be made based on the corresponding OSs. Other than the traditionally aligned z/OS and z/VM OSs, the mainframe also can run OSs, such as Linux, and distributed OSs through its blade extension capabilities.

z/OS is the most common OS that runs on the mainframe. With its long history (it was used in an earlier form in the first mainframe systems in 1960s), this OS has been enriched to run efficiently on the mainframe architecture, and is a robust platform for hosting an organization's critical applications.

z/VM, the second most common OS that runs on the mainframe, is widely used in most organizations as a virtualization host. This function gained importance in recent years because of the growth of server consolidation from distributed to mainframe platforms through Linux on System z. There are a few organizations that host their applications directly on z/VM. There are few organizations that host z/OS guests on z/VM for flexibility and dynamism, making the application of z/VM inevitable.

Linux, which is an open source platform, became available for the mainframe in 1999. The mainframe's virtualization technology allows thousands of Linux images to operate simultaneously as isolated systems. Multiple images of Linux on the mainframe tend to constitute most of the processor use in a Linux based environment.

The latest mainframe hardware can host blade servers, such as the IBM POWER7 blades or the IBM System x blades. This capability means that the mainframe can process the distributed workloads for these servers. Although the application workloads on these platforms are processed by the corresponding processors, the mainframe still manages the workload on these platforms or processors through its unified resource manager component.

Lastly, mainframe processors host OSs such as z/VSE and z/TPF. IBM continues to support these OSs on the mainframe to enable those organizations using these OSs to continue with their business processing.

Execution workloads

As with OS workloads, we classify the workloads based on some common factors that are not specific to any OS. z/OS, being the most commonly hosted operating platform on the mainframe, has some specific classifications within this list of workloads that run on the mainframe processor. Some of these workloads operate on *special* processors, which reduce the load on the general processor that runs all other instructions. These special workloads are known as *execution workloads* because the classification of these workloads is based on the nature of the workload and its source.

Here are the types of execution workloads:

- ▶ *I/O requests*: Accessing the data that is stored on disk or tape is the most frequent and critical part of any application or task execution. I/O data requests constitute a major part of instruction execution in any processing environment. Because mainframes are the data hub of many large businesses, I/O processing takes precedence in the consideration for specialty processor designs.
- ▶ *Database*: The mainframe is known for its huge data processing capabilities. Even the modern applications that have their web processing and business logic processing on non-mainframe platforms have large mainframe servers for their data processing components. This fact is more than enough to justify the workload of data processing instructions on mainframe servers. Data processing can be classified as access to local data and of remote data in a distributed environment.
- ▶ *Transaction processing*: Most of the mainframe applications are based on transactional processing. Transactions are started and run on Transaction Managers to run a predefined unit of work that includes database access, business logic processing, and messaging. Because most of the world's banking transactions run on the mainframe, the transactional workload demands for critical processing requirements are tied to availability and response time. This requirement holds true for transaction processing of other business critical non-banking applications as well.
- ▶ *Java*: The emergence of Java as a programming language created a major breakthrough and greatly transformed the use of the Internet and applications. Open MVS technology on the mainframe supported the running of Java on the mainframe platform before Linux. The mainframe became the gateway to the external world, which led to end-to-end business processing on a single box. Thus, Java contributes to a measurable share of workload processing in mainframe processing.
- ▶ *Security*: Security on the mainframe is stronger than any other platforms. This security is one of the unique selling points of the mainframe, which encourages customers to port their applications to the mainframe, especially in the banking and financial industries where security is their first priority. Security on the mainframe is handled through the System Authorization Facility (SAF) system component, which interfaces between a security product, such as RACF, and various resource managers that run on the mainframe. Every resource access initiates a SAF request to RACF, which is responded to with an approval or rejection. With enormous processing concurrently happening on the mainframe, which holds many resources, the workloads of these security requests account for many of the processing demands.
- ▶ *Web*: Today's world has gone mobile, and the Internet dominates business. Such a transformation of consumer behavior has demanded that applications be enabled end-to-end on the Internet and be accessed through mobile computing devices. These web requests are similar to transactional calls, but have proportionately bigger numbers that are run simultaneously. This demand is forecasted to increase, so a special focus on processing requirements is needed to cope with the demand.
- ▶ *System monitoring and recording*: The recording of system events in the mainframe is performed at a minute level through a continuous monitoring mechanism. Every activity of the system is monitored and measured continuously for various requirements, such as workload balancing, accounting, debugging and auditing. The System Management Facility (SMF) is the key component that provides full instrumentation of all baseline activities running on the mainframe OS, including I/O, network activity, software use, error conditions, processor use, and so on. Uninterrupted processing is required to perform effective monitoring and recording of system events to safeguard against any system failures and to provide critical data for auditing and debugging.

- ▶ *System event and performance reporting:* Performance and capacity management of large servers such as the mainframe demand detailed analysis of system use reports, including various hardware and software components. Such reports are generated from real-time and historical reporting mechanisms that are managed by system components such as IBM Resource Measurement Facility™ (IBM RMF™). Such reports also are used for sizing decisions and yearly software licensing. Serious performance tuning recommendations are also gathered from such reports, making the reporting component more critical in business.
- ▶ *System command processing:* One of the key strengths of mainframe is its flexibility to interact with the hardware or the OS through various system and operator commands. It is possible to skip the running of an instruction on the processor through such commands. Thus, command processing is critical and requires special processing requirements to provide a seamless interface to the hardware or OS for the systems administrator.
- ▶ *Paging and swapping:* The core concept of the mainframe OS is its virtualization. The mainframe OS uses sophisticated paging algorithms to effectively manage virtual storage. Many system components, both hardware and software, contribute to the handling of the movement of pages and several additional tables to track the most current version of each page. Efficient paging requires uninterrupted processing of the algorithmic instructions.
- ▶ *Workload management:* The mainframe's processing and virtualization is efficient because of its effective workload management. The Workload Manager (WLM) component of the mainframe OS translates the business goals into processing execution to best use of the mainframe's resources, maintain the highest possible throughput, and achieve the best possible system responsiveness. In addition, the Intelligent Resource Director gives WLM the ability to manage resources across the entire cluster of system images in a clustered sysplex environment. Thus, WLM demands critical processing requirements so that it can manage rest of the mainframe workload effectively.
- ▶ *System recovery:* The mainframe is known for its self-healing mechanisms if there is a component or system failure, and it provides continuous availability and reliability to its users. Components such as Automatic Restart Manager (ARM) and Sysplex Failure Manager (SFM) ensure that any system failures are handled appropriately according to the defined rules. The goals of failure management in a sysplex are to minimize the impact that a failing system might have on the sysplex workload so that work can continue with little or no operator intervention. The SFM is integrated into the OS base. All installations should take advantage of SFM and implement it when they are configured in either a basic or an IBM Parallel Sysplex®. ARM can, without operator intervention, restart a job or task that failed, even on a different system if there is a total system failure. These components are critical for system execution and needs on-demand processing whenever necessary.
- ▶ *Network and data communication:* As the backbone of any data center design, the mainframe serves thousands of user requests every second. Mainframes were communicating to the external world using the IBM proprietary SNA protocol until the mid-1990s, when TCP/IP was began to be supported by mainframe, which allowed the mainframe to participate in the Internet. Through various advancements, network communication on mainframe has had tremendous improvements in speed, security, and quality of service. As the world shrinks because of developments in public and private network infrastructure, there is demand to retain quality and speed. Such a demand has proportionally increased the network load on the mainframe servers, as they form the processing and data hub of most IT environments.

- ▶ *Coupling facility*: Parallel Sysplex, introduced in 1994, is a resource sharing technology that has had remarkable advancement over the years. Applications interface with coupling facility structures by using a special instruction set called Cross-system Execution Services (XES). These instructions run in a resource sharing environment involving multiple logical partitions (LPARs) and form a substantial amount of processing in complex clustered or distributed environments. The present-day demand for data and application availability has led to a major shift in focus on to this technology.
- ▶ *IBM Geographically Dispersed Parallel Sysplex™ (IBM GDPS®)*: This sophisticated data mirroring technology involves the processing of special instructions to coordinate the consistency points of the application data. The cyclical process repeats every few seconds to maintain the recovery point objective (RPO) at the lowest possible value. This leads to a considerable workload of instructions on the processor, which is critical to run without delay to achieve the recovery objectives.
- ▶ *Cryptography*: The mainframe has the latest security mechanisms to deliver a safe and flexible operating environment for the users. Cryptography, one of the main components of security, is incorporated within the mainframe architecture to provide encryption and decryption services. Powerful encryption or decryption techniques involve extensive processing to provide fool-proof and secure data that cannot be hacked easily during storage or transmission. A specific set of complex instructions that perform this task are required to run with less delay to provide maximum performance along with strong security. Such extensive processing needs a special processor with unique capabilities to handle the cryptographic requirements.
- ▶ *TSO and ISPF*: The most common user interface to the mainframe is the Time Sharing Option (TSO). TSO allows users to log on to the mainframe through emulator sessions and navigate the OS through the TSO command interface. Users also use Interactive System Productivity Facility (ISPF) to easily navigate and issue commands through a panel-based interface within TSO. TSO and ISPF use the mainframe's virtual address space concepts to support hundreds of users to use the system resources simultaneously. This translates to many processor requests from these workloads in a development environment with more batch users compared with production systems, which are used for business processing.
- ▶ *IBM Language Environment® and JCL*: As the technology on the mainframe continues to advance, many applications are developed to support the increasing demands of the customers. The mainframe's Language Environment includes a set of compilers and linkage editors that are employed in developing application components. A development system has a continuous activity from such compilers to build and maintain applications. Similarly, another commonly used language on mainframe is the JCL. Almost any of the mainframe tasks needs a JCL to perform. Parsing JCL statements is done by a special parser within the OS. Heavily batch-oriented systems rely on a JCL parsing workload. Thus, compilers and JCL parsers assume the major share of the processing workload in development systems.
- ▶ *Extensible Markup Language (XML)*: Similar to Java, XML played its own part in the transformation of applications' accessibility over the Internet. The dynamics and flexibility of XML is combined with the strengths of the mainframe to deliver critical services to Internet clients. Many COBOL applications were opened to the Internet by converting them to XML, leading to substantial workload of XML processing on mainframe.

With all the above classification of workloads, the processor still services an enormous workload of general requests from various components of the OS and the applications. These workloads are served by the general-purpose processors and are still important in processing unclassified workloads that are based on the requirements.

3.3 Virtualization

Virtualization refers to a set of techniques whose purpose is to deliver an abstract view of computer resources, which is sometimes called an *image* of a computing environment. The virtualized computing environment is provided with virtual resources, behaving, from the users' standpoint, as though they were real ones. Virtual resources can be fully simulated by the virtualization mechanisms or be backed by their real counterparts. Programs can be run in virtualized environments, as mechanisms are in place to proceed with an initial load of a program (IPL) and start it.

Most virtualization implementations allow the creation of multiple virtualized environments in a single physical system that, by sharing physical resources through their virtualized images, lead to an optimized use of these physical resources. Optimizing the use of physical resources is one of the prevalent benefits users find in using virtualization in a commercial environment.

Today, techniques are available to apply virtualization to any physical entity that is part of an IT logical infrastructure, such as disk storage or communication links, with a scale ranging from virtualizing a set of installations as a *cloud* down to the creation of multiple virtualized computing environments on a PC.

Virtualization and simulation

Virtualization can be fully achieved by simulation only, as is the case when the aim of virtualization is to provide an environment that is architecturally different from the host system. However, virtualization in a business context performs more efficiently when the virtualized environments are conforming to the native architecture of the host system. In this latter case, the host hardware can be directly used, under the control of the virtualization mechanisms, by the guest programs without going through a simulation layer. However, even that situation, a simulation is required when a host program, for example, modifies the environment, as this modification affects the virtualized environment and not the real system.

The System z implementation of virtualization, either with z/VM or PR/SM, is a good example of this latter approach, as it provides zArchitecture compliant virtualized environments. Virtual CPUs are backed by real processing units (PUs) in the machine, so that the instructions flows issued from the guest programs in the virtualized environments are dispatched for running on real PUs. This avoids as much as possible the need to provide simulation of instructions and ensures maximum performance for guest program execution.

Virtualization and time slicing

Time slicing is the base mechanism that is used with virtualization when it comes to sharing physical resources between several virtualized environments. As an example, real PU run time can be allocated on a time slice basis so that different virtualized environments can share the same real CPU.

Time slicing in the virtualization context also has issues of its own:

- ▶ The time-slicing mechanism should accommodate for asynchronous events that might, for example, require immediately swapping the current virtual environment execution flow with an instruction flow put on hold waiting for this event.
- ▶ The dispatching mechanism, for security reasons, also should ensure that residual data is cleared before changing the allocation of a shared real resource.
- ▶ The dispatching mechanism also must ensure the integrity of the state information that is saved and restored when execution contexts are switched to active or inactive.

In many implementations of virtualization, the time slicing and dispatching algorithms are refined to the point that users can assign relative performance goals to the virtualized environments, as it is the case with System z PR/SM or z/VM.

Where there is a strong requirement to achieve close to native performance, an implementation of virtualization can offer the capability of dedicating real resources to a specific virtualized environment (that is, resources that are not to be shared with other environments). This eliminates most of the time-slicing and dispatching impact. This capability is also offered by System z PR/SM and z/VM.

3.3.1 Virtualization on System z

This section describes virtualization techniques and mechanisms in the IBM System z mainframe. Virtualization can be implemented as purely software functions that are dedicated to providing the environment image to the users, in which case a special OS, usually called a hypervisor, creates and manages the virtualized environments. IBM z/VM is the System z software hypervisor that creates and manages VMs in to which programs can be loaded and run.

Virtualization also can be implemented as a set of hardware and firmware mechanisms, as is the case with the PR/SM facility of System z, which is used to create and manage LPARs. PR/SM is described in “Hardware or software implementation” on page 49. Users can run z/VM inside a PR/SM LPAR, thus achieving nesting of virtualized environments.

Figure 3-1 shows a schematic example of virtualized environments creation in System z. The virtualized environments run z/OS, z/VM, and Linux for System z OSs, with each managing the running of their own applications.

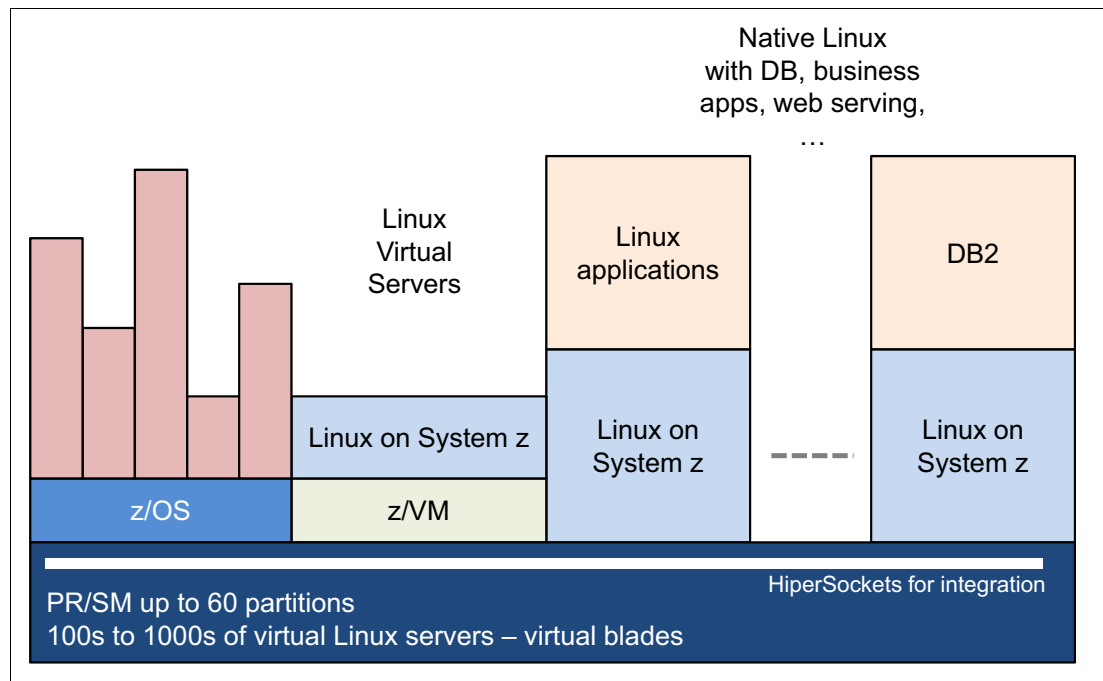


Figure 3-1 Multiple virtualized environments in a System z

It is obvious that whatever the virtualization mechanism is, it brings a certain amount of impact to program running, as it also consumes physical system resources. However, today there is a general recognition of virtualization benefits offsetting the inherent cost of virtualization.

Issues with virtualization: How they are addressed in System z

The virtualization mechanisms can be thought of as a layer of translation for the interactions between the running programs that *see* virtual resources and the real resources that are used by these programs. This translation layer remains transparent for the system's users, except for the set of users in charge of administering the virtualization mechanisms.

Hardware or software implementation

The virtualization mechanisms can be implemented as hardware and firmware functions or as purely software functions. The choice generally is a trade-off between performance and adaptability. It is expected that a hardware implementation of the virtualization layer runs more efficiently, but is less adaptable than a software implementation.

Both implementations are available for the System z in a non-exclusive manner because a software-driven virtualization (the z/VM hypervisor) can be operated on top of the system built-in hardware virtualization (PR/SM) facility.

It is worthwhile to emphasize the wealth of experience that IBM has accumulated over years of implementing virtualization in the mainframe, as z/VM was initially released as VM/370 (VM) in 1972 and PR/SM made available in the IBM mainframes in 1988.

Performance

There is a cost to operate the virtualization layer that ultimately translates into additional processor time consumption and additional management tasks that are incurred by the organization. Then, the virtualization benefits must offset this extra cost.

To achieve optimum virtualization performance, System z uses dedicated hardware mechanisms at the core of its virtualization implementation that are used both by the built-in hardware virtualization (PR/SM) and the software virtualization (z/VM).

Configuration of the virtual environment

Configuring a virtual environment is the operation in which a system administrator specifies which logical resources are available to the programs running in the virtual environment, such as the memory size, the number of PUs, and I/O devices and their access paths. This information usually is kept in a permanent storage area that is used by the virtualization mechanism when it comes to creating the environment or statically or dynamically modifying it.

The configuration of a virtualized environment also has certain security connotations:

- ▶ Only authorized people can assign virtual resources to a virtualized environment in agreement with the installation policy addressing workload allocation and assignment of performance goals. It also is of paramount importance that this allocation of resources does not lead to unexpected sharing of real resources or unexpected communications paths between different virtualized environments.
- ▶ The virtualization mechanism should strictly enforce allocating virtual resources to the proper virtualized environment, according to the system administrator's specifications.

Virtualized environment security features

Virtualization brings its own set of security issues that are related to the operations of the virtualization mechanisms or to their administration:

- ▶ The administrative access to virtualization mechanisms' configuration and controls should be secure.
- ▶ VMs accessing physical resources should be duly authenticated and authorized to do so.
- ▶ VMs should be strictly isolated from interferences by other environments.

Although these concerns look similar to the security concerns found when designing an OS, one must realize that they pertain to the integrity of the virtualized environment. *Integrity* in this case is focusing on the capability of the virtualized environment to strictly react as a real one, and to be isolated fully from interferences or penetrations by the other coexisting virtualized environments.

The IBM approach for mainframe virtualization security is to provide a virtualized environment with as much integrity as though it were a real system standing by itself and leave the guest software to manage its own security. IBM has this capability certified by independent laboratories for the PR/SM facility of each new mainframe model and for new releases of z/VM.

The z/VM or PR/SM certification is performed against the Common Criteria (ISO15408) standards.

3.3.2 Goals and benefits of virtualization

Virtualization has been available for more than three decades in the IBM mainframes (the VM/370 hypervisor was generally available in 1972, preceded in the late 1960s by early implementations of hypervisor concepts such as the IBM System/360 CP-67 OS). Since then, virtualization has been constantly in high demand from users because of the strong business drivers that surfaced through the years that can be efficiently satisfied by virtualizing the computing environment. Although the magnitude of the benefits of virtualization may vary depending on the workload context, the specific virtualization technologies that are used, or the existing IT infrastructure, users can expect to achieve gains in one or many of the following areas:

- ▶ Higher physical resources use

Virtualization enables the dynamic sharing of physical resources and resource pools between several virtualized environments, resulting in higher resource use, especially for variable workloads where the average needs are much less than an entire dedicated resource.

- ▶ Lower management costs

Virtualization can improve staff productivity through the following actions:

- Reducing the number of physical resources that must be managed
- Hiding some of the resource complexity
- Simplifying common management tasks through automation, better information, and centralization
- Enabling workload management automation

Virtualization has its own management costs, which are offset by this productivity improvement.

- ▶ More flexibility

Virtualization enables computing resources to be deployed and reconfigured dynamically to meet changing business needs.
- ▶ Higher availability

Virtualization enables physical resources to be removed, upgraded, or changed without affecting user operations in the virtualized environment.
- ▶ Increased scalability

Resource partitioning and aggregation enable a virtual resource, depending on the product, to be much smaller or much larger than an individual physical resource, meaning that the user can make scale adjustments without changes to the physical resource configuration.
- ▶ Interoperability and investment protection

Virtual resources can provide compatibility with interfaces and protocols that are unavailable in the underlying physical resources. This is increasingly important for supporting existing systems and ensuring compatibility with earlier versions.
- ▶ Improved provisioning

Virtualization can enable resource allocation to use a finer degree of granularity than individual physical units allow.
- ▶ Consolidation

As virtualization enables multiple applications and OSs to be supported in one physical system, it can be used to consolidate servers into VMs on either a scale-up or scale-out architecture. It also enables systems to treat computing resources as a uniform pool that can be allocated to VMs in a controlled manner.

3.4 Overview of mainframe security architecture

The mainframe is said to be the most *securable* platform, but is not necessarily a *secure* platform. Security is not just about the product itself. It is about the organization's security practices, which are related to their processes and people who are part of those processes. The mainframe's reputation for security is because of the philosophy and methodology of "integration by design" on which it is built. In a mainframe, security components are integrated from chip to software.

The security features on a mainframe are deeply integrated into the hardware, within every chip and the processor. These features also expand to the hypervisor, which uses the secure system design in the middleware layer. These features expand to the software layer to create a complete secure entity across multiple points of interception.

This is where the *process* gains importance. The mainframe platform provides a model to secure an organization's data and environment, and it enables them to build simplified processes around its integrated security features.

This section shows the concepts and technology underlying the mainframe architecture, and how its security design helps provide such a philosophy and model.

3.4.1 Hardware and microcode design

The computer architecture of a computing system defines its attributes as seen by the programs that are run in that system, that is, the conceptual structure and functional behavior of the server hardware. System z architecture, also known as z/Architecture, spans such attributes of multiprocessing, virtualization, dispatching, interrupts, storage protection, and so on. This section covers these concepts from the hardware perspective. The design of OS and the security functions within the software components are described in 3.4.2, “Software design” on page 57.

System z is made up of hardware products, including a central processor (CP), software products, and an underlying OS, such as z/OS. Other types of software, such as the system application programs and the user applications, also run on the system. The CP is the functional hardware unit that interprets and processes program instructions. The CP and other system hardware, such as channels and storage, make up a server complex. The System z architecture and functional design provides a highly secure computing environment by preserving the integrity of its operations with a clear isolation of events.

To better explain how some z/Architecture instructions are implemented in System z, we now introduce the concept of *microcode*. The vast majority of the z/Architecture instruction set is implemented through microcode. Microcode is a design option, not an architecture option, that is used to implement CP logic. To make it simple, the CP has two pieces: data control and data flow. Instructions are run in the data flow where data is transformed, but the sequence and timing of each of the multiple operations that are done in the data flow is ordered from the data control. It is similar to an orchestra where musicians, like pieces of data flow, know how to play their instrument, but they need guidance and tempo from the maestro, the data control, to perform properly.

In a microcoded CP, for each possible instruction of the instruction set, there is one micro-program that tells data control what to do to run the instruction in the data flow. The micro-program has, in a special language, the sequence of orders to be sent by the data flow. These micro-programs are loaded in to a special internal memory in the CP called control storage at the time of hardware initialization (*power-on reset*). Decoding an instruction consists of finding the address in the control storage of its micro-program. The opposite of microcoding is *hardwiring*, in which the logic of data control for each instruction is determined by Boolean hardware components. The advantage of microcoding is flexibility, where any correction or new function can be implemented by changing or adding to the existent microcode. It is also possible that the same CP may switch instantaneously from one architecture to another (such as from ESA/390 to z/Architecture) by using another set of microcode to be loaded into a piece of its control storage.

Multiprocessing

Allowing many processes at the same time in a system can cause a single CP to be heavily used. In the 1970s, IBM introduced a *tightly coupled multiprocessing complex*, allowing more than one CP to run more than one process (task) simultaneously. All these CPs share main storage (main storage, central storage, and real storage are different terms for the same memory), which is controlled by a single z/OS copy in such storage.

To implement tightly coupled systems, the following items are needed in the architecture:

- ▶ Shared main storage, which allows several CPs to share main storage
- ▶ CP-to-CP interconnection and signaling
- ▶ Time-of-Day Clock (TOD) synchronization to ensure that all TODs in the same server are synchronized
- ▶ Prefixing implemented by the Prefix Register and related logic

A tightly coupled multiprocessor has more than one CP, and a single MVS image, sharing main storage. The CPs are managed by the single MVS image, which assigns work to them. This provided more power and potentially more availability because you could conceptually continue processing even if one of your CPs failed. These machines were available either as Attached Processors (APs), where only one CP had an I/O subsystem, and multiprocessors (MPs), where each CP had access to its own I/O subsystem. In addition to providing more capacity on an MP, the servers, I/O channels, and storage can be physically “partitioned”, meaning that two separate copies of the OS can be run on the servers.

Logical partitions and dispatching

The next major hardware advance, in terms of flexibility for running multiple copies of the OS, was PR/SM with the LPAR feature. PR/SM provided the ability, even on a server with just one CP, to run up to four LPARs. You can split your production applications across several system images, and have a separate development system, or a test system, all on a server with a single CP. Such a configuration did not provide much protection from CP failures if you had just one CP, but it did help protect against software failures. It also provided the ability to create a test environment at a lower cost, thus enabling you to ensure that all software was tested before your production applications were run on it.

Physical CPs can be dedicated or shared. If dedicated, the physical CP is assigned permanently to a logical CP of just one LPAR. The advantage of this is less LPAR overhead. Although the use of shared CPs does have more of an impact, this impact is nearly always more than offset by the ability to have one LPAR use CP capacity that is not required by another sharing LPAR. Normally, when an OS that is using a shared CP goes into a wait, it releases the physical CPs, which can then be used by another LPAR. One of the significant reasons for the increased number of LPARs is server consolidation, where different workloads spread across many small machines may be consolidated in LPARs of a larger server.

LPAR also continues to be used to run different environments, such as system programmer test, development, quality assurance, and production, in the same server. This context is shown in Figure 3-2.

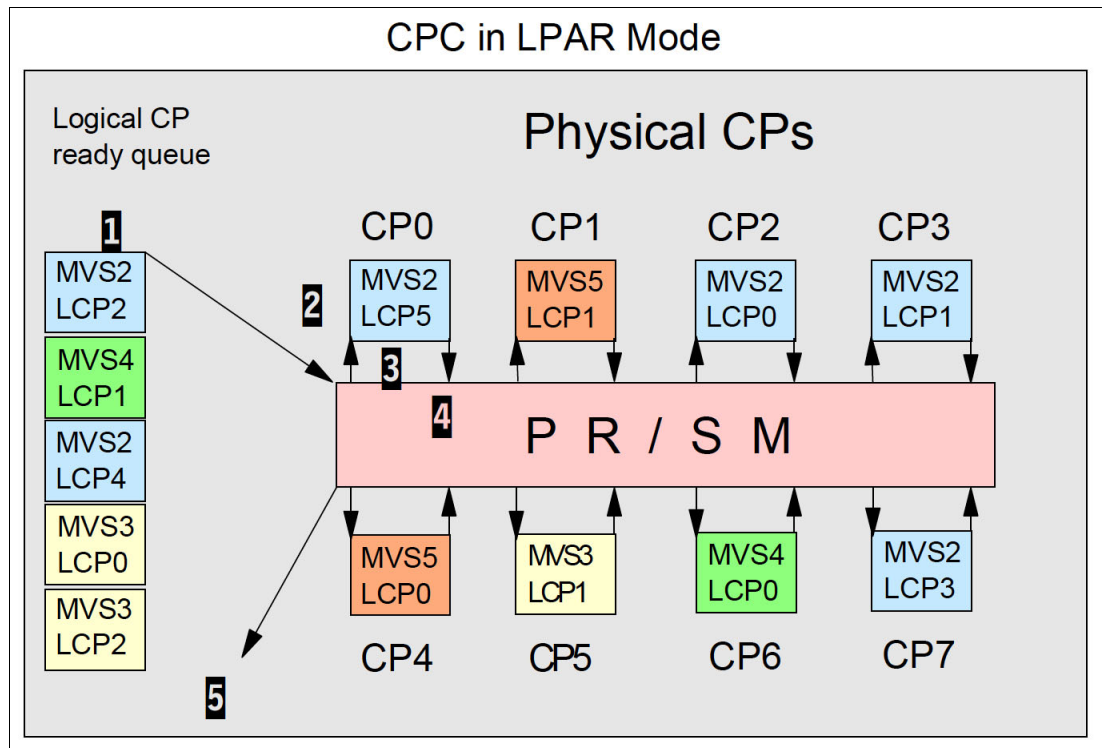


Figure 3-2 LPAR logical CP dispatching

The code that provides the LPAR dispatching function, associating a physical processor with a logical CP, is called LPAR Scheduler Licensed Internal Code (LIC). LPAR LIC logic runs on all of the physical CPs. LPAR LIC dispatches a logical CP on a physical CP by issuing the Start Interpretive Execution (SIE) instruction, with the logical CP represented by a state description control block (SDCB) as a parameter. There is one SDCB per each logical CP. This causes the OS code or application code in the LPAR to run on the physical CP through the logical CP. The logical CP is dispatched on the physical CP by copying the LPAR's logical CP status from the specific SDCB to the corresponding actual entities. When the logical CP is intercepted, its status is saved in SDCB and the LPAR LIC is dispatched automatically on the physical CP again. This code then chooses another logical CP to dispatch, and the whole process starts again.

Storage protection

This is one of the mechanisms that is implemented by z/Architecture to protect main storage. With multiprocessing, hundreds of tasks can run programs accessing physically any piece of main storage. Storage protection imposes limits and a task can access only the main storage locations with its own data and programs, or, if allowed, to read areas from other tasks. Any violation of this rule causes the CP to generate a program interrupt that is called as a *protection exception*, and the OS ends that task abruptly. The input of storage protection is a real storage address, and all real addresses that are manipulated by CPs or channels must go through the storage protection verification before being used as an argument to access the contents of main storage.

For each 4 KB block of main storage, there is a 7-bit control field that is called a *storage key*. This key is made up of access control bits (also called as the PSW key), a fetch bit, a reference bit, and a change bit. The current PSW is a storage circuit within the CP. It contains information that is required for the running of the currently active program, that is, it contains the current state of a CP. The OS may alter or inspect storage key bits by issuing special assembly language instructions that are based on the following logic.

- ▶ If a running program has a PSW key equal to 0000, it may access any frame in memory.
- ▶ If the fetch bit is off in a frame, any program can read the contents of that frame.
- ▶ To read the contents of a frame where the fetch bit is on, the PSW key of the running program must match the access control 4 bits in the storage key of the frame.
- ▶ To alter (write) the contents of a frame, the PSW key of the running program must match the access control 4 bit in the storage key of the frame.

z/OS uses storage protection by managing frame storage key values and running the program PSW key field in the current PSW. For example, several z/OS routines run with a PSW key value of 0, and others run with PSW key value of 1. Application programs or normal user code has a PSW key value of 8.

Interrupt processing

An interrupt occurs when the CP detects some special events during processing. Once interrupted, the CP saves the existing PSW into a main storage location, fetches a new PSW from a known storage location and loads it, and stores information about the cause of the interrupt in another main storage location. All these main storage locations are kept in an 8 KB area at the beginning of the real storage for each processor, which is called the Prefix Storage Area (PSA).

There are six types of interruption conditions: restart, external, supervisor call (SVC), program check, machine check, and I/O. In a sense, an interrupt is a sort of branching, but there is a logical difference between a BRANCH instruction that is issued in a program and an interrupt. A BRANCH is simply a twist in the logic of the program. In an interrupt, however, one of the six interruption conditions occurred that must be brought to the attention of the OS owning the program.

A restart interrupt is operator-initiated, a program check is CP-initiated, an SVC is OS initiated, an I/O interrupt is initiated by the I/O supervisor, a machine check is initiated by an equipment malfunction detection mechanism, and external interrupts by different causes that are not connected with the active program.

Time of day

Each CP in an installation provides access to TOD clock, which is shared by all CPs in the installation. A TOD programmable register is associated with the TOD clock. Each CP has its own clock comparator, CP timer, and TOD programmable register. The TOD clock is a 104-bit counter register inside each PU. The TOD clock provides a high-resolution measure of real time suitable for the indication of date and time of day. This timing is precious information for OSs, databases, and system logs. The cycle of the clock is approximately 143 years (from all bits zero to all bits zero again). TOD follows the Coordinated Universal Time (UTC) that is derived from the atomic time TA1 (based in Cesium 133 radioactivity), and is adjusted with discrete leap seconds to keep reasonably close to UT1 (based on the Earth's rotation). Incrementing the TOD clock does not depend on whether the CP is in a wait state or whether the CP is in operating, load, stopped, or check-stop states. The TOD cannot be used by MVS for time accounting for tasks in z/OS.

The CP timer is a binary counter with a format that is the same as that of bits 0 - 63 of the TOD clock, except that bit 0 is considered a sign. When both the CP timer and the TOD clock are running, the stepping rates are synchronized so that both are stepped at the same rate when a specified amount of time has elapsed. The CP timer is used by MVS for accounting purposes, that is, it is used to time how much CP a task or a service request consumes.

Clustering

Since the early 1970s, mainframes have been designed as multiprocessor systems, even when only a single processor was installed. Most of the mainframe installations typically use clustering techniques, although they do not normally use the terms *cluster* or *clustering*. A clustering technique can be as simple as a shared physical storage device configuration where manual control or planning is needed to prevent unwanted data overlap. More common today are configurations that allow sharing of locking and enqueueing controls among all systems. Among other benefits, this automatically manages access to data sets so that unwanted concurrent use does not occur.

A systems complex, or *sysplex*, is a collection of z/OS systems that cooperate, using certain hardware and software products, to process work. It is a clustering technology that can provide near-continuous availability. The most sophisticated of the clustering techniques is a *Parallel Sysplex*, which allows the linking of linear scalability to create a powerful commercial processing clustered system. Every server in a Parallel Sysplex cluster has access to all data resources, and every *cloned* application can run on every server. When used with *coupling* technology, Parallel Sysplex provides a *shared data* clustering technique that permits multisystem data sharing with high performance read/write integrity. As a result, work requests that are associated with a single workload, such as business transactions or database queries, can be dynamically distributed for parallel execution on nodes in the sysplex cluster that is based on available processor capacity.

A Parallel Sysplex relies on one or more *coupling facilities* (CFs). A coupling facility is a mainframe processor, with memory and special channels, and a built-in OS. It has no I/O devices, other than the special channels, and the OS is small. A CF functions largely as a fast scratch pad. It is used for three purposes:

- ▶ Locking information that is shared among all attached systems
- ▶ Cache information (such as for a database) that is shared among all attached systems
- ▶ Data list information that is shared among all attached systems

The information in the CF is in memory and a CF typically has a large memory. A CF can be a separate system or an LPAR itself.

A major difference between a sysplex and a conventional large computer system is the improved growth potential and level of availability in a sysplex. The sysplex increases the number of PUs and z/OS OSs that can cooperate, which in turn increases the amount of work that can be processed. Sysplex design characteristics help businesses to run continuously, even during periods of dramatic change. Sysplex sites can dynamically add and change systems within a sysplex, and configure the them for no single points of failure. Through this state-of-the-art cluster technology, multiple z/OS systems can be made to work in concert to more efficiently process the largest commercial workloads. With sysplex, the ability to perform rolling hardware and software maintenance in a nondisruptive manner allows business to implement critical business function and react to rapid growth without affecting customer availability.

3.4.2 Software design

The latest technology and trends have created much flexibility and many improvements in hosting applications on different OSs. In the case of the mainframe, z/OS has proven to be a strong supporting OS for the underlying hardware architecture because the fundamental concepts and functions of z/OS aligns with the strengths of the hardware architecture and design.

This section focuses on z/OS. This section also describes the hypervisor function that is provided by z/VM that is widely deployed with z/OS.

z/OS provides concepts and functions such as dispatching, address spaces, SVCs, and address translation, to ensure that each instruction is run on the processor with great isolation and integrity in an efficient manner. Also, z/OS isolates the decision-making entities of security clearly from the resource managers that makes those security requests. This is achieved through functions such as SAF and External Security Managers (ESMs). z/OS also maintains an excellent record of all system events through its event recording mechanisms with functions, such as SMF.

Dispatchable units

Multiprocessing capabilities of the System z architecture can be better explained with respect to z/OS by starting with the concept of *process*. A *process* is the serial running of programs to solve one problem of a business unit. Programs and CPs were developed because processes must be run. A process can start in one CP, be interrupted, and later resume in another CP, and so on. Similarly, a *re-entrant* program can be run by different CPs on behalf of different processes. All the processes wait for the processor in a queue that is based on their priorities.

In the OS, processes are *born* and *die*, either normally or abnormally. A process dies normally when its last program completes normally. A process dies abnormally when one of its programs tries to run something wrong or forbidden. The amount of resources that is consumed is charged to the process, and not to the program. Also, when there are queues for accessing resources, the priority to be placed in such queues depends on the process and not on the program.

In z/OS, processes are called *dispatchable units* (DUs), which consist of tasks and service requests. The control program creates a task in the *address space*, as a result of initiating the running of the process, also called the job-step task, by using an SVC. You can create additional tasks in a program to run their business processes. These business processes constitute the different workloads on mainframe.

Virtual storage

z/OS uses both types of physical storage, central and auxiliary. Main storage is also referred to as main storage, or real storage. In z/OS, each user has access to *virtual storage*, rather than physical storage. This use of virtual storage is central to the unique ability of z/OS to interact with many users concurrently while processing the largest workloads.

Virtual storage means that each running program can assume that it has access to all of the storage that is defined by the architecture's addressing scheme. The only limit is the number of bits in a storage address. This ability to use many storage locations is important because a program may be long and complex, and both the program's code and the data it requires must be in main storage for the processor to access them.

z/OS supports a 64-bit addressing scheme, which allows an address space to address, theoretically, up to 16 EB of storage locations. In reality, the mainframe has much less main storage that is installed, depending on the model of the computer and the system configuration. To allow each user to act as though this much storage really exists in the computer system, z/OS keeps only the active portions of each program in main storage. It keeps the rest of the code and data in files that are called *page data sets* on auxiliary storage, which usually consists of a number of high-speed direct access storage devices (DASDs). Virtual storage, then, is this combination of real and auxiliary storage. z/OS uses a series of tables and indexes to relate locations on auxiliary storage to locations in main storage. It uses special settings (bit settings) to track the identity and authority of each user or program. z/OS uses a variety of storage manager components to manage virtual storage.

A virtual address identifies a location in virtual storage. When a virtual address is used for an access to main storage, it is translated by means of dynamic address translation (DAT) to a real address, which is then further converted by prefixing to an absolute address. A real address identifies a location in real storage.

Address space

The range of virtual addresses that the OS assigns to a user or separately running program is called an *address space*. This is the area of contiguous virtual addresses that are available for running instructions and storing data. The range of virtual addresses in an address space starts at zero and can extend to the highest address permitted by the OS architecture. For a user, the address space can be considered as the runtime container where programs and their data are accessed. z/OS provides each user with a unique address space and maintains the distinction between the programs and data belonging to each address space. Within each address space, the user can start multiple tasks by using TCBs that allow multiprogramming.

A z/OS address space is like a UNIX process, and the address space identifier (ASID) is like a process ID (PID). Further, TCBs are like UNIX threads in that each OS supports processing multiple instances of work concurrently. In z/OS, the use of multiple virtual address spaces provides virtual addressing capability to each job in the system by assigning their own separate virtual address space.

With an address space, errors are confined to that address space, except for errors in commonly addressable storage, thus improving system reliability and making error recovery easier. Programs in separate address spaces are protected from each other. Isolating data in its own address space also protects the data.

Dynamic address translation

DAT is the process of translating a virtual address during a storage reference into the corresponding real address. DAT is implemented by both hardware and software through the use of page tables, segment tables, region tables, and translation lookaside buffers. If the virtual address is already in main storage, the DAT process may be accelerated through the use of translation lookaside buffers. If the virtual address is not in main storage, a *page fault* interrupt occurs, and z/OS is notified to bring the page from auxiliary storage.

The hardware function of DAT is in charge of translating the virtual address, constructing a location for the real address in main storage, or generating a page fault. z/OS is in charge of maintaining the tables, deciding the candidate pages for real storage frames, maintaining auxiliary storage (or page data sets), supporting page faults, and running the page stealing process.

DAT allows different address spaces to share program or read-only data because virtual addresses in different address spaces can be made to translate to the same frame of main storage. Otherwise, there must be many copies of the program or data, one for each address space.

Supervisor calls

An SVC is a type of interrupt that is triggered in the CP by the running of the SUPERVISOR CALL instruction.

z/OS runs tasks in two different modes: *problem* mode and *supervisor* mode. The instructions of a task identify their mode to the processor by setting the value of a bit in the PSW. All normal user tasks and general programs run in problem mode. System tasks or services run in supervisor mode and are triggered by SVC interrupts.

These interrupts occur when the program issues an SVC to request a particular system service. An SVC interrupts the program being run and passes control to the supervisor so that it can perform the service. Programs request these services through macros, such as OPEN (open a file), GETMAIN (obtain storage), or WTO (write a message to the system operator).

Every SVC is identified by a number that is picked by the interrupt handler to identify the routine that is associated to it from the SVC table. After the request is processed by the z/OS SVC routine, the interrupted program can regain control by restoring of its registers and the PSW.

System Authorization Facility

The SAF is part of z/OS and provides the interfaces to the callable services that are provided to perform authentication, authorization, and logging. SAF does not require any other product as a prerequisite, but overall system security functions are greatly enhanced and complemented if it is used concurrently with other ESMs.

The key element in SAF is the SAF router, which provides a common focal point for all products providing resource control. This focal point encourages the use of common control functions that are shared across products and across systems. The resource managing components and subsystems call the SAF router as part of certain decision-making functions in their processing, such as access-control checking and authorization-related checking. These functions are called *control points*.

The resource manager may provide its own security, but is more likely to call SAF to perform an authorization check. The SAF conditionally directs control to ESM, or to a user-supplied processing routine, or both, when receiving such requests from a resource manager. This context is shown in Figure 3-3.

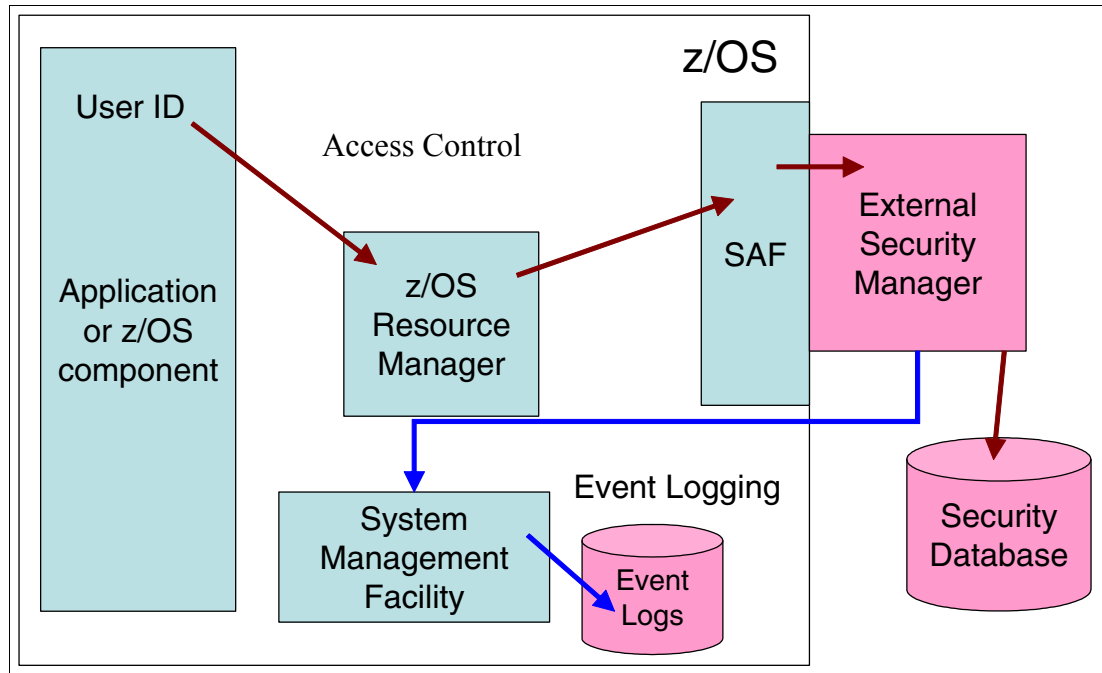


Figure 3-3 z/OS security overview

SAF passes the user ID, resource name, and access type that are requested to the ESM, which refers to its own database to gather enough information to pass back to the resource manager. The resource manager decides to allow or deny access based on the security information that is received from the ESM. Additionally, the external security manager may request that an event log record be created by the SMF.

SAF supports the use of common control points across products and across systems. Applications and system components call these common control points to interface with the ESM. Security on z/OS is therefore centralized on SAF and the installed ESM. z/OS does not contain an ESM, although there are several available to the installation. When there is no ESM installed, SAF creates the security constructs that are needed by system services. SAF provides an installation with centralized control over system security processing by using a system service that is called the *SAF router*. The SAF router provides a focal point and a common system interface for all products providing resource control. ESMs provide tables to SAF that direct specific calls for security functions to specific routines within the ESM. The use of these tables allows z/OS to provide support for pluggable ESMs, thus giving the installation the flexibility to determine which ESM to use. SAF and the SAF router are present on all z/OS systems, regardless of whether an ESM is installed or not.

When an authentication request is made to z/OS, a call to SAF occurs. SAF returns a security context that contains the user ID and the user ID's system-wide attributes. These attributes indicate whether the user has special privileges, such as the ability to run auditing functions or operator commands. The security context remains with the process or thread until it is removed or replaced through subsequent SAF calls. When a process attempts to access a resource, such as a file, printer, or system service, the access control list (ACL) of the resource is checked by the ESM to determine whether the user ID has the requested privilege. The ESM, through SAF, returns specific codes to indicate the status of the request.

SAF is accessed through the RACROUTE macro. RACROUTE provides the services to authenticate a user ID, interrogate access permissions, perform security event logging, and obtain a security context for address spaces and tasks running on the system. Regardless of the ESM installed, applications and system services use RACROUTE. The RACROUTE feature of z/OS removes the need for the application requesting security services to understand the underlying system security infrastructure that is implemented by the installation. The application does not need to know which ESM is installed, or indeed if one is present at all. The SAF router uses the routing table to associate the correct ESM programs with the related RACROUTE call.

Understanding security on z/OS means understanding how programs interface with the ESM. Irrespective of the ESM, applications must issue a RACROUTE, or have one issued on their behalf, to gather the information that is needed for subsequent authorization checks.

Application programming interfaces

System services are designed with security integrated into their functionality. For example, when a data set is opened by a user, the system code that is responsible for accessing the data set contacts the security manager to assess whether the access is permitted. The application the user is running has no need to worry about the data set security.

Just using z/OS does not ensure application security. Security should be considered while designing the application as it is difficult to retro-fit security into a program after it is written. However, there are many interfaces into the security functions of z/OS based. There are APIs for batch, UNIX System Services, COBOL, Java, or C/C++.

External Security Managers

When someone talks about the security of a mainframe, it is usually the ESM to which they refer. Although ESM provides the responses for security requests, it is often misunderstood that ESM is just a decision function and there are many other security and integrity concepts outside of it. The ESM distinguishes between the resources to be protected and the entities that want to access them. Entities are nothing but the users or applications on their behalf and resources are everything that is protected. The ESM provides the capability to describe uniquely resources and users. When users attempt to access a resource, the system calls the ESM to indicate whether that user has the requested access permissions. The ESM responds to the request with the information it has stored in its database. It is then the system's decision, not the ESM's, to allow or deny the access request.

The ESM retains information about the users, resources, and access authorities in security profiles within its database, and refers to the profiles when deciding which users should be permitted access to protected system resources.

Here are the major advantages of using a security product for securing access to resources:

- ▶ One product may be used to implement the security requirements for multiple subsystems.
- ▶ All of the security information may be stored and maintained in one place.

Having a centralized database repository with the entire system's security specifications has eliminated, or minimized, the previous requirements to have security information distributed among several subsystems, and to have the security enforcement functions implemented in multiple products.

There are several security managers available for z/OS. Here are some of the most popular ones:

- ▶ Secure Server for z/OS RACF from IBM
- ▶ eTrust CA-ACF2 Security for z/OS from Computer Associates
- ▶ eTrust CA-Top Secret Security for z/OS from Computer Associates

Most of the examples or references in this publication use the perspective of RACF to illustrate the concepts.

Authorized program facility

Many system functions, such as SVCs or special paths through SVCs, are sensitive. The processor can run in supervisor or problem state as indicated by bit 15 in the current PSW. When in problem mode state, the processor refuses to run a privileged instruction, generating a program interrupt that causes the program to end abnormally. All privileged instructions, if they are misused, might jeopardize the integrity and the security of the multi-transaction z/OS system. Also, certain SVC routines when misused by the caller program might compromise system integrity and system security. Hence, access to these routines must be restricted to only authorized programs.

z/OS has a feature that is called the *authorized program facility* (APF) that allows selected programs to perform restricted system functions. The installation identifies which libraries contain those *special* programs at the time of system initialization and those libraries are called APF libraries.

An APF-authorized program can activate in the supervisor state and can access or modify system control blocks to perform its operations. It can run privileged instructions in the supervisor state. It can turn off logging. APF can be used to restrict the authorization to call SVC routines or other sensitive system functions to a set of known programs. z/OS also ensures that a job step running a program from any authorized library has all the other modules or programs it calls as authorized. This helps in avoiding counterfeiting of unauthorized loops to authorized programs. Every APF-authorized program must also be link-edited with a special flag called an *authorization code* (AC=1) to enable the privileged status,

APF is a configuration mechanism containing the list of libraries that a system or a security administrator wants to authorize. This list normally contains system libraries and the libraries of other OSs components or subsystems that run authorized programs that interact with core system functions. There can be business applications containing user programs that might need to be authorized according to their design and business process requirements. As the authorization is only at the library level, any program or module within the authorized library enjoys the privileged status until that library is in the APF list. Also, any new program or module can be added to the library that is in the APF list by users that have update access to that library, thus authorizing a new program with privileged status. Care and caution must be taken when you add libraries to an APF list and in maintaining access authorizations to those libraries.

System Management Facility

SMF is a component of z/OS that collects and records system and job-related information. SMF maintains a data set of event records, capturing every event of the system, whether the event is related to accounting, auditing or logging. There are many types of SMF records that have a specific meaning to security auditors. From a security aspect, the ESM creates records for many types of events such as user activity, access authorizations, and failure events. z/OS provides services that you can use to extract the security event data from SMF, such as the SMF Dump Utility.

The volume and variety of information in the SMF records enables installations to produce many types of analysis reports and summary reports. For example, by keeping historical SMF data and studying its trends, an installation can evaluate changes in the configuration, workload, or job scheduling procedures. Similarly, an installation can use SMF data to determine system resources that are wasted because of problems, such as inefficient operational procedures or programming conventions. SMF runs as a separate task on z/OS to perform this continuous data capturing. It captures everything starting from system initialization until the shutdown. It initializes during the early stages of system start and is usually the last subsystem to be brought down. The last z/OS command that is used to bring down the SMF subsystem is `HALT EOD`, where EOD means *End of Day*.

With its extensive event recording capability, which is used for various purposes, the SMF data itself is sensitive, and requires protection from unauthorized access. It also must be selective about what events or records to capture and which ones to ignore while considering the business or security needs and the impact to performance.

File system and catalog

z/OS manages data through *data sets*. The term data set refers to a file that contains one or more records. A data set can be a source program, a library of programs, or a file of data records that is used by a processing program. Data set records are the basic unit of information that is used by a processing program. The characteristics of traditional z/OS data sets differ considerably from the file systems that are used in UNIX and PC systems.

Almost all z/OS data processing is record-oriented. Byte-stream files are not present in traditional processing, although they are a standard part of z/OS UNIX. Data sets have control attributes that include the record format, the record length, and the block size. Users must define the amount of space to be allocated for a data set (before it is used), or these allocations must be automated through the use of a z/OS component that is called the Data Facility Storage Management Subsystem (DFSMS). With DFSMS, the z/OS system programmer or storage administrator can define performance goals and data availability requirements, create model data definitions for typical data sets, and automate data backup. DFSMS can automatically assign, based on an installation policy, those services and data definition attributes to data sets when they are created. Other storage management-related products can be used to determine data placement, manage data backup, control space use, and provide data security.

An *access method* defines the technique that is used to store and retrieve data. Access methods have their own data set structures to organize data, system-provided programs (or macros) to define data sets, and utility programs to process data sets. Access methods are identified primarily by the data set organization. z/OS users, for example, use the basic sequential access method (BSAM) or queued sequential access method (QSAM) with sequential data sets. z/OS libraries are known as partitioned data sets (PDS) and contain *members*. Source programs, system and application control parameters, and executable modules are almost always contained in libraries. Virtual Storage Access Method (VSAM) is another access method that provides much more complex functions than other disk access methods, and is primarily for applications. z/OS UNIX System Services allow users to create UNIX file systems and file system directory trees on z/OS, and to access UNIX files on z/OS and other systems.

DASD volumes are used for storing data and executable programs, including the OS itself, and for temporary working storage. DASD labels identify DASD volumes and the data sets they contain. One DASD volume can be used for many different data sets, and space on it can be reallocated and reused. On a volume, the name of a data set must be unique. A data set can be located by device type, volume serial number, and data set name. These requirements can be shortened to knowing only the data set name if the data set is cataloged.

The *system catalog* is a single logical function, although its data may be spread across the master catalog and many user catalogs. In practice, almost all disk data sets are cataloged. One side effect of this is that all cataloged data sets must have unique names.

Still, many elements of UNIX have analogs in z/OS. In z/OS, the user prefix that is assigned to z/OS data sets points to a user catalog. Typically, one user owns all the data sets whose names begin with his user prefix. For example, all the data sets belonging to the user ID IBMUSER begin with the *high-level qualifier* (prefix) IBMUSER. There might be different data sets named IBMUSER.C, IBMUSER.C.OTHER and IBMUSER.TEST under the same prefix. In the UNIX file system, IBMUSER can have a user directory named /u/ibmuser. Under that directory, there might be a subdirectory that is named /u/ibmuser/c, and /u/ibmuser/c/pgma might point to the pgma file.

A comparison of z/OS data sets and file system files is shown in Figure 3-4.

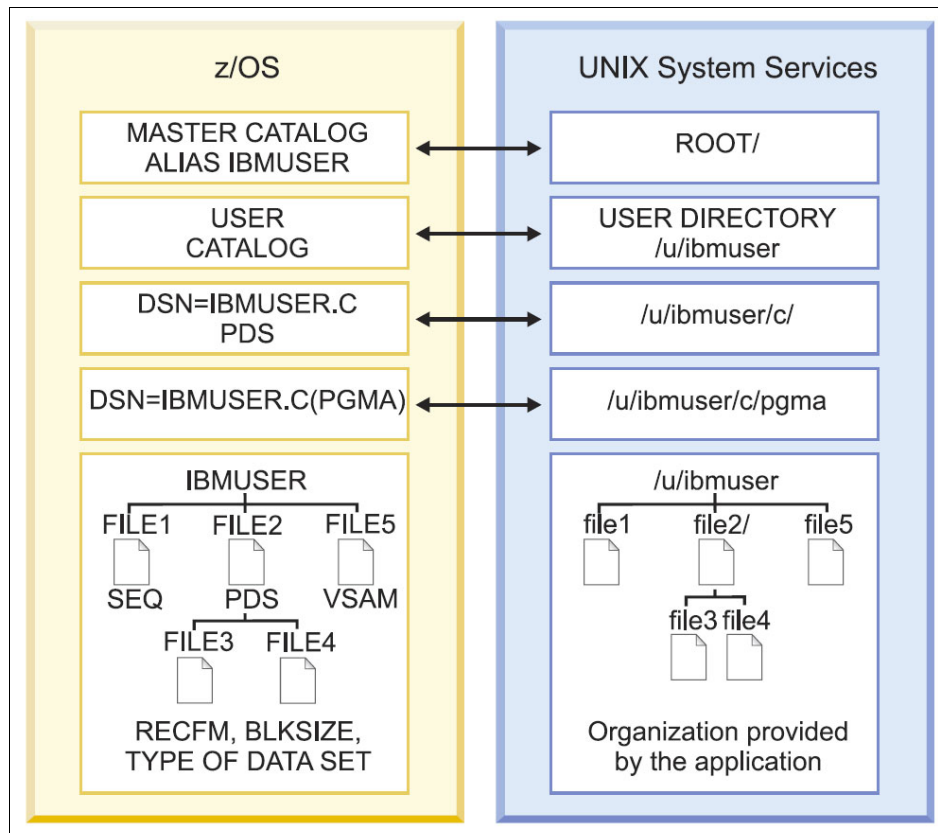


Figure 3-4 Comparison of z/OS data sets and file system files

Of the various types of z/OS data sets, a PDS is most like a user directory in the file system. In a PDS such as IBMUSER.C, there might be members (files) PGMA, PGMB, and so on. Along the same lines, a subdirectory such as /u/ibmuser/c can hold many files, such as pgma and pgmb.

Workload Manager

For z/OS, the management of system resources is the responsibility of the WLM component. WLM manages the processing of workloads in the system according to the company's business goals, such as response time. WLM also manages the use of system resources, such as processors and storage, to accomplish these goals.

WLM is suited to a sysplex environment. It tracks system use and workload goal achievement across all the systems in the Parallel Sysplex and data sharing environments. For example, WLM can decide the z/OS system on which a batch job should run, based on the availability of resources to process the job quickly.

WLM has three objectives:

- ▶ To achieve the business goals that are defined by the installation by automatically assigning sysplex resources to workloads based on their importance and goals. This objective is known as goal achievement.
- ▶ To achieve optimal use of the system resources from the system point of view. This objective is known as throughput.
- ▶ To achieve optimal use of system resources from the point of view of the individual address space. This objective is known as response and turnaround time.

Thus, WLM must make decisions that represent trade-offs between conflicting objectives. Other z/OS components, transaction managers, and database managers can communicate to WLM a change in status for a particular address space (or for the system as a whole), or to use the WLM decision-making power.

Hypervisor

Hardware partitioning and hypervisors are the two main implementation approaches for server virtualization. Hardware partitioning subdivides a physical server into separate computing environments, each of which can run an OS instance and the associated compatible applications. Software partitioning employs a software-based *hypervisor* to enable individual OSs to run on any or all of the CPUs.

Hypervisors use a thin layer of code to achieve fine-grained, dynamic resource sharing. Hypervisors allow multiple OSs to run on a host computer at the same time. Hypervisor technology originated in the IBM VM/370, the predecessor of the z/VM. z/VM allows the sharing of the mainframe's physical resources, such as disk, memory, network adapters, and CPUs. These resources are managed by a hypervisor. z/VM's hypervisor is called the Control Program (CP). CP is responsible for virtualizing your zSeries machine's real hardware, and allows many VMs to simultaneously share the hardware resource. CP also handles the creation and management of VMs. It can be considered the OS component of z/VM because it is responsible for managing real devices and resources and sharing them among various tasks and users that need them.

The VM simulates the existence of a dedicated real machine, including processor functions, memory, networking, and I/O resources. OSs and application programs can run in VMs as guests. Guests can run on either first-level z/VM or second-level z/VM. A first-level z/VM is the base OS that is installed on top of the real hardware. A second-level OS is a system that is created upon the base z/VM OS. Therefore, z/VM as a base OS runs on the hardware, and a guest OS runs on the virtualization technology. When the user logs on to z/VM, the hypervisor creates a VM, which can run one of many different mainframe OSs, such as z/OS, z/TPF, Linux, z/VSE, CMS, or z/VM.

3.5 Role of the mainframe within an enterprise security architecture

This section takes a closer look at how the mainframe fits into today's IT infrastructure of a modern and interconnected organization. This chapter has described the drastically changing roles that the mainframe has played since the 1960s. While focusing on a centralized enterprise security architecture strategy, it becomes apparent that security management on the mainframe must fit into this new role as well.

Instead of relying on physical access patterns or segregated applications and data stores, the mainframe must work hand-in-hand with centralized functions, such as policy control, identity provisioning, and security intelligence.

To describe the functional security within an enterprise security architecture in a way that is as complete as possible, this section follows the outline of the IBM Security Blueprint. In particular, this section investigates the Foundational Security Management components that are introduced in 2.2.1, "Foundational Security Management" on page 24:

- ▶ Risk and Compliance Assessment
- ▶ Command and Control Management
- ▶ Security Policy Management
- ▶ Identity, Access, and Entitlement Management
- ▶ Data and Information Protection Management
- ▶ Software, System, and Service Assurance
- ▶ Threat and Vulnerability Management

3.5.1 Risk and Compliance Assessment

These services enable the IT organization to collect, analyze, and report security information and security events to identify, quantify, assess, and report on IT-related risks that can contribute to the organization's operational risk. Security dashboards provide situational awareness to allow day-to-day risk management. This component covers *risk aggregation and reporting, IT security risk processes, business controls management, resiliency and continuity management, compliance reporting, and legal discovery services*.

Functional overview

There are three areas of primary inputs into the Risk and Compliance Assessment services:

- ▶ Compliance metrics that are driven by organizational policy and industry and regulatory standards.
- ▶ Events and artifacts that are provided by security intelligence and analytics services that assess the organization's IT infrastructure, applications, data management, and security solutions.
- ▶ IT security knowledge that can be delivered through internal or external skills or services.

These different areas provide input for the organization's risk posture. *Risk Management* practices can now assess the risk posture along with the compliance metrics and the target values for the adequacy for mitigating related risks to the level set by the organization. The Command and Control Management practice can then create new and updated service directives and objectives, which lead to new or updated policies.

On top of this flow, the compliance metrics are used to identify non-compliance postures for IT systems and services. The *Compliance Management* practice assesses compliance of the IT environment by identifying and examining gaps between actual and intended compliance values that are defined in the compliance metrics.

Another important element in the Risk and Compliance Assessment practices is provided by the Analytics Services. These services can collect and provide information about events in a synthesized, correlated, platform-independent, and less technical format. The aggregation of security logs and subsequent derivation of security information, which now is understandable by less technical people on the business level, is provided to the Risk and Compliance Assessment services to further analyze data in a risk context.

Accurate, complete, and timely data is a foundation for effective risk management. However, data alone does not ensure that you make effective decisions about risk. The correct information must be presented to the correct people at the correct time. Reports that are based on risk data should be accurate, clear, and complete. They should contain the correct content and be presented to the appropriate decision-makers in a time frame that allows for an appropriate response.

Role of the mainframe

The role of the mainframe in an enterprise-wide Risk and Compliance Assessment concept is that of a compliance monitoring, compliance reporting, and accountability.

Compliance monitoring is the mechanisms of measuring compliance on system. *Compliance reporting* is the mechanisms of reporting on the state of compliance of the system. Together, these capabilities allow an organization to obtain an almost real-time look at their overall compliance posture.

The mainframe has one of the efficient event recording and reporting mechanisms across all platforms, which is implemented by the SMF. SMF is activated during the early stages of system initialization and starts recording every event across the system functions and components. Also, SMF is the last subsystem to be deactivated during a system shutdown. This architecture design ensures that no single event or information that is associated with it is left unnoticed. Such an extensive collection of data allows you to mine and report risks and compliance issues within each component. For example, every access to a system resource is logged in to SMF and makes it fool-proof because it cannot be bypassed or disabled. It is so effective that the whole OS comes to a halt if it is unable to log data to SMF because of any issues. It continues processing only through operator intervention resolving the issue. SMF is a system component and works independently. Almost all the other components and subsystems coordinate with SMF and use it to aid their own recording and reporting functions. There are various SMF utilities and tools that can report and present the data based on your specific needs.

With the ESMs, the ability to log information, such as attempted accesses to a resource and to generate reports containing that information, can prove useful to a resource owner and is important to a smoothly functioning security system. For example, RACF can identify and verify a user's ID, recognize the resources that the user can access, and record the events where user-resource interaction is attempted. This function records actual access activities or variances from the expected use of the system. Logging all access attempts allows you to collect valuable data that can be used for compliance monitoring and reporting. ESMs can log the information and perform real-time alerting by sending messages to the operator console that can be captured and set to trigger further actions through automation tools. ESMs also can maintain statistics information, such as the date, time, and number of times that a user logs on to the system and the number of times a user accesses a specific resource. This information can help analyze and control operations more effectively.

Many of the subsystems that are deployed on the mainframe can contribute towards the collection of important and valuable log information for themselves. For example, in DB2 for z/OS, the combination of collecting audit event records with audit policy, processing the resultant events into DB2 tables, and writing SQL based audit-based processes allows you to create a relevant audit trail report. In addition, products such as IBM InfoSphere® Guardium® helps satisfy most requirements for monitoring and alerting without impacting SLAs and performance, and without requiring changes to databases or applications. With the InfoSphere Guardium framework, the collection of events are now policy-driven, and does not rely on the use of native DBMS trace facilities. IBM InfoSphere Guardium S-TAP® for DB2 for z/OS (also referred to as InfoSphere and S-TAP) collects and correlates data access information from various DB2 sources to produce a comprehensive view of business activity for auditors.

In the context of Risk and Compliance Assessment, Centralized Analytics Services can help evaluate and communicate your overall risk posture. The Analytics Services provide a Control Center for logging, viewing, analyzing, alerting, and reporting on events across, rather than within, the IBM Security Framework Security Domains. Including a complete collection of mainframe logs and events is crucial when it comes to an enterprise-wide risk and compliance assessment. An example is a timeline of configuration changes for a mainframe database to prove to an auditor who authorized the changes that caused an issue of non-compliance.

IBM Health Checker for z/OS provides a foundation to help simplify and automate the identification of potential configuration problems before they impact system availability and compliance. It consists of a framework, which manages functions such as check registration, messaging, scheduling, command processing, logging, and reporting. The framework is provided as an open architecture in support of check writing. Checks evaluate settings and definitions specific to products, elements, or components. Checks are provided separately and are independent of the framework. The architecture of the framework supports checks that are written by IBM, independent software vendors (ISVs), and users.

3.5.2 Command and Control Management

These services provide the command center for *security management* and the *operational security capabilities* for IT and non-IT assets and services to ensure protection, response, continuity, and recovery.

Command and Control Management fulfills both a strategic and a tactical role. The strategic role involves defining security policies. The tactical role involves coordination of the security operations. It covers the following topics:

- ▶ Ensuring that physical and operational security is maintained for locations, assets, humans, the environment, and utilities
- ▶ Providing surveillance and monitoring of locations, perimeters, and areas
- ▶ Providing top-level incidents that are delivered by security intelligence and analytics for further investigation
- ▶ Enforcing entry controls
- ▶ Providing for positioning, tracking, and identification of humans and assets
- ▶ Providing a focal point for continuity and recovery operations

Functional overview

The Service Management Infrastructure is fundamental to Command and Control Management because it relies on the Service Management Infrastructure to coordinate communication to other foundational security services. The personnel that are associated with the Command and Control Management services are also actors in the Service Management Infrastructure processes.

For example, a change that impacts security might need be approved by Supervisory Control and Delegation of Authority if the authority for approval of a specific level of changes (such as a major update to the security architecture of the network perimeter) is not correctly delegated and therefore is above the clipping level of established delegations.

Security Service Levels are the key output of Command and Control Management, and therefore are the most important data item for the Foundational Security Management services.

Role of the mainframe

The role of the mainframe in an enterprise-wide command and control management concept of autonomic monitoring and reporting on service levels.

Mainframes monitor business services and track them against business objectives and technology infrastructures. A mainframe shows the operational status of services using prebuilt reports, scorecards, and dashboards for fast data analysis. It helps you assess service levels throughout an organization for more effective service management. The service levels for mainframe include the availability of OS, software, storage, and subsystems. The mainframe OS is suited for continuous, high-volume operation with high security and stability.

For example, the WLM component of the mainframe ensures that the service levels are managed effectively to meet the business and compliance requirements. It is so effective that the inputs to WLM are provided in natural business language that is interpreted and converted into rules to manage the required service levels. WLM adjusts to the response times of workloads based on the historical trend that is aligned with the overall service level objectives. The capability of mainframe hardware to host distributed platforms on blade servers through the z/BX extensions help centralize the workload management functions through the Unified Resource Manager component. These business service level objectives help maintaining compliance with respect to licensing and use.

In terms of the mainframe's strategic role in command and compliance for security, the mainframe by itself defines certain core security policies at the base level through its configuration, including the concepts of real storage management and APF. Any unexpected violation of security policy terminates the requesting task with an exception and abnormally ends it to protect the system functioning and resume its operation. It also *dumps* the contents of real storage and system status during the time of such an exception so that the administrator can debug and identify the cause of the issue so that corrective actions can be taken. Some of the functions, such as the Resource Recovery Services (RRS), help recover from abnormal failures to ensure application availability. The mainframe is designed to enable redundancy at every level of its design and is capable of recovery from any kind of failure, either within the system or outside.

IBM has a strong security compliance program for the mainframe. This is evident from the EAL certificate levels for various mainframe components and the z/OS Statement of Integrity that ensures action on any identified or reported security vulnerabilities. Every piece of code that runs on mainframe is reviewed on security aspects by an independent security forensic team to certify its release. This includes even the APARs and fixes that are released by IBM. Any such security vulnerabilities are immediately fixed through a *security integrity* APAR and customers can subscribe to be notified of such releases. Some of the critical fixes are even released as highly pervasive APARs (HIPERs) to alert the customers of potential vulnerability. These mechanisms help ensure reporting, recovery, and compliance of security policies and requirements.

3.5.3 Security Policy Management

These services provide all the capabilities and repositories to author, discover, analyze, transform, distribute, evaluate, and enforce security policies.

Security Policy Management involves defining the security policies that are aligned with the business goals to reach compliance levels and mitigate risks to an acceptable level. It deals with setting up a governance framework to define and enforce policies and measure their effectiveness, reporting back to Governance, Risk, and Compliance.

Functional overview

The Security Policy Infrastructure is one of the key components for Security Policy Management, as it provides the containers for the various policies, related standards, procedures, and guidelines. It can automate the workflow for the various administration activities, and the deployment and communication with Policy Decision Points and Policy Enforcement Points.

Security policies represent the key output of Security Policy Management and are the most frequently appearing data item for Policy Administration, Policy Decision Points, and Policy Enforcement Points. A security policy is a document that outlines the rules and practices for system access. This document regulates how an organization manages, protects, and distributes its sensitive information and lays the framework for the security of the organization.

Role of the mainframe

The role of the mainframe in an enterprise-wide Security Policy Management concept is to enforce policy compliance.

Centralized control for consistent execution of security policies across multiple applications and users enables mainframes to provide authorized user access to systems, and a portfolio of products and services. It provides efficient management of the assessing, planning, implementing, auditing, and monitoring and maintaining of identities and access privileges.

A multitiered approach to comprehensive, integrated security management of a mainframe includes the following items:

- ▶ Prevention: Establish security policies and incorporate them into Information Systems.
- ▶ Access: Establish a policy-based infrastructure that allows management of access to sensitive or critical information assets.
- ▶ Enforcement: Facilitate compliance with standards, policies, rules, and settings.

Security in the mainframe spans into every component and their respective policies. The Input/Output Control Data Set (IOCDs) (see “Input/Output Control Data Set” on page 86) on the Support Element defines the basic set of configuration and policies that identifies the execution mode of the system, and allocation of resources to each LPAR and their limits. The support element also defines the processor and real storage allocation policies for the active configuration. Within the OS, various components manage and coordinate the resource allocation and authorization within and outside their limits in an efficient way. OS configuration defines the basic real storage allocation and initialization principles. These policies define the limits of different areas in the address space map for the current system, establishing the boundaries for system and user access. Similarly, the storage management subsystem (SMS) defines and enforces policies for logical storage allocation and accesses for users and administrators. WLM policies define and enforce service levels, resource allocation, use limits, and execution priorities. Access policies and rules in JES and SDSF allows authorized users to access the spool data within their limits and perform restricted functions for their role. Lastly, the IP Security policies of the TCP/IP stack let the administrator define and enforce policies to manage the network communication to the mainframe system.

The ESM, such as RACF, is integrated with all the other components of the mainframe, and serves the role of a security decision point across the platform. It can operate independently as a security policy enforcement function or can inherit the security policy from a centralized policy enforcement system. A security administrator can define policies in RACF that are effective for overall identity and Access Management on a particular mainframe image or group of images. The user or group profiles, and the access rules that are contained within the security server database, act as the second layer of security policies that control the accesses to protected resources for authorized users.

The comprehensive set of security add-on tools for the Security Server RACF makes it easier than ever before for organizations to automate access, authentication, and intrusion management of their mainframe environment.

3.5.4 Identity, Access, and Entitlement Management

These services provide capabilities that are related to roles and identities, access rights, and entitlements. The correct use of these capabilities can ensure that access to resources is given to the correct identities, at the correct time, and for the correct purpose. These services can also ensure that access to resources is monitored and audited for unauthorized or unacceptable use.

Functional overview

Identity lifecycle management begins with enrolling an individual who needs to access IT resources within the organization. The induction of a new person should be directed by one *authoritative repository*, which is typically the human resources department.

One of the objectives of an identity lifecycle management system is to maintain *one identity per individual*. After that identity is established, it must be maintained, modified, and monitored throughout its lifecycle. Eventually, when an individual leaves the organization, the identity with all its assigned attributes must be removed from all corporate systems.

Identity Management systems provide administrators in Command and Control Management with the tools and technologies to provision, maintain, and remove *one or more user IDs* for multiple target systems. They follow policy, standards, and procedures that are defined by Security Policy Management to create access profiles for each user ID. Those profiles can contain password policies, access restrictions for time-of-day or location, single sign-on profiles, and more.

Based on the job role of the individual, administrators also assign and provision *entitlements* to various applications and IT systems that can be deployed anywhere in the organization.

All of these activities are typically carried out under the Command and Control Management umbrella on a centralized identity lifecycle management system. Every one of these activities must be logged for audit, analyses, and security intelligence purposes. This system usually deals with people, but it can also handle hardware and network resources and even applications.

After the management aspects around identities, access, and entitlements are defined, the information must be provisioned to the Identity, Access, and Entitlement Infrastructure, which consists of the Policy Decision Points (PDP) and Policy Enforcement Points (PEP) that make authorization decisions and enforce them during run time.

Other IT infrastructure components include *access control points* to prevent unauthorized access to data, applications, and other IT resources both from a business operations perspective and from an IT administration perspective. These control points are driven by the policies and entitlements that are provisioned by the identity lifecycle management system.

In many cases, the credentials that are used in an authentication request are *signed or encrypted* so that a PDP can correctly validate the credentials.

As part of their design, applications typically use a set of application-specific roles. These roles define who can interact with an application, and in what way, to access the various services that the application provides. The application platform is typically responsible for defining associations between the application-specific roles and the organizational roles that are managed by the Identity, Access, and Entitlement Management system. These associations are then translated into access control policies that the application platform uses to grant or deny access to the application at run time.

These components and principles represent some of the major artifacts for Identity, Access, and Entitlement Management; this is not an exhaustive list, but it serves as the foundation to highlight the role of the mainframe in this context.

Role of the mainframe

The role of the mainframe in an enterprise-wide Identity, Access, and Entitlement Management concept is that of a Policy Decision Point and Policy Enforcement Point.

User ID, role, and entitlement information is provisioned from the centralized identity lifecycle management system to the mainframe, where it is typically stored inside an ESM such as RACF, or a combination of RACF and an LDAP-based directory. Technically, this information is provisioned using specific adapters that translate or transform the general user ID, role, and entitlement information into mainframe (RACF or LDAP) format.

Any further changes to this information, including its removal in case a person is dismissed from the organization, is provisioned in the same way.

At regular intervals, the centralized identity lifecycle management system should run verification, that is, *reconciliation* tasks. Here, the actual data in RACF is compared to the centrally stored information, and any manual changes that are administered on RACF itself are revealed and can be acted upon if they represent a policy violation. This way, the organization can ensure that manual changes to identity-related data on the mainframe can be detected. This can help reduce insider-related fraud and reveal bad administration practices.

The internal architecture design of RACF implements identity through *users* and *groups*. All the protected resources are authorized for a defined level of access to an identified user or a group. The groups can be defined based on different roles or business functions, according to the organization's security setup. A single user can be *connected* to any number of groups and with different levels of authorization. This enables the user to confirm the necessary levels of access that are granted to the connected group to various resources. In addition, RACF also allows *surrogate* access to enable users to present themselves with alternative identities to be authorized for specific functions.

After the identity-related information is provisioned to the mainframe, it can be used by other subsystems and applications in the usual way. First, RACF authenticates users and acts as the PDP by applying the applicable entitlements for a specific user. Then, acting as the PEP, the subsystem receives the access control decision from RACF and enforces proper access to its resources. Along every step of the way, proper audit and log records are written to document the complete authentication, policy decision, and policy enforcement steps.

3.5.5 Data and Information Protection Management

These services provide capabilities that protect unstructured and structured data from unauthorized access and data loss, according to the nature and business value of information. It also provides use and access monitoring and audit services.

Functional overview

One of the important element of Data and Information Protection Management is Cryptography, Key, and Certificate Infrastructure. Database servers, content repositories, and archive media capture data at rest must encrypt data in case the storage media is subject to out-of-band attacks, such as media theft, making the data and information protection management systems dependent on the Cryptography, Key, and Certificate Infrastructure. In addition to cryptographic protection for data that is stored on storage media, additional measures might be needed to protect the media and storage systems from tampering, theft, and copying.

Application security and network security are important for data and information protection management because compromised applications might be able to access the database servers and content repositories using the credentials of the application and issue unauthorized queries to them. Although message-level encryption and connection-level encryption can be used to protect data in transit, a secured network is important to prevent out-of-band attacks, such as copying traffic for later decryption, man-in-the-middle attacks, and DNS cache poisoning.

Role of the mainframe

Basically, any data that is on the mainframe is protected if the mainframe is configured and maintained properly. The architecture of the mainframe, if used properly, ensures that even the transient data that is on the real storage cannot be accessed or corrupted by an unauthorized user. Such functions are implemented through the use of storage protection keys and authorized program facilities.

Although ESMs such as RACF protect *data-at-rest* from unauthorized access and tampering, specific subsystems and middleware, such as DB2 for z/OS, organize their own security mechanisms and rules to prevent unauthorized access and maintain the integrity of data. This is true for systems that deal often with *data-in-motion*, such as CICS Transaction Server or IBM WebSphere MQ for z/OS, which implement features to ensure end-to-end data security for their transactions or messages. The diagnostic functions within the mainframe, such as the system dump processing, also ensures that the diagnostic information is clearly sanitized off any application or business data, and can be shared with external parties, such as the IBM Support, for problem analysis.

Encryption Facility for z/OS uses the existing strengths of the mainframe and z/OS. It is a host-based facility that uses existing centralized key management in z/OS and the hardware encryption capabilities of IBM mainframes. When sharing encrypted data between z/OS systems, the Encryption Facility for z/OS software also can use the mainframe hardware functions to compress the data, before the data is encrypted and written to the tape or DASD media. The z/OS services provide the basic certificate-handling functions that are needed to exchange Encryption Facility encrypted files with RSA protection of the data key. The services that are provided to generate and sign certificates can be seen as a local and limited certificate authority capability. These services also can be used within a wider implementation of a public key infrastructure.

There are two facilities that can assist in the certificate services that you need for Encryption Facility for z/OS. The first is the RACF **RACDCERT** command that is used in the RACF database, which is a method of creating and managing certificates and RSA keys. The second is the set of utility ISPF panels that is provided in ICSF. The z/OS base component that provides the cryptographic APIs that invoke the hardware coprocessor is the ICSF. ICSF provides cryptographic coprocessors administration facilities for those coprocessors that require a master key to be set.

3.5.6 Software, System, and Service Assurance

These services address how software, systems, and services are designed, developed, tested, operated, and maintained throughout the software lifecycle to create predictably secure software. This component covers the following items:

- ▶ Structured design
- ▶ Threat modeling
- ▶ Software risk assessment
- ▶ Design reviews for security
- ▶ Source code reviews and analysis
- ▶ Dynamic application analysis
- ▶ Source code control and access monitoring
- ▶ Code and package signing and verification
- ▶ Quality assurance testing
- ▶ Supplier and third-party code validation

Functional overview

Application Security, Network Security, and Storage Security are a few of the elements of Software, System, and Service Assurance.

Although secure coding practices, static analysis, and secure design practices can limit the vulnerabilities in an application, the applications typically rely on Application Security enforcement points to help detect and prevent attacks, such as cross-site scripting and SQL injection. Applications have dependencies in the Network Security infrastructure to make certain that ports are available for remote connections and to ensure the appropriate isolation of network traffic. Certain application-layer attacks can be detected and prevented through deep packet inspection and other types of network traffic analysis.

Applications define their dependency on storage infrastructure, and storage infrastructure components can be included in a VM image. In both cases, the definitions might impose security requirements on the storage infrastructure, including requirements to encrypt storage media, put it in a physically secure environment, and maintain data for a specified retention period.

Role of the mainframe

The role of the mainframe in an enterprise-wide Software, System, and Service Assurance concept is that of an Application Performance Management.

The constraints of unavailable or incomplete software environments can delay innovation and kill critical application development projects. In a mainframe, using the software development lifecycle (SDLC) decoupled from constraints allows parallel development and early testing for more agile delivery of new customer-facing functionality to market, with much higher quality and performance at lower infrastructure.

There are programmers and applications that use the z/OS system to run their programs, perform testing, create applications, and perform development functions. Applications may use system-supplied security services that are available with an ESM, or they may supply their own services to protect access. Application security can be used as an enhancement to the ESM security. z/OS provides an SAF interface so that applications are flexible enough to manage their security requirements.

Centralized security secures the information that is distributed among several subsystems, and allows security enforcement functions to be implemented in multiple products.

3.5.7 Threat and Vulnerability Management

These services provide capabilities that identify vulnerabilities in deployed systems and receive reports of vulnerabilities from outside sources, determine the appropriate response, and make proactive changes to deployed systems to maintain the security of the deployed system. Other capabilities collect security events and information from a wide range of sources to gain insight and detect possible threats through event correlation and security intelligence and analytics.

Functional overview

Event and logs are the most essential objects for the Threat and Vulnerability Management services, as they contain all the collected log and event information that is necessary to identify attacks.

Design is one of the important components for Threat and Vulnerability Management services, as you can use these services to derive potential attack and testing scenarios for vulnerability discovery and threat analysis services.

A deep and broad IT security knowledge is of key importance in Threat and Vulnerability Management. The type of knowledge that is required includes a deep technical understanding of platform-specific security functions and the ability to understand the performance of security attacks in a step-by-step manner. Besides having knowledge, security experts that work in Threat and Vulnerability Management always must be up to date on new technologies so that they can identify potential new types of threats that might come with these innovations. Alongside the IT Security Knowledge, it is also necessary to have skills in using the various security analysis and testing tools.

Finally, the provision of these services requires the ability to understand new security attack patterns and also the skills to keep up to date efficiently on newly discovered threats and vulnerabilities.

Role of the mainframe

The role of the mainframe in an enterprise-wide Threat and Vulnerability Management concept is Security Intelligence.

Threats on the mainframe are often detected, isolated, analyzed, and handled by mainframe-specific products. Consolidating mainframe security threats at the organizational level can help analyze trends, pinpoint sources, and lead to timely efficient enterprise-wide threat responses that minimize damage.

Sensitive personal and critical business data must be adequately protected, especially as we find new ways to use the information in cloud environments and big data analytics. Data can be protected at various levels, such as tables, files, or storage devices, and in various ways, such as access controls, masking, encryption, or network communications, by using ESMs or SAF APIs. Mainframes continue to offer strong security and privacy solution enhancements to protect data at rest and in motion.

Organizations are facing increasing challenges in meeting the requirements of regulations for IT security. This is related to the concept of knowing who users are, what applications and resources they are entitled to access, and what they did. ESMs on the mainframe provide strong internal security controls that are related to user identities and their accesses to meet these requirements. Thus, organizations can identify and remediate inappropriate access violations, and ensure that their IT assets are adequately protected.

Automated event analysis, alerts, and reports help you quickly find problems with attributes of resources, security settings, configuration changes, and potential security threats. In mainframes, these tasks can be done by using IBM Health Checker for z/OS, which provides a foundation to help simplify and automate the identification of potential configuration problems before they impact system availability, or by using external tools.

3.6 Statement of Integrity and certification levels

This section describes the different levels of certification with respect to security standards in the industry and how the mainframe platform delivers the confidence of being securable. For example, z/OS, from its beginnings as OS/MVT and later as MVS in the 1970s, was built on S/360, then S/370, and now System z, a robust hardware architecture that provides for the safe execution of multiple applications for multiple users on one single platform. Various components of the mainframe platform, such as hardware, OS, and software products, work together to provide this unprecedented level of securable environment for business-critical applications.

Certifications provide assurances about the security and integrity of a product, process, or person. They conform to ISO/IEC standards. The Common Criteria is recognized in every country that abides by the Common Criteria Recognition Arrangement (CCRA). The need for certifying security products or functions arises from the fact that many consumers lack the knowledge, expertise, or resources to judge whether the security levels of their products or environment is appropriate. Over the years, this has become an important issue because data, information, and applications within computer systems are critical for any organization. An organization has a reasonable requirement that information that is contained in their systems remains private, while still being available as needed, and is not subject to unauthorized modification.

The main objective of such an evaluation of the security products and functions is to support the procurement of IT products with security features that meet the consumer's expectation through a standard criteria. It also fosters development practices that are oriented towards such features that comply with those security standards to pass the evaluation, also providing a base for any other requirement that an organization or an auditor might have.

Evaluation standards and criteria were put in place in the early 1980s in different countries, such as the Trusted Computer System Evaluation Criteria (TCSEC) standard that is issued by the United States Government Department of Defense (DoD). Europe implemented its own evaluation standard, the Information Technology Security Evaluation Criteria (ITSEC), in 1990, and so did Canada with the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC). Although aimed at providing users of IT products with a fair and independent expert evaluation of the products' security-related functions, these standards diverged in their evaluation criteria and rating systems. Furthermore, the resulting evaluations were not standardized or internationally recognized.

In October 1998, after two years of intense negotiations, government organizations from the United States, Canada, France, Germany, and the United Kingdom signed a historic mutual recognition arrangement for Common Criteria-based evaluations. This arrangement, officially known as the Arrangement of the Recognition of Common Criteria Certificates in the field of IT Security, was a significant step forward for government and industry in the area of IT product and protection profile security evaluations. The partners in the arrangement share the following objectives in the area of Common Criteria-based evaluation of IT products and protection profiles:

- ▶ To help ensure that evaluations of IT products and protection profiles are performed to high and consistent standards and are seen to contribute significantly to confidence in the security of those products and profiles
- ▶ To increase the availability of evaluated, security-enhanced IT products and protection profiles for national use
- ▶ To eliminate duplicate evaluations of IT products and protection profiles
- ▶ To continuously improve the efficiency and cost-effectiveness of security evaluations and the certification/validation process for IT products and protection profiles

The purpose of this arrangement is to advance those objectives by bringing about a situation in which IT products and protection profiles that earn a Common Criteria certificate can be procured or used without the need for them to be evaluated and certified/validated again. It seeks to provide grounds for confidence in the reliability of the judgment on which the original certificate was based by declaring that the certification/validation body associated with a participant to the arrangement meets high and consistent standards. The arrangement specifies the conditions by which each participant accepts or recognizes the results of IT security evaluations and the associated certifications/validations conducted by other participants and to provide for other related cooperative activities.

The Common Criteria standards are today under the ISO 15408 set of standards.

For more information about the Common Criteria, including access to the reports for each level of certification that is achieved for a multitude of IBM products, go to the following website:

<http://www.commoncriteriaportal.org/>

3.6.1 Evaluation Assurance Levels

The scope of the Common Criteria evaluation addresses the implementation of security-related functions in a product, that is, the components or functions supporting data integrity and confidentiality, access control, and so on. The evaluation is independent of the level of implementation in the product, as it can be hardware, firmware, or software functions. The evaluation is also technology-neutral.

The Common Criteria evaluation process addresses the following items:

- ▶ The analysis of the manufacturer's processes and procedures with evidence that the procedures are applied during the product design, build, and release cycles
- ▶ Theoretical analysis of the target through its design and how the design matches requirements
- ▶ Verification of mathematical proofs, if any, produced during the design stage
- ▶ Theoretical vulnerability analysis
- ▶ Analysis of the product's user documentation
- ▶ Independent functional testing that might include penetration testing

The Common Criteria model uses a standardized set of security functions or the security functional classes, such as security audit, communications, Cryptographic Support, user data protection, identification and authentication, security management, privacy, protection of the trusted security functions, resource use, Target of Evaluation (TOE) access, and trusted path.

In the Common Criteria evaluation, *Evaluation Assurance Levels* (EAL) are the ratings that are given to the evaluated products using a *package* of criteria, which includes the requirements or *protection profiles*⁸, but also includes considerations about the context of the use of the product, and the depth and rigor of the development and maintenance processes. There are seven levels of EAL:

- ▶ EAL 1 - Functional test: The testing is performed without assistance from the product's development team.
- ▶ EAL 2 - Structural test: More aspects of the product and its development and manufacturing processes are looked at with the help of the product's developers.
- ▶ EAL 3 - Methodical test and check: The design of the product is looked at for appropriate security considerations. The depth of functional testing and examination of the processes is increased with respect to EAL 2.
- ▶ EAL 4 - Methodical design, test, and review: The analysis goes deeper than for EAL 3. An informal security policy model of the product is also requested.
- ▶ EAL 5 - Semiformal design and test: At this level, more stress is put on vulnerability analysis and testing, along with an assessment of the rigor of development practices.

⁸ For additional descriptions of the terms that are used in the model, see the Common Criteria evaluation documentation at <http://www.commoncriteriaportal.org/>.

- ▶ EAL 6 - Semiformally verified design and testing: even more vulnerability analysis and testing. The development process goes under a semi-formal examination.
- ▶ EAL 7 - Formally verified design and testing: This is the highest assurance level that can be achieved. High resistance to penetration is required from the product. There is also a requirement for extended test results, both by the product developers and by the independent organization.

3.6.2 Certification levels for the mainframe

Various components of the mainframe platform were certified at different EAL ratings in the past. It continues to get the highest levels of certification that is practically feasible, and has technology and standards on par with other platforms or even higher ratings for certain key functions and components.

At the time of writing, z/OS, which is a highly complex OS, is certified at the EAL 4+ rating. Although there are other UNIX based OSs at this level, the value and strength of mainframe security is established by the other functions and components that support the OS.

The PR/SM component, which has provided a base for a highly virtualized environment for mainframe computers for decades, is rated at EAL 5+, which is higher than any similar components on other platforms. Similarly, RACF for z/OS is rated at EAL 5+, the highest rating for a security management product for both mainframe and non-mainframe platforms. EAL5 is the highest commercial grade assurance level and exceeds what other commercial platforms offer.

z/VM V6.1 with RACF/VM is rated at EAL 4+ for the OSPP with Labeled Security, and z/VM V6.3 is undergoing evaluation for the same. The various Linux on System z implementations are all certified OSPP at EAL 4+ as well.

Similarly, various other software products and functions that run on mainframe platform are also rated at competitively higher EAL levels than their corresponding products or functions on other platforms. A detailed listing of the latest EAL ratings for all the certified products and functions can be found at the following website:

<http://www.commoncriteriaportal.org/products/>

Thus, the mainframe provides the confidence of being a highly securable environment for business-critical applications compared with any other platform.

3.6.3 Statement of integrity

Because of the experience that IBM gained from designing OSs in the 1960s, it understands early on the distinct concepts of *integrity* and *security*, and how important they would be to the future of computing. So, mainframe OSs, in particular MVS and VM, were designed so that formal integrity statements could be made about them. These integrity statements drew boundaries around the capabilities of the OSs and application programs running on them. It was recognized that there had to be a class of programs that could function and interact at the level of the OS, but also there had to be fixed restraints on the abilities of *normal* application programmers so that they did not disrupt the ability of the OS.

The ability to strictly classify these two sets of programs was defined and implemented in the first releases of MVS and subsequently a similar concept was applied to VM systems. This was formalized as the *statement of integrity*, which was issued publicly. It included a commitment from IBM to fix any issue that allowed those strict controls to be circumvented.

Therefore, IBM in the 1970s already had initiated its long-term commitment to system integrity within its OS designs and development practices.

The 1973 MVS statement of integrity has formed the basis for more than three decades of the MVS successors' industry leadership in system security. The fact that IBM was able to make such emphatic claims so early in the life of the MVS family of OSs, and has been able to maintain that claim for all the years since, speaks volumes for the stability and reliability of the hardware and software.

The z/OS "System Integrity" is defined as the inability of any program that is not authorized by a mechanism under the installation's control to circumvent or disable store or fetch protection, access a resource that is protected by the z/OS Security Server (RACF), or obtain control in an authorized state. The current z/OS statement of integrity can be found at the following website:

http://www.ibm.com/systems/z/os/zos/features/racf/zos_integrity_statement.html

With the statement of integrity, IBM also commits to address and resolve any system integrity problem that is reported.

Integrity of ISV/OEM products and user applications: It is valid to assume that the programs that IBM provides that must operate in an authorized state on z/OS have been scrutinized during their development phase to ensure that they do not expose the integrity of system operations. However, this might not be the same case for programs from other vendors. So, software vendors should provide certification or assurance that their authorized programs behave as specified when running in z/OS. Also, these programs should be examined or trusted by the installation for proper design and coding.

The z/OS statement of integrity also addresses the RACF ESM. Following the MVS statement of integrity, IBM wanted to ensure that the same levels of assurance could be given to applications running on VM, and also to OS guests running on VM, such as MVS. Hence, IBM developed a statement of integrity for VM/SP that was specific to VM. The current z/VM statement can be found at the following website:

<http://www.vm.ibm.com/security/zvminteg.html>

This statement is different because it relates to the ability of one guest OS to be prevented from interfering with the operation of another guest or of the VM hypervisor. However, the promise that is made by the VM statement of integrity is as important as the one made for MVS.

3.7 Conclusion

Businesses today rely on the mainframe to accomplish the following tasks:

- ▶ Perform large-scale transaction processing (thousands of transactions per second)
- ▶ Support thousands of users and application programs concurrently accessing numerous resources
- ▶ Manage terabytes of information in databases
- ▶ Handle large-bandwidth communication

With exemplary strengths in its core architecture, the mainframe competes and serves as the most securable platform for many organizations. The key functions that are built in to its hardware and software architecture displays the robust mechanism that can protect critical business applications. It has retained its technology at highly securable levels for decades, passing industry certifications and assurance.



Hardware components

This chapter explains the concepts behind the different hardware components of a mainframe environment. It also explains their functions and their applications. This chapter describes the mainframe processor architecture, different components within a central processor complex (CPC), the external interfaces for the mainframe, and other peripheral devices.

There are different types of processors within a processing unit (PU), such as the general-purpose processors (GPs), System Assist Processors (SAPs), and specialty processors, such as the Integrated Facility for Linux (IFL), Integrated Information Processor (zIIP), and Application Assist Processor (zAAP). This chapter describes the functioning of enterprise system connections (IBM ESCON), fiber connection (IBM FICON®), channels, and channel paths.

This chapter also provides information about the functioning of Processor Resource/System Manager (PR/SM) and the processor storage architecture. In addition, it also describes the Coupling Facility (CF) and timers.

Lastly, this chapter describes various external devices that interface with the mainframe, such as the storage or disk subsystems, tape libraries, terminal devices, and printers. This description also includes network interface devices such as the Open Systems Adapter (OSA) and the communication controllers.

In general, this chapter focuses the following areas for the hardware components:

- ▶ Their descriptions, architectures, functions, and applications.
- ▶ Their relationship with the overall mainframe architecture
- ▶ Their key functions and features that relate to the security concepts

4.1 System components

There is a casual notion that more than half of administrators or programmers who work on mainframe computers have not seen a mainframe or even know how it physically looks. This section describes the components within the mainframe *box*. There are various terms that are used to describe the mainframe hardware. As with a desktop, a personal computer, or a notebook, the physical mainframe hardware components are confined within a large box. Although there are certain variations for mainframe in terms of external storage devices and peripheral devices, which are outside of this 'box', key components such as the processor, memory, and other adapters are within it.

4.1.1 The complete picture

Traditionally, when the mainframe computers were huge in size, filling a large room or a complete data center, the entire computing system was distributed into physically separated components. The processing systems, storage systems, input/output systems, and other peripheral devices, such as printers or display units, were placed separately and were interconnected by cabling or networking mechanisms. The processing system was packaged in a box containing the PUs, main storage, channel adapters, and other key components. When there was only one PU, it was initially called by the name *central processing unit* (CPU). But, when mainframe computers grew in technology, complexity, and processing power, more PUs were added and were called a CPC. This term struck firmly as the physical mainframe box is called as CPC, even today. Although there are even references to call it a *central electronic complex* (CEC), the term CPC is preferred and is widely used. Another common term that is used to represent the CPC is a *footprint*, which can be seen in product announcement letters or on the Internet. The CPC is also sometimes called as a *frame*, referring to the physical structure within which the components are hosted.

Today, the CPC is much more advanced than the traditional mainframe computers. A CPC consists of plug-and-play processor units (PUs), hot-swappable memory cards, redundant array of I/O cages, an integrated battery feature with failover, and dual support elements (SEs) that are hosted within the box. The CPC is connected to other peripheral devices and external storage devices through high-speed fiber and network cabling. The hardware microcode controls the functioning of all the components within the CPC. With an energy-efficient footprint, the current day mainframe hardware wins greatly in server consolidation, and total cost of ownership (TCO).

Let us look at the *inside* of a CPC. Figure 4-1 on page 85 shows the IBM System zBC12 frame. Being the latest in the series, we use zBC12 as a reference for rest of this chapter.

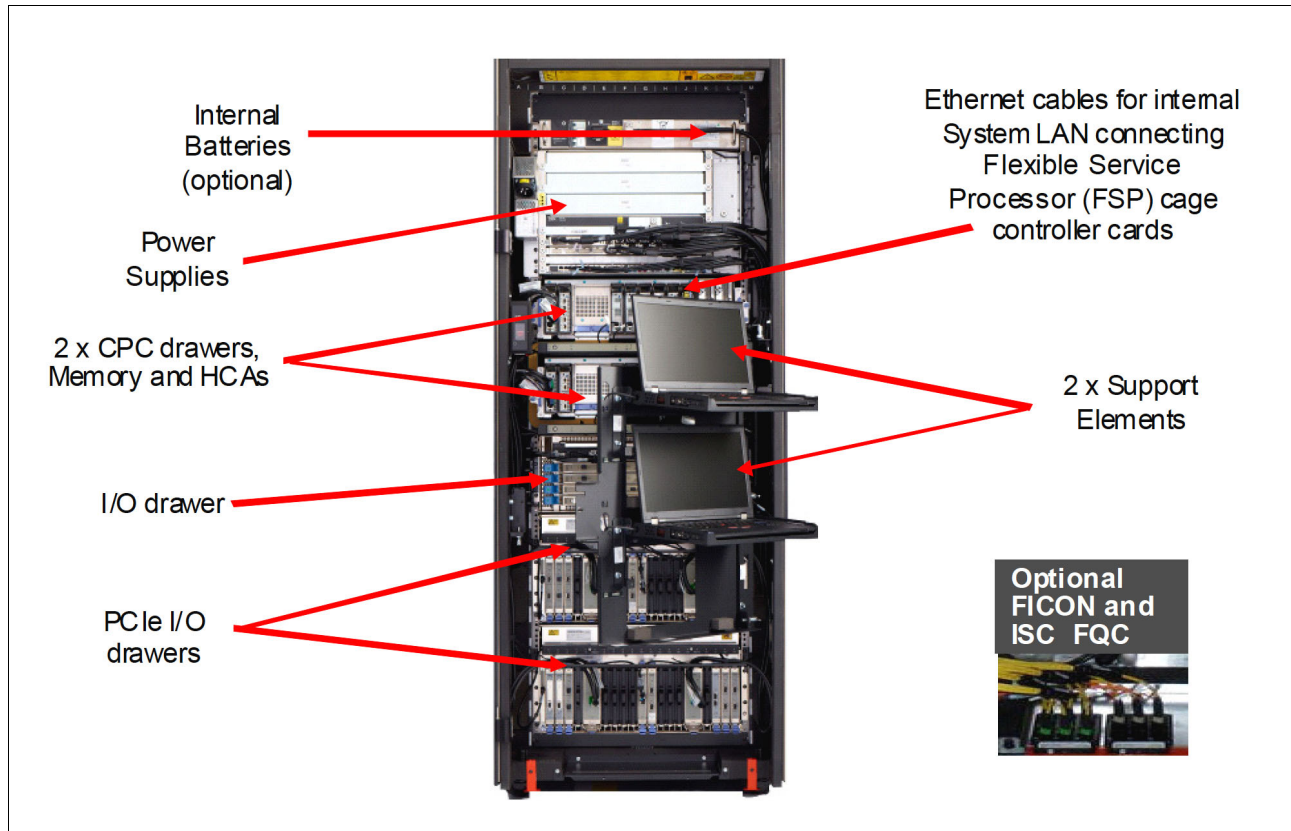


Figure 4-1 Front view of processor drawers and I/O drawers

Figure 4-1 shows the front (and rear) view of the zBC12 with two processor drawers, one I/O drawer, and two PCIe I/O drawers. There also are two SEs, power cages, and internal batteries. Each of these components is explained in the following sections.

4.1.2 Microcode

The *microcode* is the control logic for any computer. It contains the hardware level instructions and functions to control the operations of the electronics within the hardware. It is also known as *firmware* or machine code.

In the mainframe, the microcode also sometimes is called the Licensed Internal Code (LIC). The LIC is microcode, basic I/O system code, utility programs, device drivers, diagnostic tests, and any other code that is delivered with an IBM machine for enabling the machine's specified functions. The microcode is upgraded on the system through the SE by installing what is called a Microcode Change Level (MCL). As the physical mainframe hardware comes with a standard set of processors and features, the LIC of the mainframe controls the activation and configuration of what features an installation is eligible and paid for.

An improved microcode, called *the millicode*, was developed later. For the millicode to function effectively with its improved features, additional processor components were introduced, such as the special set of registers, that is, the general-purpose, access, and control registers (CRs), and the Program Status Word (PSW). These designs continue to evolve and improve in the latest generations of mainframe computers.

Processor Resource/System Manager

The hardware and firmware that provides partitioning is known as PR/SM. It is the PR/SM functions that are used to create and run LPARs. This difference between PR/SM (a built-in facility) and logical partitions (LPARs) (the result of using PR/SM) is often ignored and the term LPAR is used collectively for the facility and its results.

PR/SM is a type 1 hypervisor that is integrated with all IBM System z models that transforms physical resources into virtual resources so that many LPARs can share physical resources.

All System z models come equipped with PR/SM, the type 1 hypervisor that enables LPARs to share system resources. PR/SM can divide physical system resources (dedicated or shared) into isolated LPARs. Each LPAR operates as an independent system running its own operating environment. On the latest System z models, you can create up to 60 LPARs running z/VM, z/OS, z/OS.e, Linux, Transaction Processing Facility (TPF), or z/VSE on a single system. PR/SM enables each LPAR to have dedicated or shared processors and I/O, and dedicated memory (which you can dynamically reconfigure as needed). LPARs with dedicated resources own the resources to which they are assigned. LPARs with shared resources appear to own the resources to which they are assigned, but the resources are shared by many LPARs. PR/SM transforms physical resources into virtual resources so that many LPARs can share physical resources.

Input/Output Control Data Set

An Input/Output Control Data Set (IOCDS) is in the hard disk drive (HDD) of the SE within the CPC. It is created by the Input/Output Control Program (IOCP) from the z/OS source data set Input/Output Definition File (IODF) that is created by the Hardware Configuration Dialog (HCD) panels. The IOCDS, which is a flat file, defines the channels, control units (CUs), and devices to the designated LPARs within the channel subsystem (CSS), within the server. The IOCDS is loaded into the Hardware System Area (HSA) and initialized during Power-on Reset (POR) by the SE.

Power-on Reset

The hardware microcode is responsible for initializing and activating the hardware features and functions that are licensed for an installation. When the mainframe is powered on, the microcode initializes the basic functions within the hardware and boots up the SE, which acts as the user interface for further operations on the CPC. It still will not be ready for initializing and running an operating system until the system is loaded with a proper configuration and allocation of processors, channels, and I/O devices. This is accomplished by a task that is called a POR.

The POR activates the *reset* profile that is defined in the SE. Every CPC in the processor cluster needs a reset profile to determine the mode in which the CPC LIC is loaded and how much main storage and expanded storage is used. The POR also loads and initializes the HSA and loads the IOCDS into it. After it is activated, the CPC is in a mode that is ready for initializing an operating system, according to the mode and configuration that is used during activation.

4.1.3 Processor

The CPC uses a packaging concept for its processors that is based on *drawers*. A processor drawer contains the Single Chip Modules (SCMs), memory, and connectors to an I/O drawer, PCIe I/O drawer, and other CPCs. There can be more than one processor drawer within the CPC based on the hardware model and configuration.

There will be at least one Storage Control (SC) SCM in the drawer for every processor, even though the latest hardware models contain more than one SC SCM in a MultiChip Module (MCM). This SC SCM is the processor L4 cache with up to 192 MB. There is a L4 cache controller and four 48 MB quadrants of L4 cache in each SC SCM. An MCM usually has more than one PU chip, each with their own set of PUs, and one or two SC chips that are integrated in to a single module for better control and performance.

The PU SCM is the chip with up to six PUs or cores. Each core has their own L1 and L2 cache and there is one L3 cache per SCM, which interconnects all the six cores, GX I/O buses, and memory controllers (MCs) with the SC chips. The MC function controls access to memory. The GX I/O bus controls the interface to the fanouts accessing the I/O. The chip controls traffic between the cores, memory, I/O, and the L4 cache on the SC chip. There is also one dedicated co-processor (CoP) for data compression and encryption functions for each core. The compression unit is integrated with the CP assist for cryptographic function (CPACF), benefiting from combining (or sharing) the use of buffers and interfaces. The assist provides high-performance hardware encrypting and decrypting support for clear key operations.

Each PU, or a core, is a superscalar, out of program order processor, having the six execution units: two fixed point (integer), two load/store, one binary floating point, and one decimal floating point. When a mainframe order is configured, PUs are characterized according to their intended use and customer requirements. Some PUs are dedicated by the system to perform certain key functions, such as an SAP. The SAP is used to offload the I/O processing workload for efficient performance. Otherwise, the PUs can be configured as the following processors:

- ▶ A GP to run the normal operating system workloads, such as z/OS, z/VM, z/VSE, z/TPF, or Linux on System z
- ▶ An IFL to run Linux for System z workloads
- ▶ An Internal Coupling Facility (ICF) to run CF workloads
- ▶ An zAAP to run Java or XML workloads
- ▶ An zIIP to run IBM DB2 DRDA® or IPsec workloads

In each MCM, 12 - 16 available PUs may be characterized for customer use. Up to three SAPs may be in an MCM; how many are used depends on the model and the book in which they are. System-wide, two spare PUs (cores) are available that may be allocated on any MCM in the system. Up to two spare PUs (cores) may be allocated on an MCM.

Registers

The CP provides registers that are available to programs, but do not have addressable representations in main storage. They include the current PSW, the general registers (GRs), the floating-point registers (FPRs) and floating-point control (FPC) registers, the CRs, the access registers (ARs), the prefix register, and the registers for the clock comparator and the CP timer.

GRs are used to keep temporary data (operands) loaded from memory to be processed or already processed. Instructions may designate information in one or more of 16 GRs. The GRs may be used as base-address registers and index registers in address arithmetic, and as accumulators in general arithmetic and logical operations. Each register contains 64-bit positions. The GRs are identified by the numbers 0 - 15, and are designated by a 4-bit R field in an instruction.

The CP has 16 CRs of 64-bit positions for each CR, and is considered as the extension of PSW. The bit positions of some of the CRs are assigned to particular facilities in the system, such as program-event recording, and are used either to specify that an operation can take place, or to furnish special information that is required by the facility. The CRs are identified by the numbers 0 - 15 and are designated by 4-bit R fields in the instructions LOAD CONTROL and STORE CONTROL. CRs are registers that are accessed and modified by z/OS through privileged instructions. All the data that is contained in the CRs is designed to contain information input by z/OS and used by functions such as crypto, cross memory services, virtual storage, and clocks.

ARs are used by z/OS to access data spaces and other address spaces by activating the AR mode in the CP. The CP has 16 ARs that are numbered 0 - 15. An AR consists of 32-bit positions containing an indirect specification of an address-space-control element. An *address-space-control element* is a parameter that is used by the dynamic address translation (DAT) mechanism to translate references to a corresponding data space or address space.

Floating-point registers (FPRs) are used to keep temporary data (operands) loaded from memory to be processed or already processed. The CP has 16 FPRs. The FPRs are identified by the numbers 0 - 15 and are designated by a 4-bit R field in floating-point instructions. Each FPR is 64 bits long and can contain either a short (32-bit) or a long (64-bit) floating-point operand. There is also an FPC register, which is a 32-bit register to control the floating-point instructions execution. It contains mask bits, flag bits, a data exception code, and rounding-mode bits.

Program Status Word

The current PSW is a storage circuit within the CP. It contains information that is required for the running of the currently active program, that is, it contains the current state of a CP. It has 16 bytes (128 bits). The PSW includes the instruction address, condition code, and other information that is used to control instruction sequencing and to determine the state of the CP. The active or controlling PSW is called the *current* PSW. It governs the program currently being run.

There are some specific bits within the PSW that are significant for certain operations, such as the storage protection or supervisor mode, which are explained in “Storage protection” on page 54 and “Supervisor calls” on page 59. Bits 64 - 127 point to the storage address of the next instruction to be run by this CP. When an instruction is fetched from main storage, its length is automatically added to this field. Then, PSW points to the next instruction address. However, there are instructions as a BRANCH that might replace the contents of this field, pointing to the branched instruction. The format of the PSW and significance of each bit is explained in *z/Architecture Principles of Operation*, SA22-7832.

The CP has an interrupt capability, which permits it to switch rapidly to another program in response to exceptional conditions and external stimuli. When an interrupt occurs, the CP places the current PSW in an assigned storage location, called the old-PSW location, for the particular type of interrupt. The CP fetches a new PSW from a second assigned storage location. This new PSW determines the next program to be run by using the First Level Interrupt Handler (FLIH). When it has finished processing the interrupt, the program handling the interrupt may reload the old PSW, making it the current PSW again, so that the interrupted program can continue. There are six types of interrupt: restart, external, supervisor call, program check, machine check, and I/O. Each type has a distinct pair of old-PSW and new-PSW locations that are permanently assigned in real storage. For each type of interrupt, there is, in main storage, a trio of locations for old PSW, new PSW, and interrupt codes. These locations are kept in an 8 KB area, called the Prefix Storage Area (PSA), at the beginning of the real storage for each processor.

4.1.4 Storage

Main storage provides the system with directly addressable fast-access storage of data. Both data and programs must be loaded into main storage from input devices before they can be processed. Main storage may include one or more smaller faster access buffer storages, which are sometimes called *cache*. A cache is physically associated with a CPU or an I/O processor. The effects, except on performance, of the physical construction and use of distinct storage media are not observable by the program. Separate cache may be maintained for instructions and for data operands. Information within a cache is maintained in contiguous bytes on an integral boundary that is called a cache block or cache line (or line, for short). Main storage, central storage, and real storage are different terms for the same memory.

Addressing

For purposes of addressing main storage, three basic types of addresses are recognized: absolute, real, and virtual. The addresses are distinguished on the basis of the transformations that are applied to the address during a storage access. Address translation converts virtual to real, and prefixing converts real to absolute.

- ▶ An absolute address is the address that is assigned to a main-storage location. An absolute address is used for a storage access without any transformations that are performed on it. The CSS and all CPUs in the configuration refer to a shared main-storage location by using the same absolute address.
- ▶ A real address identifies a location in real storage. When a real address is used for an access to main storage, it is converted, by means of prefixing, to an absolute address. At any instant, there is one real-address to absolute address mapping for each CPU in the configuration. When a real address is used by a CPU to access main storage, it is converted to an absolute address by prefixing. The particular transformation is defined by the value in the prefix register for the CPU.
- ▶ A virtual address identifies a location in virtual storage. When a virtual address is used for an access to main storage, it is translated by means of DAT, either to a real address, which is then further converted by prefixing to an absolute address, or directly to an absolute address.

Hardware System Area

HSA is allocated in the lower portion of book 0 in main storage. HSA is a piece of main storage that does not belong to any specific processor. After a POR, it will consist of the following items:

- ▶ Millicode, which is a sort of subroutine from the microcode
- ▶ PR/SM object code
- ▶ Copy of Coupling Facility control code (CFCC) object code
- ▶ Logical processor definitions from IOCDS
- ▶ I/O configuration data from IOCDS, constituted mainly by UCWs

When a new configuration is dynamically activated, HSA is updated to reflect the new hardware I/O configuration definition.

4.1.5 Coupling Facility

CF enables high-performance multisystem data sharing. The CF contains one or more mainframe processors and a special licensed built-in operating system called CFCC. The information in the CF is in memory and a CF typically has a large memory. A CF can be a separate system or an LPAR can be used as a CF. The former is called an ICF, and the latter is called a stand-alone CF.

A typical stand-alone CF is an IBM 9674 S/390 CF. This is a special mainframe processor that runs only the CFCC and cannot run any other normal mainframe operating system, such as a z/OS or z/VM. Alternately, internal CF can be configured in all available models of mainframe today. They run CFCC in at least one of the images that are dedicated for the purpose in the configuration. There also can be dedicated processors that are designated to run the CFCC for isolation and performance. Both z/OS images and a CF might fail at the same time if there is a processor complex failure. In this situation, some structures cannot be rebuilt in another CF. In general, an internal CF should be used as backup for a production CF if you do not also use CF duplexing.

An important design aspect of a Parallel Sysplex is synchronizing the TOD clocks of multiple servers, which allows events occurring on different servers to be properly sequenced in time. As an example, when multiple servers update the same database and database reconstruction is necessary, all updates are required to be time stamped in proper sequence. In the past, a separate device that is known as the IBM Sysplex Timer was required to keep the TOD clocks of all participating servers synchronized with each other to within a small number of microseconds. It was dictated by the fastest possible passing of data from one server to another through the CF structure. Today's implementation uses the Server Time Protocol (STP), which is a server-wide facility that is implemented in the LIC. STP presents a single view of time to PR/SM, and is designed to provide the capability for multiple mainframe servers to maintain time synchronization with each other. It is the follow-up to the Sysplex Timer.

4.2 Devices

Mainframe computers once were spread across a large room with its various components attached together to the central processor. This concept, although it has passed through various advancements, is still true because the CPC must connect to other peripheral devices to provide a complete picture of *the mainframe*. Some of these devices, such as the communication adapters, might be attached within the CPC under the same box or frame. They have cables connecting them to the processor circuitry and also enjoy the redundancy capabilities of the CPC. There are still various other external devices that talk to the CPC through channel connectivity and their controllers. The devices include the storage devices, Hardware Management Consoles (HMCs), or even network-attached printers. This section provides an overview of those devices, their controllers, and connectivity mechanisms.

4.2.1 Channel connectivity

The CSS is a function that is formed by channels, subchannels, SAPs, and I/O devices that are attached through control units (CUs). The CSS contains *channels*, which are much simpler than an SAP, and can communicate with I/O CUs and manage the movement of data between main storage and the CUs. The channels can send channel commands from the memory to a CU, transfer data during read and write operations, receive the status at the end of operations, and receive sense information from CUs, such as detailed error information.

Within the CSS are *subchannels*. One subchannel is provided for and dedicated to each I/O device that is accessible to the CSS. Each subchannel provides information concerning the associated I/O device and its attachment to the CSS. The subchannel also provides information concerning I/O operations and other functions involving the associated I/O device. The subchannel is the means by which the CSS provides information about associated I/O devices to CPs, which obtain this information by running I/O instructions. The actual number of subchannels that are provided depends on the model and the configuration. In z/Architecture, the I/O operation is created when a privileged START SUBCHANNEL (SSCH) instruction is run by the input/output supervisor (IOS), which is a z/OS component that issues the instruction on behalf of a task. It finishes when an I/O interrupt is received by the CPU, forcing the running of the IOS component again.

A subchannel provides the logical representation of a device to the program and contains the information that is required for sustaining a single I/O operation. A subchannel is assigned for each device that is defined to the LPAR. An individual I/O device may be accessible to the CSS by as many as eight different *channel paths*, depending on the model and the configuration. The channel is accessed by the operating system through the one-byte *channel path identifier* (CHPID), as defined in z/Architecture. A subchannel is also called as a Unit Control Word (UCW) and is stored in the HSA.

To communicate with the I/O devices that are attached to the channels, both the operating system and the CSS must know the I/O configuration. An SAP is a PU that runs I/O LIC, or what is called I/O microcode. The SAP relieves z/OS of processing an I/O request operation. It does the scheduling of an I/O operation by finding an available channel path to the device and ensures that the I/O operation starts. SAP, however, is not in charge of the movement between main storage and the channel.

A CU provides the logical capabilities that are necessary to operate and control an I/O device and adapts the characteristics of each device so that it can respond to the standard form of control that is provided by the CSS. A CU can be housed separately, or it can be physically and logically integrated with the I/O device, the CSS, or within the server itself. Communication between the CU and the CSS takes place over a channel path. The CU accepts control signals from the CSS, controls the timing of data transfer over the channel path, and provides indications concerning the status of the device.

The *unit address* (UA) or *device address* is used to reference the device during the communication between a channel and the CU serving the device. The UA is two hex digits *00 - FF*, and is stored in the UCW as defined to HCD. It is transmitted over the I/O interface to the CU by the channel.

4.2.2 Adapters

I/O channels are components of the System z CSS and IBM z/Architecture. They provide a pipeline through which data is exchanged between systems, or between a system and external devices. z/Architecture channel connections are referred to as *channel paths*. The most common attachment to a z/Architecture channel is a CU accessed through an Enterprise Systems Connection (IBM ESCON) or Fibre Connection (FICON) channel. The CU controls I/O devices such as disk and tape drives. System-to-system communications are commonly implemented using InfiniBand (IFB) coupling links, Inter System Channels (ISC), Integrated Cluster Bus (ICB) channels, and channel-to-channel (CTC/FCTC) connections. Internal Coupling (IC) channel, IBM HiperSockets™, and channel-to-channel connections can be used for communications between LPARs within a system. The Open Systems Adapter (OSA) provides direct, industry-standard local area network (LAN) network connectivity and communications in a multivendor networking infrastructure.

I/O connectivity is implemented through several features, some of which implement the PCIe standard and are housed in PCIe I/O drawers, with others housed in either I/O drawers or I/O cages. In an I/O cage, features are installed in I/O card slots, with four I/O card slots forming an I/O domain. Each I/O domain uses an IFB-MP card in the I/O cage and a copper cable that is connected to a Host Channel Adapter (HCA-C) fanout in the processor cage. I/O drawers provide increased I/O granularity and capacity flexibility, as compared with the I/O cage.

Communication paths

This communication is managed by protocols such as ESCON, FICON, or FICON Express, which determines the channel type and the host adapter type in the I/O CU. However, independently of the channel type and the I/O CU host adapter type, the information that is needed by the channel to run the I/O operation (together with the I/O CU) is described in a channel program. Making an analogy, the CPU runs programs that are made of instructions, and the channel runs channel programs that are made of Channel Command Words (CCWs). The channel program is written by z/OS components such as VSAM, QSAM, BSAM, and BPAM.

IBM introduced the parallel channel with System/360 in 1964. The I/O devices were connected using two copper cables that are called *bus cables* and *tag cables*. A bus cable carries information (one byte each way), and a tag cable indicates the meaning of the data on the bus cable.

ESCON is a part of the Enterprise Systems Architecture/390 (ESA/390) and, by extension, of the z/Architecture. ESCON replaces the previous S/370 parallel OEMI with the ESCON I/O interface, supporting additional media and interface protocols. By replacing the previous bus and tag cables and their multiple data and control lines, ESCON provides half-duplex serial bit transmission, in which the communication protocol is implemented through sequences of special control characters and through formatted frames of characters. Since 1990, ESCON has replaced the parallel channel as being the main channel protocol for I/O connectivity, using fiber optic cables and a new “switched” technology for data traffic. An ESCON channel can reach up to 16 CUs (the CUA field has 4 bits), with 256 devices each (the UA has 8 bits). An ESCON channel runs commands that are presented by the standard z/Architecture or ESA/390 I/O command set, and it manages its associated link interface (link level/device level) to control bit transmission and reception over the optical medium (physical layer). On a write command, the channel retrieves data from main storage, encodes it, and packages it into device-level frames before sending it on the link. On a read command, the channel performs the opposite operation.

IBM announced a new I/O architecture for 9672 G5/G6 servers called FICON. The zSeries server builds on this architecture by offering higher speed FICON connectivity. FICON is widely used in the System z environment and provides additional strengths and capabilities compared to the IBM ESCON technology. Many additional capabilities have been included in support of FICON since it was originally introduced. Some CUs and CU functions might require FICON use exclusively. FICON supports up to 256 I/O logical CUs (8 bits).

Network connectivity

The Open Systems Adapter-Express (OSA-Express) features bring the strengths of System z servers, such as security, availability, enterprise-wide access to data, and systems management, to the client/server environment. OSA-Express2, OSA-Express3, and OSA-Express4S features are designed to provide direct, industry standard LAN connectivity in a multivendor networking infrastructure.

HiperSockets provide a seamless network to consolidate servers into an advanced infrastructure intra-server network. HiperSockets create multiple independent, integrated, virtual LANs within a System z system. The HiperSockets technology provides high-speed connectivity between combinations of LPARs or virtual servers within a System z server. The technology eliminates the need for any physical cabling or external networking connection between these virtual servers. This network within the box concept minimizes network latency and maximizes bandwidth capabilities between z/VM, Linux on System z, z/VSE, and z/OS images, or combinations of these items. HiperSockets are also possible under z/VM, which allows you to establish internal networks between guest operating systems, such as multiple Linux servers.

Coupling links

IC links are LIC-defined links to connect a CF to a z/OS LPAR in the same server. These links are available on all System z servers. The IC link is a System z server coupling connectivity option that enables high-speed, efficient communication between a CF partition and one or more z/OS LPARs that are running on the same server. The IC is a linkless connection (implemented in LIC) and so does not require any hardware or cabling. An IC link is a fast coupling link that uses memory-to-memory data transfers. IC links do not have PCHID numbers, but do require CHPIDs. IC links require a channel path definition at the z/OS and the CF end of a channel connection to operate in peer mode. They are always defined and connected in pairs. The IC link operates in peer mode and its existence is defined in HCD/IOCP.

Intersystem Channel-3 (ISC-3) provides the connectivity that is required for data sharing between the CF and the systems. These links also provide connectivity for STP messaging between any System z server. ISC-3 links support point-to-point connections (directly connecting CFs and systems) and require a unique channel definition at each end of the link. ICB links are members of the family of coupling link options available on IBM System z10® and previous System z servers. They are faster than ISC links, attaching directly to a Self-Timed Interconnect (STI) bus of the server. The ICB features are highly integrated, with few components, and provide better coupling efficiency (less server impact that is associated with coupling systems) than ISC-3 links. IFB coupling links are high-speed links on the latest System z servers.

4.2.3 Controllers

System z is a complex system that is integrated with various hardware and software components. There are various controllers that help manage these hardware and the operating systems.

Hardware management

The HMC is a hardware management instrument for System z. You use it to configure and manage servers, partitions, I/O channels, and other resources of multiple System z servers. It also provides system resource views and system administration tasks that include real-time monitoring, backup, and recovery. These management functions are provided for multiple servers through the SE through HMC communication.

The HMC communicates to each CPC through the SE. The IBM System z has an integrated SE that is inside of the same frame that the CPC is in. An alternative SE also is provided for the option to automatically fail over if a hardware problem occurs. These are locally attached notebooks that are fitted within the frame in latest models. The SE is used by service personnel to perform maintenance operations on the system. The SE is used for critical operations, such as upgrades to the LIC, and managing PR/SM configuration and system time.

The hardware can be managed by using either the SEs that are directly attached to the server or the HMC. Using the communication path through the SE, the HMC can perform numerous operations and tasks to assist in the maintenance and operations of the CPC, LPARs, I/O, and other physical and logical entities on the system. Working from an HMC, an Operator, System Programmer, or IBM technical personnel can perform basic operations on System z servers. Here are some of the common capabilities of the HMC:

- ▶ Load the System z hardware configuration.
- ▶ Load or reset a system image.
- ▶ Add and change the hardware configuration (most of them dynamically).
- ▶ Access the hardware logs.

All of these functions can be run by using a web interface in a secure environment. The HMC and the SE also act as master consoles that can communicate with the operating system. These consoles can be activated when there is no communication with the channel-attached or the system-defined master console, or any other consoles, and can be used for operations and diagnosis.

Console controllers

Traditionally, mainframe computers had channel-attached consoles that were connected over SNA and co-axial cables. These were units similar to dumb terminals but with some additional special characteristics. These console units were connected through the IBM 3174 Cluster controllers and could be activated during the early stages of IPL of z/OS or other operating systems that run on System z. Each LPAR requires one or more local 3270 devices/sessions for MVS consoles and a few TSO terminals. The local 3270 devices require a local IBM 3174 Control Unit for each LPAR. As the number of LPARs grows, the number of local 3174 CUs also grows. Lack of additional 3174 CUs can inhibit the use of additional LPARs for test and development purposes. The 3174 CUs were connected through parallel channels, but ESCON based controllers were available during later stages. IBM no longer produces these controllers, as these were replaced by less-complex CUs, such as the IBM 2074 CUs that were introduced in late 1990s.

The IBM 2074 Control Unit can be used with any operating system that supports local, non-SNA, DFT 3270 devices. The 2074 is a shared CU, internally running OS/2, that can be used simultaneously by multiple LPARs, using EMIF-capable channels. For example, a single 2074 might provide z/OS consoles (and TSO sessions) for 10 LPARs. The LPARs might be in a single z/OS or spread across multiple z/OS systems. The 2074 does not use existing “real” 3270 devices or connections. It does not use co-axial cables or any of the family of “real” 3270 terminals. Instead, it uses TCP/IP connections (TN3270e) over local network to connect terminals (usually PCs) to the 2074. It uses ESCON channels to connect the 2074 to System z channels, either directly or through ESCON directors. The display (and keyboard with integrated mouse) that is provided with the 2074 can be used for a limited number of 3270 sessions. IBM eNetwork Personal Communications is included with the 2074 to provide 3270 client sessions on the 2074 console. In the case of a single 2074, with all OS/390 consoles operating through it, a complete 2074 failure causes all the 3270 sessions to be lost. The 2074, even with the second ESCON adapter, does not support multiple paths to a 3270 session. The 2074 does not support channel multipathing at the CU level.

The Open Systems Adapter-Express Integrated Console Controller (OSA-ICC) function, the latest in the series of console controllers, supports TN3270 enhancements (TN3270E) and non-SNA DFT 3270 emulation. It can help reduce cost and complexity by eliminating the requirement for external console controllers, such as the IBM 2074 and IBM 3174. The OSA-ICC support is a no-charge function that is included in LIC on latest System z servers. It is available through the OSA-Express2 and OSA-Express 1000BASE-T Ethernet features, and supports Ethernet-attached TN3270E consoles. The OSA-ICC provides a system console function at IPL time and operating systems support for multiple LPARs. Console support can be used by z/OS, z/VM, z/VSE, and TPF. The OSA-ICC also supports 328x printer emulation for TSO/E, CICS, IMS, or any other 3270 application that communicates through VTAM.

Communication controllers

IBM developed communication controllers to serve two important networking needs:

- ▶ As gateways for connecting mainframe host servers to networks offloading network functions, such as line control and device support, thus saving host CPU cycles
- ▶ As line concentration devices that could be placed in remote locations to save on communications line charges by consolidating the traffic from many remote devices into a few higher speed communication lines

These were huge devices, namely the 374x series of controllers that served various purposes as a network gateway to mainframe computers for decades. As the computing environment moved to TCP/IP from SNA, the use of these devices became sparse and there was a need for a convergence between the applications and devices. To simplify the architecture of these massive controllers, which consumed much power and floor space, IBM introduced Enterprise Extenders (EE). These allow for the use of SNA transport protocols (namely APPN and HPR) over an Internet Protocol (IP) network. It enables the use of IP-based infrastructural network components for use in delivering SNA traffic. This leads to the consolidation of your network infrastructure into one strategic IP-based network.

4.2.4 External storage

External storage consists of auxiliary storage, such as disk devices, and removable storage devices, such as tape or virtual devices.

Disk storage devices

IBM 3390 disk drives are commonly used on current mainframes. The associated CU (3990) typically has four channels that are connected to one or more processors (probably with a switch), and the 3390 unit typically has eight or more disk drives. This storage was later succeeded by the IBM 2105 Enterprise Storage Server®. The 2105 unit is a sophisticated device. It emulates many CUs and 3390 disk drives. The Host Adapters appear as CU interfaces and can connect ESCON or FICON channels.

The physical disk drives are commodity SCSI-type units. A number of internal arrangements are possible, but the most common involves many RAID 5 arrays with hot spares. Practically everything in the unit has a spare or fallback unit. The internal processing to emulate 3990 CUs and 3390 disks is provided by four high-end RISC processors in two processor complexes; each complex can operate the total system. Internal batteries preserve transient data during short power failures. A separate console is used to configure and manage the unit. The 2105 offers many functions that are not available in real 3390 units, including IBM FlashCopy®, Extended Remote Copy, Concurrent Copy, Parallel Access Volumes, Multiple Allegiance, a huge cache, and so on.

A simple 3390 disk drive (with CU) has different technology than the 2105, but the basic architectural appearance to software is the same. This allows applications and system software that are written for 3390 disk drives to use the newer technology with no revisions. Maintaining application compatibility over long periods of technology change is an important characteristic of mainframes.

Lately, IBM has a wide range of product offerings that are based on open standards and share a common set of tools, interfaces, and innovative features. The IBM System Storage® DS8000® family is designed as a high-performance, high capacity, and resilient series of disk storage systems. The DS8000 offers high availability, multiplatform support, including System z, and simplified management tools to help provide a cost-effective path to an on-demand world. The DS8870 is the fifth-generation IBM high-end disk system in the DS8000 series. It is designed to support the most demanding business applications with its exceptional all-around performance and data throughput. The DS8000 architecture is server-based. Powerful POWER7 processor-based servers manage the cache to minimize disk I/Os to maximize performance and throughput.

Removable storage devices

When the necessity to archive and transport data arose, there came a need to transfer contents from the expensive magnetic disk storage devices to the less expensive and relatively slower tape media to meet this purpose. Mainframes had efficient tape drives and storage media for many decades. External media of 348x or 349x formats were commonly used for storing data from mainframe. These were low-capacity media, but their drives were comparatively faster during their time. The 3490 media could store about 800 MB of uncompressed data and were usable with the 3490 tape drives. The tape drives were either stand-alone or semi-automatic, where they could load a limited number of tapes that were mounted on to a queued rack.

These low-capacity media were later replaced by the 359x media and drives in mid 1990s. These media could store up to 6 GB of uncompressed data, which was far more than what their 349x predecessors could store. There were also different variants of media and drives in this series with many advanced capabilities and storage capacities.

Growth of data in the Internet world gave birth to automated tape libraries with more tape drives fixed to them and automated robots capable of picking the labeled tape media from their slots to load into the drives. As the later models of tape drives from different vendors could handle different formats of data, the libraries were flexible to be accessible not only by the mainframes, but also by the distributed platforms, in large data centers. For example, one of the latest variants, the IBM TS3500 Tape Library can manage up to 900 PB of data within a single library image, and can handle both LTO Ultrium and IBM 3592 tape drive families.

Virtual tape devices

To protect data effectively, deliver uninterrupted data access, address regulatory requirements, and archive business data, organizations need a solution that can manage both primary and backup data that is active, inactive, or even archived. As data centers and the data they store continue to grow, data protection often becomes more complex. This protection might include moving more primary data to tape, continuous backup, multiple metro and global sites, simplifying offsite recovery, disaster recovery verification, and many other objectives and processes to ensure an enterprise level of business continuance. This growth in complexity can in turn lead to higher data management impact and costs.

To achieve the capability of retaining data at various layers and levels, mainframe computers started to deal with a new set of devices that are called Virtual Tape Solutions (VTS). These devices emulate tape devices through the physical high-capacity disk volumes, but can be controlled through policies to manage data effectively between physical disk and tape media.

The operating system functions identify the differences in data at each migration level, which can be either active on the actual DASD device, which is stored and managed by the VTS at the migration level-1, or archived to physical tape at migration level-2. In practice, the VTS creates an additional layer between the physical disk and the tape media. The latest of the VTSs are the IBM TS7700 Virtualization Engines, which optimize data protection and business continuance for System z data. Through the use of virtualization and disk cache, the TS7700 family operates at disk speeds while maintaining compatibility with existing tape operations. They also optimize the volume addressability by providing the ability of about 4,000,000 virtual volumes and about 1500 virtual drives.

4.2.5 Terminals

Mainframes were traditionally centralized data processing systems. Before personal computers came into existence, programmers and operators, or even the general users of a mainframe, all connected to them using terminals. In specific, they were called *dumb terminals*, as they did not have their own processing capabilities and were used to connect to the centralized mainframe. They consisted of a display device and a keyboard, and were connected to the mainframe using the 3274 CUs and coaxial cables. They were mostly time-sharing systems that were connected in a token-ring network. These terminals were also known as *3270 terminals*, representing the data stream they used to communicate with the mainframe. These data streams were transferred as control blocks to the mainframe operating system. At some point after the introduction of 3270 terminals, 3270 protocol converters began to appear, which allowed ordinary ASCII terminals to access 3270 applications by masquerading as 3274 CUs to the mainframe and converting between 3270 data streams and the escape sequences of the particular ASCII terminal.

With the increase in the number of users and the shift of technology to IP-based applications and personal computers, emulation of 3270 terminals came into practice. Software applications were developed to emulate the 3270 data streams over a Internet Protocol network. These products were portable, easy to use, and also provided a variety of functions with their user interface. They also ran on different operating systems that were on modern day personal computers. IBM eNetwork Personal Communications is a good example of a widely used 3270 emulation client.

4.2.6 Printers

Mainframes process large amounts data for various business processes and applications. A large amount of data, such as forms and reports, are printed on paper. There were high-speed line printers in the past that solved these purposes. These printers were mostly channel attached and were local. Although they catered to the necessities during that time, as the technology leaped through boundaries, it demanded a similar advancement in printing as well. These channel-attached printers are connected through coaxial cables to their controllers similar to the terminals, possessed limitations in speed of printing and data transfer, and also lacked in data formats and printing flexibility that were available in small-scale printers that were used on distributed platforms. These printers gave way to more flexible and large-scale printing devices to cater to the requirements of mainframe computers, still compete in features and flexibility that were available on distributed systems.

The IBM 6400 line printers became more commonly used in the 1990s and can support printing over IP and are shareable with other distributed platforms as well. There was also a tremendous improvement with new protocols and standards for printing ECBDIC data formats from mainframe applications. The latest printers available today are capable of high-speed printing, supports any platform or data formats from mainframe to personal computers, are centrally manageable, and can print over long-distance networks. There are also competing products available from different vendors providing transparency and scalability to the mainframe customers.

4.3 Conclusion

The mainframe has a well-coordinated system that helps perform parallel task execution with perfect isolation at all its levels. It is now clear how the hardware components of the mainframe, starting from its processor design, the microcode, internal adapters, and external devices, blend well in coordinating with each other to achieve this objective.

It is clearly evident that the mainframe stands apart from other operating platforms with its robust architecture of hardware and their interconnectivity. Much of these concepts were prevalent since the first mainframe computers and the respective terminologies and architecture holds good even in modern day mainframes. This fact proves the maturity of design of this reliable computing platform and their adaptability towards technology growth. With this, there are similar core functions at the operating system and software level that complement the hardware design.



Software components

This chapter describes the concepts and functions of the mainframe operating systems. It describes z/OS, z/VM, Linux on System z, and (briefly) z/VSE and z/TPF.

This chapter explains z/OS internals and some of the z/OS key functions that contribute to its reliability, strength, flexibility, and security. These functions include authorized program facility (APF) and supervisor calls (SVCs), and functions such as the Systems Management Facility (SMF) and System Authorization Facility (SAF). This chapter describes some of the z/OS subsystems, such as the database systems, transaction servers, middleware systems, and other system support functions, such as the z/OS Health Checker.

This chapter provides information about the network-related functions and their security-oriented features, such as the Secure Sockets Layer (SSL), IP Security (IPSec), encryption, and firewalls. This chapter shows how these features are managed using software components such as the Integrated Cryptographic Service Facility (ICSF). This chapter also shows some of the monitoring tools and facilities that are available with z/OS.

This chapter briefly describes similar concepts for other major operating systems, such as z/VM and Linux on System z. You will see the characteristics of a typical application software on the mainframe environment.

In general, this chapter focuses the following areas for the software components:

- ▶ Concepts and functions of each operating system and its functions, subsystems, and applications
- ▶ How they interface with each other in a heterogeneous environment
- ▶ How they relate to the overall security of the mainframe and the enterprise

5.1 Operating systems

One great advantage of mainframe computers is their capability of running different operating systems on the same hardware. This is not the case with other distributed hardware platforms, which are designed for a specific operating system. The operating systems that can run on a mainframe vary from one another in terms of their principles and operation, but still manage to accommodate the hardware architecture they run upon. Some of these operating systems evolved out of specific functions or business needs and later took the form of a full-fledged operating system themselves. Others, such as Linux on System z, came out as a result of a college project, which became successful and then commercial.

5.1.1 z/OS

z/OS is an integrated enterprise server operating system. It incorporates into one product a leading-edge and open communications server, distributed data and file services, Parallel Sysplex system support, object-oriented programming, a distributed computer environment (DCE), and an open application interface. As such, it is uniquely suited to integrate today's heterogeneous and multivendor environments. Although the official product name of the operating system is z/OS, the former product name, MVS, is still used when describing any aspect of the operating system. By incorporating the base operating system, z/OS continues to build on the classic strengths of MVS: reliability, continuous availability features, and security. This provides a scalable system that supports massive transaction volumes and many users with high performance, advanced system and network management, security, and 24x7 availability.

z/OS services

z/OS, as an operating system, provides program management services that lets you create, load, modify, list, read, and copy executable programs. The z/OS system provides solutions for the following major areas:

- ▶ **Data management:** z/OS provides a set of functions to manage storage resources on the system, supports storage and retrieval of data on disk, optical, and tape devices, program management functions, and device management functions to define and control the operation of input and output storage devices. Distributed FileManager (DFM) supports access to remote data and storage resources.
- ▶ **Softcopy publications services:** These services improve productivity in systems installation and management.
- ▶ **Security services:** Security and Cryptographic Services are a set of products and features that is used to control access to resources, and to audit and manage the accesses with appropriate centralized or decentralized control. These services form the basis for all security services for traditional applications, UNIX applications, and distributed systems.
- ▶ **System management services:** The functions and features that are provided with z/OS allow robust control and automation of the basic processes of z/OS, thus increasing availability, improving productivity of system programmers, and providing a consistent approach for configuring z/OS components of products.
- ▶ **Network communication services:** z/OS enables world class TCP/IP and Systems Network Architecture (SNA) networking support, multivendor and multiplatform connectivity, connectivity to a broad set of users, and support for multiple protocols.
- ▶ **Applications enablement services:** These services provide a solid infrastructure in which you can build new applications, extend existing applications, and run OLTP and batch processes.

- ▶ **z/OS UNIX System Services:** z/OS contains the UNIX applications services (shell, utilities, and debugger) and the UNIX System Services (kernel and runtime environment). The shell and utilities provide the standard command-line interface that is familiar to interactive UNIX users. z/OS includes all the commands and utilities that are specified in X/OPEN XPG4.2. With IBM Language Environment, z/OS supports industry standards for C programming, shell and utilities, client/server applications, and most of the standards for thread management, thus allowing transparent data exchange and easy portability of applications in an open environment.
- ▶ **Distributed computing services:** These services are achieved by a set of features and functions. Network File System acts as a file server to workstations, personal computers, or other authorized systems in a Internet Protocol network. Remote files are mounted from the mainframe (z/OS) to appear as local directories and files on the client system. DCE enables data encryption standard (DES) algorithms and the commercial data masking facility (CDMF). Distributed File Services (DFS) SMB allows users to access data in a distributed environment across a wide range of IBM and non-IBM platforms. SMB can automatically handle the conversion between ASCII and EBCDIC.
- ▶ **E-business services:** The IBM HTTP Server provides for scalable, high performance web serving for critical e-business applications. It is exclusive to z/OS. This element was previously known as a base element of z/OS under the names IBM Lotus® Domino® Go, the Internet Connection Secure Server (ICSS), and the Internet Connection Server (ICS).
- ▶ **Print services:** Application output can be electronically distributed and printed or presented over the web.

z/OS base elements

The z/OS system consists of base elements that deliver essential operating functions, in addition to the services provided by the Base Control Program (BCP) functions, such as communications support, online access, host graphics, and online viewing of publications. The base elements are shown in Figure 5-1.

Alternate Library for REXX	ICKDSF
Base Control Program (BCP)	Integrated Security Services
Bulk Data Transfer base (BDT)	ISPF
BookManager Read	JES2
Common Information Model (CIM)	Language Environment
Communications Server	Library Server
Cryptographic Services	Metal C Runtime Library
DFSMSdfp	MICR/OCR
Distributed File Service	Network File System (NFS)
EREP	OSA/SF
ESCON Director Support	Run-Time Library Extensions
FFST	SMP/E
GDDM	TIOC
HCD	TSO/E
High Level Assembler (HLASM)	z/OS UNIX
IBM HTTP Server	3270 PC File Transfer Program
IBM Tivoli Directory Server for z/OS	

Figure 5-1 z/OS base elements

Shipped as part of the z/OS system are the base operating system, products, and features, for example, z/OS UNIX System Services and LAN services. In addition to these features, products such as TSO/E, ISPF, IBM Graphical Data Display Manager (IBM GDDM), and IBM BookManager® READ, which provide essential operating system functions, are included in the base OS and are called base elements. Certain base elements can be dynamically enabled and disabled because a client can choose to use a vendor product instead of IBM products.

The idea of the z/OS system is to have elements and features instead of licensed programs. This concept might be more easily explained by saying that z/OS consists of a collection of functions that are called base elements and optional elements. The optional elements (features) are either integrated or nonintegrated. These optional features, both integrated and nonintegrated, are also tested as part of the integration of the entire system.

5.1.2 z/VM

z/VM is a virtualization platform that has a native operating system (Conversational Monitor System (CMS)), but can host other operating systems, including Linux on System z and z/OS. z/VM can provide a highly flexible test and production environment.

The z/VM implementation of IBM virtualization technology provides the capability to run full-function operating systems, such as Linux on System z, z/OS, and others as “guests” of z/VM. z/VM supports 64-bit IBM z/Architecture guests and 31-bit IBM Enterprise Systems Architecture/390 guests. z/VM provides each user with an individual working environment known as a *virtual machine* (VM). The VM simulates the existence of a dedicated real machine, including processor functions, memory, networking, and input/output (I/O) resources. Operating systems and application programs can run in VMs as guests. For example, you can run multiple Linux and z/OS images on the same z/VM system that also is supporting various applications and users. As a result, development, testing, and production environments can share a single physical computer. A VM uses real hardware resources, but even with dedicated devices (such as a tape drive), the virtual address of the tape drive might be the same as the real address of the tape drive. Therefore, a VM only knows “virtual hardware” that might exist in the real world.

Figure 5-2 on page 103 shows the layout of the general z/VM environment.

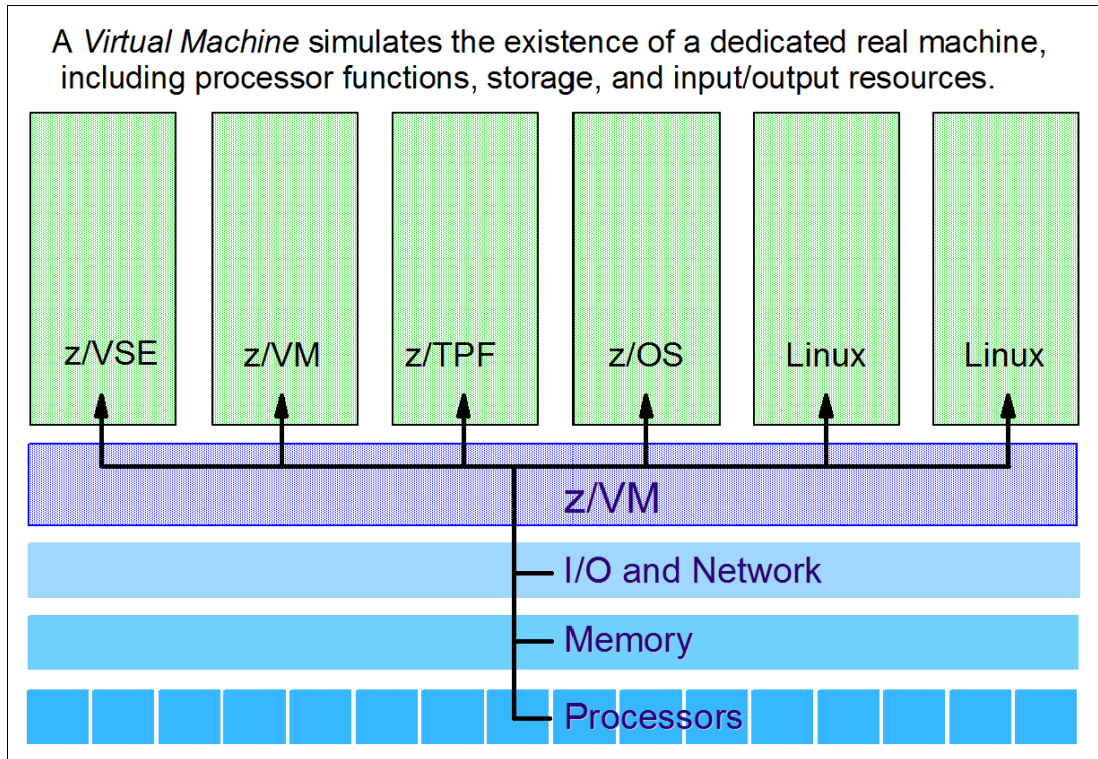


Figure 5-2 General z/VM environment

z/VM guest implementations are software-level partitioning. The z/VM operating system runs either on an LPAR or, on older hardware, on the complete mainframe (basic mode). Then, VMs are created on top of the z/VM system to host guest systems. A first-level z/VM is the base operating system that is installed on top of the real hardware. A second-level operating system is a system that is created upon the base z/VM operating system. Therefore, z/VM as a base operating system runs on the hardware, and a guest operating system runs on the virtualization technology. There is a first-level z/VM operating system that sits directly on the hardware, but the guests of this first-level z/VM system are virtualized. By virtualizing the hardware from the guests, you can create and use as many guests as needed with a small amount of hardware.

z/VM consists of the following components and facilities (base products):

- ▶ Control Program (CP)
- ▶ Conversational Monitor System (CMS)
- ▶ Transmission Control Protocol/Internet Protocol (TCP/IP) for z/VM
- ▶ Advanced Program-to-Program Communication/Virtual Machine (APPC/VM)
- ▶ Virtual Telecommunications Access Method (VTAM) Support (AVS)
- ▶ Dump Viewing Facility
- ▶ Group Control System (GCS)
- ▶ Hardware Configuration Definition (HCD) and Hardware Configuration
- ▶ Manager (HCM) for z/VM
- ▶ Language Environment
- ▶ Open Systems Adapter Support Facility (OSA/SF)
- ▶ Restructured Extended Executor/Virtual Machine (REXX/VM)
- ▶ Transparent Services Access Facility (TSAF)
- ▶ Virtual Machine Serviceability Enhancements Staged/Extended (VMSES/E)

CP is primarily a real-machine resource manager. CP provides each user with an individual working environment known as a VM. Each VM is a functional equivalent of a real system, sharing the real processor function, storage, console, and input/output (I/O) device resources. When you first log on to z/VM, CP controls the working environment. Many of the facilities of z/VM are immediately available to you. For example, you can use CP commands to do various system management tasks. However, most of the work that is done on z/VM requires the CMS or a guest operating system (such as z/OS) to help with data processing tasks and to manage work flow. CP provides connectivity support that allows application programs to exchange information with each other and to access resources on the same z/VM system or on different z/VM systems.

CMS provides a high-capacity environment that supports many interactive users. CMS can help you perform a wide variety of tasks. For example, you can write, test, and debug application programs for use on CMS or a guest system's base product, as listed here:

- ▶ Run application programs that are developed on CMS or guest systems
- ▶ Create and edit data files
- ▶ Process jobs in batch mode
- ▶ Share data between CMS and guest systems
- ▶ Communicate with other system users
- ▶ Provides a useful file system for storing data

5.1.3 Linux on z

Linux and System z make a great team. The Linux kernel is always the same, independent of the hardware platform on which it runs. The operating system is modular in design, which allows for ease of portability between system and processor architectures. It is open standards-based, supporting rapid application portability, and it can be quickly adapted to suit changing business needs. These features combine to provide Linux users with access to a large application base. Linux for System z can run natively on the System z hardware, or up to hundreds of virtual Linux servers can run simultaneously under z/VM, providing massive scalability within a single server, and unique server consolidation capabilities that reduce both cost and complexity.

Linux for System z supports the 64-bit architecture that is available on System z processors. This architecture eliminates the previous main storage limitation of 2 GB. Linux can run in a System z single image, a System z LPAR or a VM guest machine, and as a z/VM guest. A special feature of System z is the Integrated Facility for Linux (IFL), which is an IBM mainframe processor that is dedicated to running the Linux operating system, with or without z/VM. This optional feature provides a way to add processing capacity, exclusively for the Linux workload, with no limit on the System z model selected. Linux can also run on IFL engines natively, that is, in a System z LPAR.

Linux is not normally supplied with z/VM or with a System z mainframe. Linux for System z is available with such IBM available offerings as the IBM eServer Integrated Platform for e-business on System z and the Integrated Facility for Linux Engine (IFL) option. It is also available from third-party distributors.

5.1.4 z/VSE

z/Virtual Storage Extended (z/VSE) is popular with users of smaller mainframe computers. Some of these customers migrate to z/OS when they grow beyond the capabilities of z/VSE.

Compared to z/OS, the z/VSE operating system provides a smaller, less complex base for batch processing and transaction processing. The design and management structure of z/VSE is excellent for running routine production workloads consisting of multiple batch jobs (running in parallel) and extensive, traditional transaction processing. In practice, most z/VSE users also have the z/VM operating system and use this as a general terminal interface for z/VSE application development and system management.

z/VSE was originally known as Disk Operating System (DOS), and was the first disk-based operating system that was introduced for the System/360 mainframe computers. DOS was seen as a temporary measure until OS/360 was ready. However, some mainframe customers liked its simplicity (and small size) and decided to remain with it after OS/360 became available. DOS became known as DOS/VS (when it started using virtual storage), then VSE/SP and later VSE/ESA, and most recently z/VSE. The name VSE is often used collectively to refer to any of the more recent versions.

5.1.5 z/TPF

The z/Transaction Processing Facility (z/TPF) operating system is a special-purpose system that is used by companies with high transaction volume, such as credit card companies and airline reservation systems.

z/TPF was once known as Airline Control Program (ACP). It is still used by airlines and has been extended for other large systems with high-speed, high-volume transaction processing requirements.

z/TPF can use multiple mainframes in a loosely coupled environment to routinely handle tens of thousands of transactions per second, while experiencing uninterrupted availability that is measured in years. Large terminal networks, including special-protocol networks that are used by portions of the reservation industry, are common.

5.2 z/OS functions

As z/OS is predominantly used as the operating platform for mainframe workloads, this chapter focuses on the components and functions of z/OS to help understand how they contribute to the overall stability and security of the platform and its applications. These core functions of z/OS makes it the robust operating system and will help you understand why mainframe is still the most securable platform in existence.

5.2.1 Initial program load

An initial program load (IPL) provides a manual means for causing an executable program to be read (by the hardware) from a designated device to main storage and for initiating execution of that program. This executable program is the IPL. You use the HMC system console to order the load of such program. You must inform three entities: IPL device number (also called SYSRES volume), IODF device number, and LOADPARM options. This HMC console is connected to and managed by the processor controller (a notebook in the processor), which is also called a support element.

The IPL program loads the z/OS kernel and the Nucleus Initialization Program (NIP) from the data set SYS1.NUCLEUS, which is in the IPL device. Then, the CPU control is passed to NIP. During the NIP processing, NIP prompts the operator to provide system parameters that control the operation of MVS. The system also issues informational messages that inform the operator about the stages of the initialization process. The LOADxx parmlib member allows your installation to control the initialization process. For example, in LOADxx, you specify the suffixes for IEASYSxx or IEASYMxx members that the system uses; the system does not prompt the operator for system parameters that are specified in those members. Instead, it uses the values in those members.

The system initialization process prepares the system control program (z/OS) and its environment to do work for the installation. The process essentially consists of the following tasks:

- ▶ System and storage initialization (virtual, real, and auxiliary), including the creation of the common area and the system component address spaces
- ▶ Master scheduler initialization and subsystem initialization

When the system is initialized and the job entry (JES) subsystem is active, the installation can submit jobs and start other address spaces through the **START**, **LOGON**, or **MOUNT** commands. Also, z/OS creates system component address spaces. z/OS establishes an address space for the master scheduler (*MASTER* address space) and other system address spaces for various subsystems and system components.

When you start z/OS, master scheduler initialization routines initialize system services such as the system log and communications task, and start the master scheduler address space (*MASTER*). Each address space that is created has a number that is associated with it, known as the address space ID (ASID). Because the master scheduler is the first address space that is created in the system, it becomes address space number one (ASID=1). Other system address spaces are then started during the initialization process of z/OS. Next, subsystem address spaces are started. The master scheduler starts the job entry subsystem (JES2 or JES3). JES is the primary job entry subsystem. Then, other defined subsystems are started. All subsystems are defined in SYS1.PARMLIB, member IEFSSNxx. These subsystems are secondary subsystems.

5.2.2 Master scheduler

The master scheduler JCL data set (commonly called master JCL) controls system initialization and processing. It contains data definition (DD) statements for all system input and output data sets that are needed to communicate between the operating system and the primary job entry subsystem, which can be JES2 or JES3. To change system initialization or processing, you can change the information in the member or use an alternative data set. You might need to change the master JCL at IPL. For example, if you plan to use jobs as the source JCL for started tasks, modify the master JCL.

You can also modify the master scheduler JCL data set to include **START** commands for other subsystems, along with the DD statements that are necessary to communicate with them. You can also delete DD statements that do not apply to your installation's interactive configuration.

5.2.3 The link pack area

The Link Pack Area (LPA) is a common non-getmainable virtual storage area. It is built at IPL time by the loading of load modules that are contained in SYS1.LPALIB and its concatenations. There are two LPAs (one above and another below the 16-M line) depending on the residence mode of the load module. Because the load modules are in the common area, they are shared by all address spaces in the system. The load modules must be refreshable, meaning that they are not self-modifying. In this context, a reentrant load module means that it can be run concurrently by several tasks, but it cannot be temporarily modified. Then, each copy can be used by any number of tasks running in any number of CPUs in any number of address spaces at the same time. Modules that are found in LPA do not need to be brought into virtual storage on request because they are already in virtual storage. The benefit of having many load modules in LPA is that it decreases the use of the private areas. You can change this order and add other load libraries to the LNKST concatenation. The LNKST concatenation is established at IPL time and can optionally be modified dynamically. It consists of SYS1.LINKLIB, followed by the libraries that are specified in one or more LNKSTxx members of SYS1.PARMLIB. The LNKSTxx member is selected through the LNK parameter in the IEASYSxx member of SYS1.PARMLIB. The building of the LNKST concatenation happens during an early stage in the IPL process, before any user catalogs are accessible, so only those data sets whose catalog entries are in the system master catalog can be included in the linklist. To include in the LNKST a data set that is cataloged in a user catalog, you must specify both the name of the data set and the volume serial number (VOLSER) of the DASD volume on which the data set is.

5.2.4 System parameters

z/OS system parameters that are required for the initialization are in the parameter library. SYS1.PARMLIB is a required partitioned data set (PDS) that contains IBM supplied and installation-created members, with lists of system parameter values. You can include user-written system parameter members in parmlib before installing a product. Because SYS1.PARMLIB can be concatenated with other PDSs, we use in this book the name parmlib to refer to all the concatenated libraries and SYS1.PARMLIB when specific to a member. Parmlib is read by the system at IPL, and later by many components and subsystems that use parmlib members to implement dynamic changes in the system through operator commands, such as the following items:

- ▶ Workload Manager (WLM) with IEAOPTxx member
- ▶ Virtual Lookaside Facility (VLF) with COFVLFxx member
- ▶ Generalized Trace Facility (GTF)
- ▶ System Management Facility (SMF) with SMFPRMxx
- ▶ TIOC Terminal I/O Coordinator (TIOC) from TSO with the IKJPRM00 member, and many others

The purpose of parmlib is to provide many customization parameters in a pre-specified form in a single concatenated data set, and thus minimize the need for the operator to enter parameters. The parmlib data sets can be blocked and can have multiple extents, but must be on a single volume, like any PDS data set. A member within the parmlib, IEASYSxx, is the primary member that contains pointers to all the other configuration members in the parmlib.

There are several concatenated parmlibs through definitions in the LOAD member, another file that is used during system initialization. The LOAD member, usually in the SYSn.IPLPARM data set, contains information about the name of the IODF data set (which describes each z/OS I/O configuration), which Master catalog to use, and which IEASYSxx members of SYS1.PARMLIB to use. Another member that is named IEASYMxx specifies, for one or more z/OS OSs in a multisystem environment, the static system symbols that suffix IEASYSxx parameters that a specific z/OS system uses. These symbols allow a system to share, for example, the same SYS1.PARMLIB between z/OS systems by transforming generic parameters into specific ones for a single z/OS instance. One or more IEASYMxx members are selected using the IEASYM parameter in the LOADxx parmlib member.

5.2.5 Linklist

Program management is a z/OS component that deals with load modules. Here, you see how you can improve the performance of program management by using LNKLST and mechanisms that are used to locate and fetch load modules, such as LLA and VLF. Initially, you see that LNKLST allows the installation to concatenate load module libraries (PDS or PDSE) with SYS1.LINKLIB. Concatenation means that the access method processes all the concatenated data sets as though they were other extents of the first data set, that is, a continuation and not a different data set.

LLA is a program management z/OS component that keeps, in virtual storage, directory entries from load module libraries, thus speeding up the fetch of a load module. VLF is another z/OS component that keeps objects in virtual storage to save I/O operations that load such objects. One of these object types is load modules. When determining which data sets LLA and VLF are to manage, try to limit the choices to production load libraries that are rarely changed, which avoids a refresh of LLA or VLF operations. Because LLA manages LNKLST libraries by default, you need only to determine which non-LNKLST libraries LLA is to manage. If, for some reason, you do not want LLA to manage particular LNKLST libraries, you must explicitly remove such libraries from LLA management. To obtain the most performance benefit, use both LLA and VLF, which reduces the I/O that is required to find directory entries and to fetch load modules from DASD.

5.2.6 Job entry subsystem

z/OS uses a job entry subsystem (JES) to receive jobs into the operating system, schedule them for processing by MVS, and control their output processing. Job entry subsystem 2 (JES2) is descended from Houston automatic spooling priority (HASP). HASP is defined as a computer program that provides supplementary job management, data management, and task management functions, such as scheduling, control of job flow, and spooling. HASP remains within JES2 as the prefix of most module names and the prefix of all messages that are sent by JES2 to the operator.

JES2 is a functional extension of the HASP II program that receives jobs into the system and processes all output data that is produced by the job. So, what does all that mean? Simply stated, JES2 is that component of MVS that provides the necessary functions to get jobs into, and output out of, the MVS system. It is designed to provide efficient spooling, scheduling, and management facilities for the MVS operating system.

However, none of this explains why MVS needs a JES. Basically, by separating job processing into a number of tasks, MVS operates more efficiently. At any point in time, the computer system resources are busy processing the tasks for individual jobs, while other tasks are waiting for those resources to become available. In its most simple view, MVS divides the management of jobs and resources between the JES and the base control program of MVS. In this manner, JES2 manages jobs before and after running the program; the base control program manages them during processing.

IBM provides two JESs from which to choose: JES2 and JES3. In an installation that has only one processor (computer), JES2 and JES3 perform similar functions. They read jobs into the system, convert them to internal machine-readable form, select them for processing, process their output, and purge them from the system. However, for an installation that has more than one processor in a configuration, there are noticeable differences in how JES2 exercises independent control over its job processing functions. Within the configuration, each JES2 processor controls its own job input, job scheduling, and job output processing. In a sysplex environment, it is possible to configure JES2 to share spool and checkpoint data sets with other JES2 systems in the same sysplex. This configuration is called Multi-Access Spool (MAS).

In contrast, JES3 exercises centralized control over its processing functions through a single global JES3 processor. This global processor provides all job selection, scheduling, and device allocation functions for all the other JES3 systems. The centralized control that JES3 exercises provides increased job scheduling control, deadline scheduling capabilities, and increased control by providing its own device allocation.

5.2.7 Job Control Language

Job Control Language (JCL) is set of job control statements that are large, enabling you to provide much information to z/OS. Most jobs, however, can be run using a small subset of these control statements. After you become familiar with the characteristics of the jobs you run, you might find that you must know the details of only some of the control statements. Within each job, the control statements are grouped into job steps. A job step consists of all the control statements that are needed to run one process, for example, a sort, a copy, or an application program. If a job must run more than one process, the job contains another job step for each of those programs. A job can have 1 - 255 steps.

5.2.8 Procedures and tasks

Certain repeated tasks can be performed by using procedures in z/OS. These are similar to the concept of subroutines that are used in various programming languages. In z/OS, procedures are mostly in JCL, but can also be written in other languages, such as REXX or CLIST. Procedures can be called by other JCLs or programs to perform a specific task as applicable to the current runtime environment and can replace substituting parameters as input to establish specific conditions for execution. Procedures can be called directly by specifying the fully qualified name of their location, through special data definition statements within the calling program, or by placing them on certain libraries that are in the automatic search order by the system.

Started tasks on z/OS are long-running tasks that run certain system functions. These are similar to the concept of a process in distributed platforms. System functions and software products are initialized and made active by running their respective started tasks. For example, z/OS functions or products such as JES, TSO, VTAM, TCP/IP, DB2, CICS, and WebSphere have one or more started tasks for their components. These are JCL procedures that are present in the system procedure library, which is a data set that is in the default system search order for procedures. These procedures might contain symbolics that can resolve an input parameter to substitute in a data set name or any other configuration at the run time by the calling program or command. z/OS started tasks can be started by the operator from the console during the IPL and might need special security profiles to be identified as a recognized started task to preserve system integrity.

Users logging in to a TSO session also use a procedure to help run certain operations or commands that are necessary for easy navigation within their TSO or ISPF session. These operations or commands include concatenation of libraries to support the appropriate ISPF profiles or panels, issue TSO commands during logon, or to run REXX or CLIST programs. This is accomplished by use of logon procedures, which are similar to other procedures but got their name based on their purpose. External Security Managers (ESMs), such as RACF, associate a user profile with their authorized logon procedures to control access to system or application functions.

5.2.9 Time Sharing Option

TSO/E is a base element of z/OS. TSO/E allows users to share computer time and resources interactively. In general, TSO/E makes it easier for people with all levels of experience to interact with the z/OS system. TSO/E has undergone continuous enhancements during its lifecycle, and it became the primary user interface to the z/OS operating system. TSO/E provides programming services that you can use in system or application programs. These services consist of programs, macros, and CLISTs. TSO/E services support a wide range of functions that are useful in writing both system programs and application programs that use the full-screen capabilities of TSO/E. CLISTs, REXX execs, servers, and command processors are specific types of programs that you can write to run in the TSO/E environment.

TSO/E offers advantages to a wide range of computer users, including system programmers, application programmers, information center administrators, information center users, TSO/E administrators, and others who access applications that run under TSO/E.

5.2.10 ISPF

The Interactive System Productivity Facility/Program Development Facility (ISPF/PDF) is a set of panels that help you manage libraries of information about the z/OS system. The libraries are made up of units that are called data sets that can be stored and retrieved. You can have various kinds of information in data sets. ISPF is a multifaceted development tool set for the z/OS operating system. Since 1975, MVS programmers have used ISPF for host-based application development productivity. ISPF forms the basis of many TSO and CMS applications and provides extensive programmer-oriented facilities as well. ISPF can be used in many ways, as these examples illustrate:

- ▶ Users can edit, browse, and print data.
- ▶ Data processing administrators and system programmers can use ISPF to do the following tasks:
 - Monitor and control program libraries.
 - Communicate with MVS through TSO commands, CLISTs, or REXX EXECs.

- ▶ Programmers can use ISPF to develop a batch, interactive, or any other type of program and its documentation.
- ▶ Terminal users can invoke a wide range of utilities, such as search, compare, and compilers.

ISPF helps programmers develop interactive applications that are called dialogs. Dialogs are interactive because ISPF uses them to communicate with terminal users through a series of panels while the users do application development tasks.

5.2.11 Data sets

A data set is a collection of logically related data; it can be a source program, a library of macros, or a file of data records that is used by a processing program. Data records are the basic unit of information that is used by a processing program. By placing your data into volumes of organized data sets, you can save and process the data efficiently. You also can print the contents of a data set, or display the contents on a terminal. You can store data on auxiliary storage devices:

- ▶ A direct access storage device (DASD): The term DASD applies to disks or to a large amount of magnetic storage media on which a computer stores data. A volume is a standard unit of auxiliary storage. You can store all types of data sets on DASD. Each block of data on a DASD volume has a distinct location and a unique address, thus making it possible to find any record without an extensive search. You can store and retrieve records either directly or sequentially. Use DASD volumes for storing data and executable programs, including the operating system itself, and for temporary working storage. You can use one DASD volume for many separate data sets, and reallocate or reuse space on the volume.
- ▶ A magnetic tape volume: Only sequential data sets can be stored on magnetic tape. Mountable tape volumes can be in an automated tape library. For information about magnetic tape volumes, see *z/OS DFSMS: Using Magnetic Tapes*, SC26-7412. You can also direct a sequential data set to or from spool, a UNIX file, a TSO/E terminal, a unit record device, virtual I/O (VIO), or a dummy data set.

The Storage Management Subsystem (SMS) is an operating environment that automates the management of storage. Storage management uses the values that are provided at allocation time to determine, for example, on which volume to place your data set, and how many tracks to allocate for it. Storage management also manages tape data sets on mountable volumes that are in an automated tape library. With SMS, users can allocate data sets more easily.

An access method is a DFSMSdfp component that defines the technique that is used to store and retrieve data. Access methods have their own data set structures to organize data, macros to define and process data sets, and utility programs to process data sets. Access methods are identified primarily by the way that they organize the data in the data set. For example, use the basic sequential access method (BSAM) or queued sequential access method (QSAM) with sequential data sets. However, there are times when an access method identified with one organization can be used to process a data set organized in another manner. For example, a sequential data set (not extended-format data set) created using BSAM can be processed by the basic direct-access method (BDAM), and vice versa. Another example is UNIX files, which you can process using BSAM, QSAM, basic partitioned access method (BPAM), or Virtual Storage Access Method (VSAM).

VSAM data sets

VSAM arranges records by an index key, by a relative byte address, or by a relative record number. VSAM data sets are cataloged for easy retrieval.

- ▶ Key-sequenced data set (KSDS): A KSDS VSAM data set contains records in order by a key field and can be accessed by the key or by a relative byte address. The key contains a unique value, such as an employee number or part number.
- ▶ Entry-sequenced data set (ESDS): An ESDS VSAM data set contains records in the order in which they were entered and can be accessed only by relative byte address. An ESDS is similar to a sequential data set.
- ▶ Relative-record data set (RRDS): An RRDS VSAM data set contains records in order by relative-record number and can be accessed only by this number. Relative records can be fixed length or variable length. VRRDS is a type of RRDS where the logical records can be variable.
- ▶ Linear data set (LDS): An LDS VSAM data set contains data that can be accessed as byte-addressable strings in virtual storage. A linear data set does not have embedded control information that other VSAM data sets hold.

Non-VSAM data sets

Non-VSAM data sets are collections of fixed-length or variable-length records that are grouped into physical blocks (a set of logical records). To access non-VSAM data sets, an application can use BSAM, QSAM, or BPAM. There are several types of non-VSAM data sets:

- ▶ Physical sequential data set (PS): Sequential data sets contain logical records that are stored in physical order. New records are appended to the end of the data set. You can specify a sequential data set in extended format or not.
- ▶ Partitioned data set (PDS): Partitioned data sets contain a directory of sequentially organized members, each of which can contain a program or data. After opening the data set, you can retrieve any individual member without searching the entire data set.
- ▶ Partitioned data set extended (PDSE): Partitioned data sets extended contain an indexed, expandable directory of sequentially organized members, each of which can contain a program or data. You can use a PDSE instead of a PDS. The main advantage of using a PDSE over a PDS is that a PDSE automatically reclaims the space that is released by a previous member deletion, without the need for a reorganization.

z/OS UNIX data sets

z/OS UNIX System Services (z/OS UNIX) enable z/OS to access UNIX files. UNIX applications also can access z/OS data sets. z/OS UNIX files are byte-oriented, similar to objects. Here are the types of z/OS UNIX files:

- ▶ Hierarchical file system (HFS): You can define on DASD an HFS data set on the z/OS system. Each HFS data set contains a hierarchical file system. Each hierarchical file system is structured like a tree with subtrees, and consists of directories and all their related files.
- ▶ z/OS Network File System (z/OS NFS): z/OS NFS is a distributed file system that enables users to access UNIX files and directories that are on remote computers as though they were local. z/OS NFS is independent of machine types, operating systems, and network architectures.
- ▶ zSeries File System (zFS): A zFS is a UNIX file system that contains one or more file systems in a VSAM linear data set. zFS is application-compatible with HFS and more performance efficient than HFS.
- ▶ Temporary file system (TFS): A TFS is stored in memory and delivers high-speed I/O. A systems programmer can use a TFS for storing temporary files.

Generation data group

Generation data group (GDG) is a catalog function that makes it easier to process data sets with the same type of data but in different update levels. For example, the same data set is produced every day but with different data. Then, you can catalog successive updates or generations of these related data sets. They are called generation data groups (GDG). Each data set within a GDG is called a generation data set (GDS) or generation. Within a GDG, the generations can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set.

Generation data sets can be sequential, PDSs, or direct (BDAM). Generation data sets cannot be PDSEs, UNIX files, or VSAM data sets. The same GDG may contain SMS and non-SMS data sets.

5.2.12 Catalog

A catalog is a z/OS VSAM data set that contains information about other data sets, such as the volume serial (the name of the volume containing the data set). It provides users with the ability to locate a data set by name, without knowing where the data set is. By cataloging data sets, your users do not need to know as much about your DASD storage setup. Thus, data can be moved from one device to another, without requiring a change in JCL DD statements that refer to an existing data set.

Cataloging data sets also simplifies backup and recovery procedures. Catalogs are the central information point for VSAM data sets; all VSAM data sets must be cataloged. In addition, all SMS-managed data sets (data sets whose performance and availability are managed by the SMS system component) also must be cataloged.

Functionally, the catalogs are divided into master (mastercat) and user catalogs (usercat). Mastercat has z/OS system data set information and pointers to usercats based on the data set name high-level qualifier of the data set being located. Usercats contain non-z/OS data set information. For each high-level qualifier associated with a specific usercat, the installation defines one alias in the mastercat pointing to the usercat. Aliases are used to tell catalog management which user catalog in which your data set is cataloged. You can augment the standard catalog search order by defining multilevel catalog aliases. There is no physical difference between a mastercat and usercats.

5.2.13 Input/output processing

Here is the flow of an I/O operation from the request that is issued by an application until the operation completes:

1. The user program grants access to the data set by issuing an OPEN macro, which invokes the Open routine through an SVC instruction. Later, to request either the input or output of data, the program uses an I/O macro such as GET, PUT, READ, or WRITE, and specifies a target I/O device. These macros generate a branch instruction, invoking an access method. An access method has the following functions:
 - Writes the channel program (with virtual addresses)
 - Implements buffering
 - Ensures synchronization
 - Runs I/O recovery

The user program can bypass the access method, but it then must consider many details of the I/O operation, such as the physical characteristics of the device.

2. There are several z/OS access methods, such as VSAM, BSAM, and BPAM, each of which offers differing functions to the user program. The selection of an access method depends on how the program plans to access the data (randomly or sequentially, for example).
3. To request the movement of data, the access method presents information about the operation to an I/O driver routine (usually the EXCP driver) by issuing the EXCP macro, which expands into an SVC 0 instruction. The I/O driver translates the channel program from virtual to real (a format that is acceptable by the channel subsystem), fixes the pages containing the CCWs and the data buffers, ensures the volume extents, and invokes the I/O Supervisor (IOS).
4. IOS, if there is no pending I/O operation for the required device (originated previously in this system), issues the Start Subchannel (SSCH) instruction to the channel subsystem. Then, the CPU continues with other work (the task running the access method is probably placed in wait state) until the channel subsystem indicates with an I/O interrupt that the I/O operation has completed. If the device is busy, the request is queued in the UCB control block.
5. The SAP selects a channel path to initiate the I/O operation. This channel runs the channel program controlling the movement of data between device, control unit, and main storage.
6. When the I/O operation is complete, SAP signals the completion by generating an I/O interrupt towards any I/O-enabled CPU.
7. IOS processes the interrupt by determining the status of the I/O operation (successful or otherwise) from the channel subsystem. IOS reads the IRB to memory by issuing the TSCH instruction. IOS indicates that I/O is complete by posting the waiting task and calling the dispatcher.
8. When appropriate, the dispatcher dispatches the task returning to the access method code.
9. The access method returns control to the user program, which can then continue its processing.

5.2.14 Storage management

Data on the mainframe is managed by a separate component within z/OS called the Data Facility Storage Management Subsystem (DFSMS).

DFSMS is composed of a suite of related data and storage management products for the z/OS system. DFSMS is now an integral element of the z/OS operating system. DFSMS is an operating environment that helps automate and centralize the management of storage that is based on the policies that your installation defines for availability, performance, space, and security.

The heart of DFSMS is the Storage Management Subsystem (SMS). Using SMS, the storage administrator defines policies that automate the management of storage and hardware devices. These policies describe data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements for the system. DFSMS is an exclusive element of the z/OS operating system and is a software suite that automatically manages data from creation to expiration. The following elements makeup DFSMS:

- ▶ DFSMSdfp: Provides storage, data, program, and device management. It is composed of components such as access methods, OPEN/CLOSE/EOV routines, catalog management, DADSM (DASD space control), utilities, IDCAMS, SMS, NFS, ISMF, and other functions.

- ▶ DFSMSdss: Provides data movement, copy, backup, and space management functions.
- ▶ DFSMShsm: Provides backup, recovery, migration, and space management functions. It invokes DFSMSdss for certain of its functions.
- ▶ DFSMSrmm: Provides management functions for removable media, such as tape cartridges and optical media.
- ▶ DFSMStvs: Enables batch jobs and CICS online transactions to update shared VSAM data sets concurrently.

In a system-managed storage environment, DFSMS automates and centralizes storage management that is based on the policies that your installation defines for availability, performance, space, and security. With the optional features enabled, you can take full advantage of all the functions that DFSMS offers.

5.2.15 SDSF

SDSF is a licensed program that helps authorized users efficiently monitor and control the operation of a JES2 or JES3 subsystem. With SDSF, you can authorize your JES users to perform the following tasks:

- ▶ Control job processing (hold, release, cancel, and purge jobs)
- ▶ Monitor jobs while they are being processed
- ▶ Display job output before deciding to print it
- ▶ Manage the system's workflow
- ▶ Control the order in which jobs are processed
- ▶ Determine the number of output jobs and the total number of records to be printed
- ▶ Control the order in which output is printed
- ▶ Control printers and initiators
- ▶ View the system log online and use commands to search for specific information in the log
- ▶ Dynamically change job data set output descriptors
- ▶ Issue JES and MVS commands that affect their jobs
- ▶ Print selected lines of the JES output data set
- ▶ Edit JCL direct from spool

SDSF consists of panels that provide immediate information about jobs, printers, queues, and resources in a JES2 system. From SDSF panels, authorized users can enter commands to control the processing of jobs and the operation of system resources. Authorized users also can issue MVS and JES2 system commands from the SDSF panels. SDSF provides an easy way to manage JES2 jobs, which can help you work more efficiently.

5.2.16 System Management Facility

The z/OS system collects statistical data for each task when certain events occur in the life of the task. The System Management Facility (SMF) formats the information that it gathers into system-related (or job-related) records. System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information about the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session. SMF data is written to the SYS1.MANx data sets.

The volume and variety of information in the SMF records enables installations to produce many types of analysis reports and summary reports. SMF formats the information that it gathers into system-related records or job-related records in the following ways:

- ▶ System-related SMF records include information about the configuration, paging activity, and workload.
- ▶ Job-related records include information about the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session.

SMF reports data that installations can use as a basis for billing algorithms and reports. SMF produces records at IPL time and at system shutdown. SMF generates records that describe changes in the system configuration, such as at IPL for online devices, when devices are added or removed, or when a processor, channel path, or storage device moves online or offline. Using SMF collected data, you can identify specific intervals when the problem program use of system resources is at a high or low level. SMF reports information about problem-program use of direct-access volumes. SMF produces several records that contain information about data set activity. Most SMF records contain general identification fields, such as the job name, step name, programmer name, reader start time, and reader start date. RACF provides options that allow your installation to control access to resources and, through SMF records, audit your security controls.

It is important to describe information that can be obtained by reviewing the SMF records that create a database that can be used to understand trends and enable you to prepare for future workload and threshold problems. The volume and variety of information in the SMF records enables installations to produce many types of analysis reports and summary reports. For example, by keeping historical SMF data and studying its trends, an installation can evaluate changes in the configuration, workload, or job scheduling procedures. Similarly, SMF data can be used to determine the system resources that are wasted because of problems, such as inefficient operational procedures or programming conventions.

5.2.17 Resource Measurement Facility

RMF is the strategic IBM product for performance analysis, capacity planning, and problem determination in a z/OS host environment. Many different activities are required to keep your system running smoothly, and to provide the best service based on the available resources and workload requirements. This work is done by system operators, administrators, programmers, or performance analysts. RMF produces reports about problems as they occur, so that an installation can act before the problems become critical. An installation can use RMF to complete the following tasks:

- ▶ Determine that a system is running smoothly
- ▶ Detect system bottlenecks that are caused by resource contention
- ▶ Evaluate the service that an installation provides to various groups of users
- ▶ Identify the workload that is delayed, and the reason for the delay
- ▶ Monitor system failures, system stalls, and failures of selected applications

RMF comes with three monitors, Monitor I, Monitor II, and III. Monitor III provides short-term data collection and online reports for continuous monitoring of system status and solving performance problems. Monitor I provides long-term data collection for system workload and resource use. The Monitor I session is continuous, and measures various areas of system activity over a long period. Monitor II provides online measurements on demand for use in solving immediate problems. A Monitor II session can be regarded as a snapshot session. RMF Spreadsheet Reporter can be used to further process the measurement data on a workstation with the help of spreadsheet applications.

5.2.18 System logger

System logger is a set of services that allows an application to write, browse, and delete log data. You can use system logger services to merge data from multiple instances of an application, including merging data from different systems across a sysplex.

For example, suppose that you are concurrently running multiple instances of an application in a sysplex, and each application instance can update a common database. It is important for your installation to maintain a common log of all updates to the database from across the sysplex, so that if the database should be damaged, it can be restored from the backup copy. You can merge the log data from applications across the sysplex into a log stream, which is simply a collection of data in log blocks in the coupling facility and on DASD.

The system logger component is in its own address space on each system in a sysplex. Some of the component processing differs, depending on whether a given log stream is a coupling facility log stream or a DASD-only log stream. The system logger component completes the following tasks:

- ▶ Provides a set of system services that allows a system logger application to use the system logger component
- ▶ Maintains information in the LOGR policy about the current use of log streams, and if used, coupling facility list structures
- ▶ Interacts with cross-system extended services (XES) to connect to and use the coupling facility for system logger applications
- ▶ Obtains local storage buffer space
- ▶ Offloads data to DASD log data sets
- ▶ Automatically allocates new DASD log data sets for log streams
- ▶ Maintains a backup copy of (duplexes) log data that is in interim storage for recovery
- ▶ Produces SMF records for system logger accounting on a single system
- ▶ Provides recovery support in the event of application, system, sysplex, or coupling facility structure failure

5.2.19 UNIX System Services

Beginning with OS/390 V2R4, z/OS UNIX System Services were merged with the z/OS base component. The OMVS address space, which is the kernel of z/OS UNIX, is started automatically at IPL. The kernel address space contains the MVS support for z/OS UNIX System Services. This address space can also be called the kernel and is the part of z/OS UNIX that contains programs for such tasks as I/O, management, control of hardware, and the scheduling of user tasks. UNIX System Services Application Services provides the following functions:

- ▶ A TSO/E command to enter the shell environment
- ▶ A shell environment for developing and running applications
- ▶ Utilities to administer and develop in a UNIX environment

5.2.20 Software delivery

Software products on mainframe, including the operating system, are available for customers in many customized methods of delivery. These different methods help the customers to plan their installation according to their business needs and resource availability. Software management within the z/OS is performed through a component called as SMP/E, which is explained in 5.2.21, “SMP/E” on page 119.

ServerPac

ServerPac is an entitled software delivery package consisting of products and service for which IBM has performed the SMP/E installation steps and some of the post-SMP/E installation steps. To install the package on your system and complete the installation of the software it includes, use the CustomPac Installation Dialog. For ServerPac orders, service is integrated with product code.

CBPDO

Custom-Built Product Delivery Option (CBPDO) is an entitled software delivery package consisting of uninstalled products and unintegrated service. There is no dialog program to help you install, as there is with ServerPac. You must use SMP/E to install the individual z/OS elements and features, and their service, before you can perform an IPL. Installation instructions are in the z/OS Program Directory.

SystemPac

IBM SystemPac is a software package, which is available for an additional fee and offered worldwide, that helps you install z/OS, subsystems (DB2, IMS, CICS, Network Control Program (NCP), and WebSphere Application Server). SystemPac is tailored to your specifications; it is manufactured according to parameters and input/output definition file (IODF) definitions that you supply during order entry. The goal is to have the system tailored to your specifications and have products that are enabled according to your specified configuration. Parameters are collected by telephone. Using a printed questionnaire as a guide, you tell an IBM representative your responses. Upon completion, a printout showing all the parameters and definitions you specified is sent to you for reference. The documentation “SystemPac Installation Guide” that comes with your order specifies the integrated service level that is applicable to your order.

ShopzSeries

ShopzSeries is an Internet application that you can use to order z/OS software products and service. Using ShopzSeries, you can order corrective and preventive service over the Internet, with delivery over the Internet or by tape. Service with ShopzSeries reduces your research time and effort by using your uploaded SMP/E consolidated software inventory (CSI) to ensure that all applicable service, including a reach-ahead service, for the installed FMIDs in the target zones is selected.

5.2.21 SMP/E

SMP/E is a tool that manages the installation of software products on your z/OS system and tracks the modifications that you make to those products. Usually, it is the system programmer's responsibility to ensure that all software products and their modifications are properly installed on the system. The system programmer also must ensure that all products are installed at the proper level so all elements of the system can work together. At first, that might not seem too difficult, but as the complexity of the software configuration increases, so does the task of monitoring all the elements of the system. It controls these changes at the element level by performing the following tasks:

- ▶ Selecting the proper levels of elements to be installed from many potential changes
- ▶ Calling system utility programs to install the changes
- ▶ Keeping records of the installed changes

SMP/E is a part of the installation, service, and maintenance processes for CBPDOs, IBM ProductPac® offerings, IBM ServicePac® offerings, and selective follow-on service for CustomPac offerings. In addition, SMP/E can be used to install and service any software that is packaged in SMP/E system modification (SYSMOD) format. SMP/E can install software if it is packaged as a SYSMOD. When SMP/E processes SYSMODs, it installs the elements in the appropriate libraries and updates its own records of the processing it has done. It installs program elements into two types of libraries:

- ▶ **Target libraries:** These libraries contain the executable code that you need to run your system (for example, the libraries from which you run your production system or your test system).
- ▶ **Distribution libraries:** These libraries contain the master copy of each element for a system. They are used as input to the SMP/E **GENERATE** command or the system generation process to build target libraries for a new system. They also are used by SMP/E for backup when elements in the target libraries must be replaced or updated.

The Consolidated Software Inventory (CSI) data sets contain all the information SMP/E needs to track the target and distribution libraries. The CSI does not contain the element itself, but rather a description of the element it represents. In the CSI, entries for the elements in the distribution and target libraries are grouped according to their installation status. These groupings are known as SMP/E zones. There are three types of zones in a CSI:

- ▶ **Global zone:** Contains entries that are needed to identify and describe each target and distribution zone to SMP/E, and stores information about SMP/E processing options. Contains status information for all SYSMODs SMP/E has begun to process and holds exception data for SYSMODs requiring special handling or that are in error. This data set is also known as the master CSI.
- ▶ **Target zone:** Contains information that describes the content, structure, and status of the target libraries. It also contains a pointer to the related distribution zone, which can be used in **APPLY**, **RESTORE**, and **LINK** when SMP/E is processing a SYSMOD and needs to check the level of the elements in the distribution libraries.
- ▶ **Distribution zone:** Contains information that describes the content, structure, and status of the distribution libraries. Each distribution zone also points to the related target zone, which is used when SMP/E is accepting a SYSMOD and needs to check whether the SYSMOD already is applied in the target zone.

5.2.22 Cross-system coupling facility

The cross-system coupling facility (XCF) component of z/OS provides simplified multisystem management. XCF services allow authorized programs on one system to communicate with programs on the same system or on other systems. If a system fails, XCF services also provide the capability for batch jobs and started tasks to be restarted on another eligible system in the sysplex.

An XCF group is a set of related members that a multisystem application defines to XCF. A member is a specific function, or instance, of the application. A member is on one system and can communicate with other members of the same group across the sysplex. Communication between group members on different systems occurs over the signaling paths that connect the systems; on the same system, communication between group members occurs through local signaling services. To prevent multisystem applications from interfering with one another, each XCF group name in the sysplex must be unique.

The terms *instance*, *exploiting instance*, or *exploiter* are used to denote the active subsystem or address space of a component, product, or function that directly exploits sysplex services. XCF is a z/OS component that provides services to allow multiple instances (named members) of an application or subsystem, running on different systems in a sysplex, to share status information and communicate through messages with each other. A member of an application can use XCF services to accomplish the following tasks:

- ▶ Inform other members of their status (active, failed, and so on).
- ▶ Obtain information about the status of other members, such as whether another member failed.
- ▶ Send messages to and receive messages from each other member, in the same or in another z/OS. The most frequent case is other z/OSs (inter-system communication).

If z/OS fails, XCF can call Automatic Restart Manager (ARM) services (a z/OS component) to provide the capability for batch jobs and started tasks to be restarted on another eligible z/OS in the sysplex.

5.2.23 z/OS Security Server

The z/OS Security Server is the IBM security product. The RACF product is a component of the z/OS Security Server, and it works together with the existing system features of z/OS to provide improved data security for an installation. If this product is installed in your environment, then RACF customization must be done. RACF helps meet the need for security by providing the following functions:

- ▶ Flexible control of access to protected resources
- ▶ Protection of installation-defined resources
- ▶ Ability to store information for other products
- ▶ Choice of centralized or decentralized control of profiles
- ▶ An ISPF panel interface
- ▶ Transparency to users
- ▶ Exits for installation-written routines

RACF security protection

RACF controls access to and protects resources. For a software access control mechanism to work effectively, it must first identify the person who is trying to gain access to the system, and then verify that the user is really that person. RACF uses a user ID in a user profile in the RACF database and a system-encrypted password or pass phrase (new with z/OS V1R8) to perform its user identification and verification.

When you define a user to RACF, you assign a user ID and temporary password. The user ID identifies the person to the system as a RACF user. The password or passphrase verifies the user's identity. The temporary password permits initial entry to the system, at which time the person is required to choose a new password.

RACF authorization checks

Having identified a valid user, the software access control mechanism must next control interaction between the user and the system resources. It must authorize not only what resources that user can access, but also in what way the user can access them, such as for reading only, or for updating and reading. This controlled interaction, or authorization checking, is done by RACF. Before this activity can take place, however, someone with the proper authority at the installation must establish the constraints that govern those interactions. With RACF, you are responsible for protecting the system resources (data sets, tape and DASD volumes, IMS and CICS transactions, TSO logon information, and terminals) and for issuing the authorities by which those resources are made available to users. RACF records your assignments in profiles that are stored in the RACF database. RACF then refers to the information in the profiles to decide whether a user is to be permitted to access a system resource.

5.2.24 Language Environment

Language Environment provides a common runtime environment for IBM versions of certain high-level languages (HLLs), namely, C, C++, COBOL, Fortran, and PL/I, in which you can run existing applications that are written in previous versions of these languages and in the current Language Environment-conforming versions, without having to recompile or relink-edit the applications. Before Language Environment, each of the HLLs had to provide a separate runtime environment. POSIX-conforming C applications can use all Language Environment services.

Language Environment combines essential and commonly used runtime services, such as routines for runtime message handling, condition handling, storage management, date and time services, and math functions, and makes them available through a set of interfaces that are consistent across programming languages. With Language Environment, you can use one runtime environment for your applications, regardless of the application's programming language or system resource needs, because most system dependencies are removed.

The assembly language is a symbolic programming language that you can use to code instructions in a mnemonic format instead of coding in machine language in binary format. Assembly language lets you use meaningful symbols that are made up of alphabetic and numeric characters, instead of just the binary digits 0 and 1 that are used in machine language instructions. This makes your coding easier to read, understand, and change. Assembly language is also enriched by a huge set of macros that make it a powerful language; however, its major quality is the performance of its object code at run time.

A HLL is a programming language above the level of assembly language and below that of program generators and query languages. The statements that are created in one of these languages can be the input to a computer program called a compiler that translates the HLL statements in machine language instructions, that is, object code. It has instructions that are recognized by the processor of the specific platform where it is supposed to be run. This object code is the fabric of the executable program, also called the load module or program object. Some of the common HLLs are C/C++, COBOL, Fortran and PL/I.

z/OS Language Environment is the prerequisite runtime environment for applications that are generated with the IBM compiler products. Language Environment provides a single language runtime environment for COBOL, PL/I, C, and Fortran. In addition to support for existing applications, Language Environment also provides common condition handling, improved interlanguage communication (ILC), reusable libraries, and more efficient application development. Application development is simplified by the use of common conventions, common runtime facilities, and a set of shared callable services.

5.3 Network functions

Networking of computers started early with the mainframe even before the age of desktop computers and Internet. There was even long-distance connectivity between the corporate data centers and their remote branches. Today, z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks (WANs), including the most popular WAN, the Internet. It provides both SNA and TCP/IP networking protocols for z/OS. z/OS Communications Server provides APIs and networking protocol support to enable Systems Network Architecture (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 distant locations. z/OS Communications Server is the IBM implementation of SNA and standard TCP/IP protocol suites on the System z platform. SNA is the IBM proprietary networking protocol, and 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.

The SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing (HPR) protocols. SNA is a data communication architecture that was 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 alternative paths that messages can take between different NCP nodes. A subarea is the collection of terminals, workstations, and phone lines that are 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. HPR allows you to migrate NCP connections to APPN connections without incurring the associated increase in storage and cycles. Earlier releases of VTAM support type 2.1 casual connection, 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, use APPN to connect two VTAMs in a peer-to-peer relationship.

TCP/IP is the set of communications protocols that are used both for the Internet and other similar networks. TCP is a transport layer protocol that is used by applications that require ensured delivery. TCP establishes a full duplex virtual connection between two endpoints. Each endpoint is defined by an IP address and a TCP port number. 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. TCP/IP runs on z/OS UNIX System Services, which are 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 z/OS functions and file types.

5.4 Application software

Other than the z/OS or the operating system components, there are many other applications and middleware that help host the business functions. There is a wide variety of such category of software products but few are common. Databases, transaction servers and middleware are the major categories into which most of these products fit.

On z/OS, there are database products such as the IMS and the DB2, both by IBM. These are hierarchical and relational, respectively. There are other non IBM database products, such as Oracle or ADABAS, which also can be hosted on the mainframe. The transaction servers, even though considered as a middleware product, form a special family of products in the mainframe. They perform various functions in applying the business logic to the data that is stored in the databases and presenting it to the users. Traditionally, the presentation was restricted to terminal-based output devices or printers, but the technology has taken it to dynamic presentation and mobile devices over the recent years. IBM has products such as the CICS Transaction Server or the IMS Transaction Manager to perform the role of transaction systems. Then, there is the larger category of products, which is the middleware. Middleware ranges from software that does business logic processing, to monitoring and debugging or high-end data analytics. Middleware includes the application servers, such as the WebSphere suite of products, to a wide range of products to perform various functions.

5.5 ISV/OEM software

A mainframe is a large hosting platform that caters to the large-scale needs of an organization. With its enormous processing power, a mainframe is deployed for a wide variety of critical business processes and applications. Although IBM continues to hold the market for the hardware and the operating systems, in addition to some of the major software products, this situation has created a large market for external independent software vendors and developers. There is an enormous list of products from various large, medium, and small vendors, including IBM, to assist users in their business applications in different ways. These span all categories of software:

- ▶ Database managers
- ▶ Transaction systems
- ▶ Application and web servers
- ▶ Messaging systems
- ▶ Scheduling systems
- ▶ System automation
- ▶ System monitoring
- ▶ System or application debugging
- ▶ Programming languages
- ▶ Device management
- ▶ Session management
- ▶ Library management
- ▶ License management
- ▶ Hardware management

The software products aid the system programmers, developers, the users in performing their tasks or associating their business functions or applications easily with the operating platform or other distributed platforms outside the mainframe.

5.6 Conclusion

The z/OS, being the most widely used operating system on mainframes, delivers exemplary performance and stability through its architecture, which that is made of up core functions and components that are described in this chapter. Some of the functions, such as the WLM, expand their capabilities to hardware and other extended platforms that work with mainframe, such as the blade server extensions. Such a scalable architecture is in the design of every component and function of the mainframe operating systems, making them flexible and pluggable in to any new technologies or standards.

Thus, z/OS and the other major operating systems that can run on a mainframe use their capabilities to deliver a stable platform for hosting business applications. Every component and function within the hardware and software is designed to complement the overall design and architecture of this oldest and long-lasting commercial computing platform.



Security solutions for IBM System z

This chapter provided an overview of security solutions for System z and how they can contribute to your environment. There are many solutions that are available that can contribute to the implementation of the IBM Security Framework and use the strengths of IBM System z. These products relate to components such as Data Security, Security Intelligence, Policy and Processes, Audits and Compliance, Application Security, Access/Privilege Management, and Identity Management. These products either complement the mainframe built-in security services or they provide additional enterprise-wide security-oriented functions.

The following products are described in this chapter:

- ▶ IBM InfoSphere Guardium for z/OS
- ▶ IBM Security QRadar
- ▶ IBM Security zSecure Suite
- ▶ IBM Security Key Lifecycle Manager
- ▶ IBM Enterprise Key Management Foundation
- ▶ Encryption Facility for z/OS
- ▶ IBM Security AppScan
- ▶ IBM Security Access Manager
- ▶ IBM Security Identity Manager
- ▶ Federated Identity Management

6.1 IBM InfoSphere Guardium for z/OS

The IBM InfoSphere Guardium solution offers a simple, powerful mean of securing critical data throughout the enterprise. It provides rapid, policy-based detection of anomalous activities that violate corporate policies, real-time responses, such as alerts, auditable workflow to ensure appropriate resolution of exceptions, and automated reporting capabilities, which simplify validation of compliance for mandates, such as SOX, PCI DSS, and data privacy regulations.

The InfoSphere Guardium for z/OS solution provides these capabilities for DB2, IMS, and VSAM on z/OS. The solution can be used independently for the mainframe environment only, or integrated with other InfoSphere Guardium database security and monitoring components throughout the enterprise to provide a secure, centralized audit repository and management point.

InfoSphere Guardium for z/OS uses lightweight software probes that are called S-TAPs to capture DB2, IMS, and VSAM activities by privileged users, mainframe-resident applications, and network clients, including those connecting through services such as JDBC, DB2, or IMS Connect. IBM event-capture technologies are used for each environment to ensure that all critical operations are captured, without the use of Class 4 and Class 5 audit traces.

Each S-TAP on z/OS is designed for the unique monitoring requirements of a particular data environment. The IBM InfoSphere Guardium S-TAP for DB2 on z/OS module monitors all DB2 activities, including SELECTs, DML, data definition language (DDL), and changes in access privileges.

To enhance performance, the underlying DB2 event-capture technology can be shared with IBM Query Monitor in systems using both offerings. The IBM InfoSphere Guardium S-TAP for VSAM on z/OS module supports a comprehensive range of VSAM file types, including entry-sequenced data set (ESDS), key-sequenced data set (KSDS), relative record data set (RRDS), virtual relative record data set (VRRDS), and linear data set (LDS), monitoring OPENs, READs, UPDATEs, DELETEs, CREATEs, and ALTERs.

The IBM InfoSphere Guardium S-TAP for IMS on z/OS module monitors both online and batch tasks, providing auditing and policy management that is related to READs, INSERTs, UPDATEs, and DELETEs.

Each S-TAP sends information that is specified by user-defined audit policies to an InfoSphere Guardium Collector for z/OS appliance. This ensures that the mainframe is not burdened with incremental storage or processing requirements, network traffic is limited, and a full audit trail is stored securely. Figure 6-1 on page 127 shows the S-TAP infrastructure.

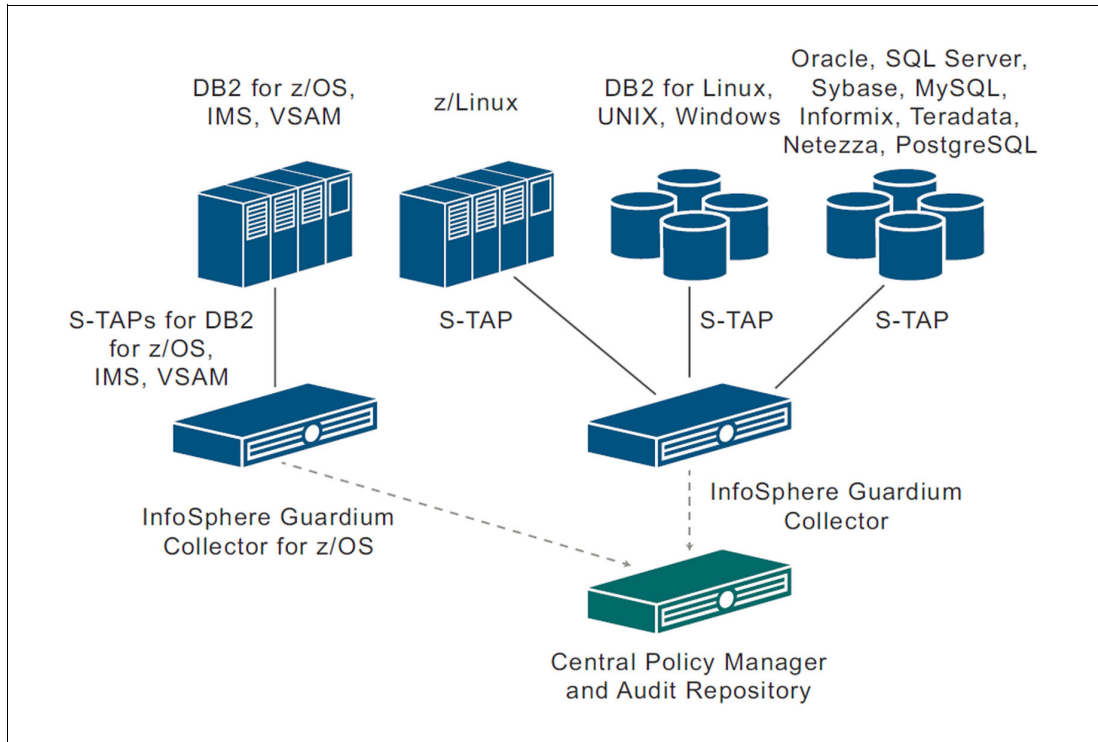


Figure 6-1 InfoSphere Guardium S-TAP

IBM InfoSphere Guardium S-TAP for DB2 on z/OS (also referred to as InfoSphere Guardium S-TAP for DB2, and S-TAP) collects and correlates data access information from various f DB2 resources to produce a comprehensive view of business activity for auditors. InfoSphere Guardium S-TAP for DB2 provides the following features and functions:

Data collection-InfoSphere Guardium S-TAP for DB2 can collect and correlate many different types of information into an administration repository:

- ▶ Modifications to an object (SQL **UPDATE**, **INSERT**, or **DELETE**)
- ▶ Reads of an object (SQL **SELECT**)
- ▶ Explicit **GRANT** and **REVOKE** operations (to capture events where users might be attempting to modify authorization levels)
- ▶ Assignment or modification of an authorization ID
- ▶ Authorization attempts that are denied because of inadequate authorization
- ▶ **CREATE**, **ALTER**, and **DROP** operations against an object (such as a table)
- ▶ Utility access to an object (IBM utilities only)
- ▶ DB2 commands entered (including the ability to determine which users are issuing specific commands)

InfoSphere Guardium S-TAP for DB2 collects SQL and IFI collector data from an audited DB2 subsystem, according to the filtering policies you set through the Guardium interface.

The InfoSphere Guardium S-TAP for DB2 collector agent is responsible for the collection of audit data in an InfoSphere Guardium S-TAP for DB2 environment. The collector agent workflow is shown in Figure 6-2. Data is filtered and sent to the InfoSphere Guardium system, which enables you to view reports on your workstation.

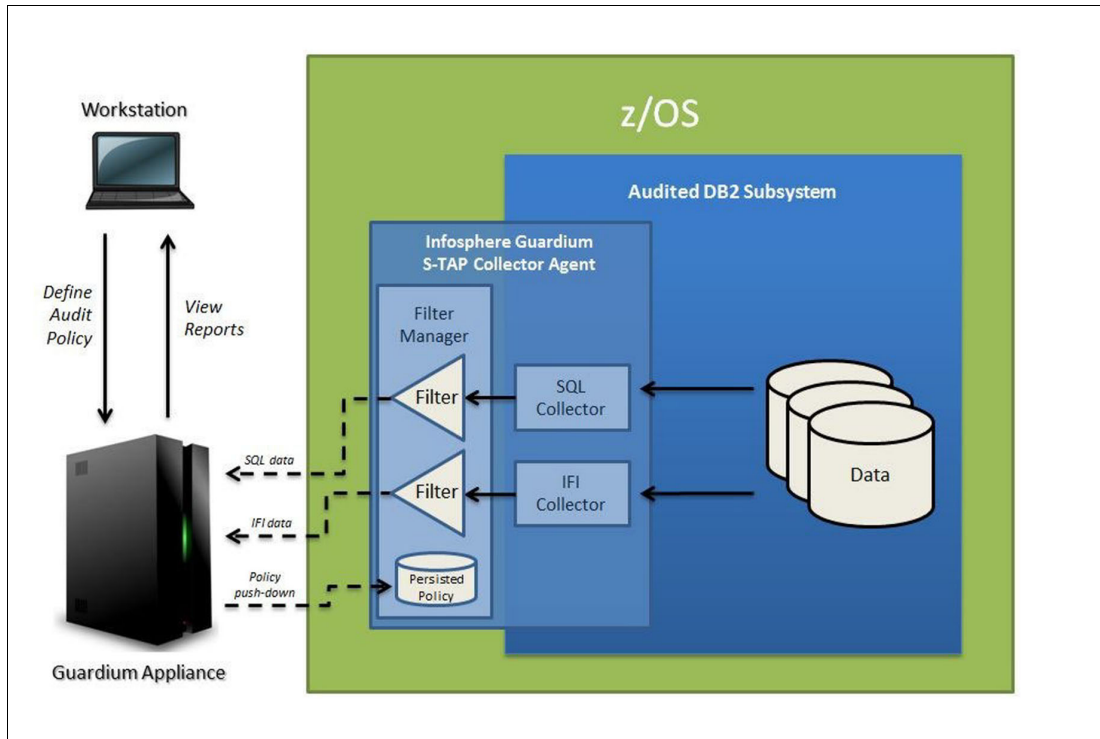


Figure 6-2 z/OS Audited DB2 subsystem with S-TAP collector agent

6.2 IBM Security zSecure Suite

The IBM Security zSecure Suite is a family of products that helps you enhance your mainframe administration, and provide a solution for audit and compliance reporting and management within your z/OS infrastructure. The products are shown in the product matrix in Figure 6-3 on page 129.

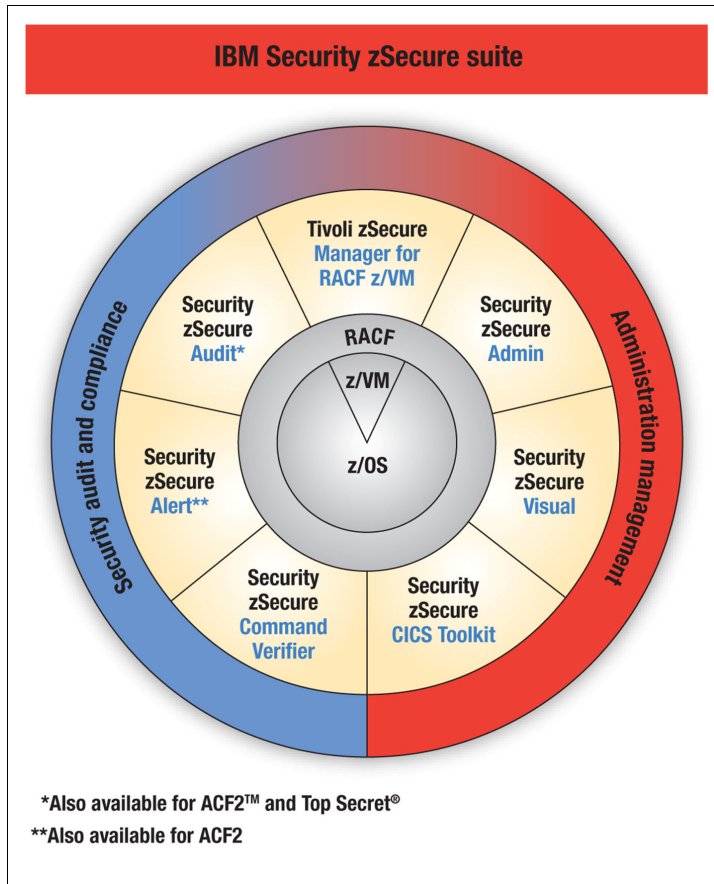


Figure 6-3 IBM Security zSecure Suite

The components of Security zSecure Suite are divided into two categories. The first category is administration management, to which Security zSecure Admin, Visual, Security zSecure for RACF on z/VM and the Security zSecure CICS Toolkit belong. The second category is defined as security audit and compliance. This category consists of Security zSecure Audit, Alert, and zSecure Command Verifier. Security zSecure Manager for RACF z/VM also includes the Audit component.

Security zSecure Suite is built upon the special purpose Common Audit and Reporting Language (CARLa).

6.2.1 Administration management with Security zSecure Suite

The proper administration of RACF databases is a cornerstone of a secure mainframe environment. Security zSecure administration products for RACF provide a “what you see is what you get” (WYSIWYG) access pattern to the RACF database. Administration for z/OS RACF is performed using either Security zSecure Admin or the Windows component Visual. The administration for RACF for z/VM is performed using Security zSecure RACF Manager for z/VM. The user interface to RACF is the same for z/OS and z/VM. The visual component offers a Windows based approach that differs from the host software.

Security zSecure Admin

This product consists of an Interactive System Productivity Facility (ISPF)-based user interface for the administration of RACF and reporting about the RACF database. It runs entirely within z/OS without requiring any collaborating component on a distributed platform.

Security zSecure Admin is intended to enable more efficient and effective RACF administration, using fewer resources, and can be used to perform the following functions:

- ▶ Automate routine tasks to simplify administration
- ▶ Identify and analyze problems to minimize threats
- ▶ Merge databases quickly and efficiently
- ▶ Display data from the active (live) RACF database
- ▶ Integrate smoothly with IBM Security zSecure Audit
- ▶ Store non RACF data to reduce organizational costs
- ▶ Analyze the access patterns in RACF database using the Access Monitor component
- ▶ Enable nondisruptive change testing using RACF Offline
- ▶ Administer multiple z/OS instance from a single system using the Security zSecure Server multi-system support

Figure 6-4 shows an ISPF panel that is generated by Security zSecure Admin. Security zSecure Admin provides an interface that makes information from RACF database visible to the user in an ISPF edit-like fashion. Using the analogy of an ISPF editor to present RACF profiles, the user can type over fields and make the changes in WYSIWYG mode. When the user makes a change to the panel and presses Enter, the proper RACF command is automatically generated. This approach is available for User, Group, Dataset, and General Resource profiles. Additionally, the user can perform various selections on the RACF data, for example, query which users have never logged on. There are several other features in Security zSecure Admin, for example, a Helpdesk function and digital certificate management functions, which are not described here. For more information about additional features, see the product reference documentation.

```

                                0 s elapsed, 0.1 s CPU
zSecure Admin+Audit for RACF USER overview
Command ==>
                                Scroll==> PAGE
All users
User      Complex  Name                               DfltGrp  Owner      RIRP  SOA  gC  LCX  Grp
--- $NIC      ZT01      NICO CHILLEMI                     ZTIVOLI  $NIC      _____  _____  X  1
--- irrcerta  ZT01      CERTAUTH Anchor                    _____  irrcerta  R      _____  CX  0
--- irrmulti  ZT01      Criteria Anchor                    _____  irrmulti  R      _____  X  0
--- irrsitec  ZT01      SITE Anchor                        _____  irrsitec  R      _____  X  0
--- AAADMIN   ZT01      WAS ADMINISTRATOR                  AACFG    SENIOR    I      _____  1
--- AAADMSH   ZT01      WAS ASYNCH ADMIN TAS              AACFG    SENIOR    P      _____  1
--- AACRU     ZT01      WAS DAEMON CR                     AACFG    SENIOR    I P    _____  C  1
--- AACTWTR   ZT01      WAS TRACE WRITER                  AACFG    SENIOR    P      _____  1
--- AAPUB     ZT01      WAS DEFAULT USER                 AAPUBG   SENIOR    IR     _____  X  1
--- AARONP    ZT01      AARON PATENAUDE                   SYSPROG  PTORRES   I  S    _____  X  1
--- AASALEH   ZT01      AHMED SALEH EG TIV                TIVUSR   SYS1     I      _____  X  1
--- AASRU     ZT01      WAS APPSVR SR                     AASRG    SENIOR    I P    _____  2
--- ABARNES   ZT01      ALAN BARNES US OPTIM              DB2USR   SYS1     I      _____  X  4
--- ACRAND    ZT01      ANTHONY RANDOLPH US              IMSUSR   SYS1     _____  _____  X  2
--- ADHUSER   ZT01      _____                          SYSPROC  RICHARD  _____  _____  2
--- ADOUARI   ZT01      ANDREA DOUARINOU FR              TIVUSR   SYS1     I      _____  X  2
--- ADPOT00   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT01   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT02   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT03   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT04   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT05   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT06   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT07   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
--- ADPOT08   ZT01      STUDENT EPS AD/PD TO              ADTUSR   PIERRE   I      _____  X  7
F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP lis F10=LEFT    F11=RIGHT   F12=RETRIEVE

```

Figure 6-4 Security zSecure Admin user selection panel

Security zSecure Access Monitor

The Access Monitor component is a feature of Security zSecure Admin. It allows you to record all access decisions that are made by RACF and stores them in sequential files, regardless of the audit flags that might have been set. Using this approach, it is possible to determine and to get rid of unused access and profiles, and prevent users from having excessive access rights. The Access Monitor is a unique way of performing comprehensive RACF cleanup. The data that is produced by the Access Monitor can be used to analyze RACF using either ISPF panels within Security zSecure or the CARLa programming language.

Security zSecure Visual

This component of the Security zSecure suite consists of a Windows based user interface running on a Windows machine. It communicates, through TCP/IP, with a z/OS started task that performs limited RACF administration on behalf of the user. The intent of Security zSecure Visual is to reduce the need for sometimes scarce RACF trained expertise by using a Microsoft Windows based GUI instead that is used to drive RACF administration. Security zSecure Visual can accomplish the following tasks:

- ▶ Decentralize RACF administration to optimize resources
- ▶ Scope down administrative capabilities
- ▶ Avoid need for TSO/ISPF rollouts
- ▶ Administer a live RACF database
- ▶ Easily clone user templates

Figure 6-5 shows an example of graphical menus that are used by Security zSecure Visual, which show a list of USER profiles in the RACF database with the REVOKED attributes. These profiles were obtained after triggering a search for USER profiles with the drop-down menu at the lower right corner of the window.

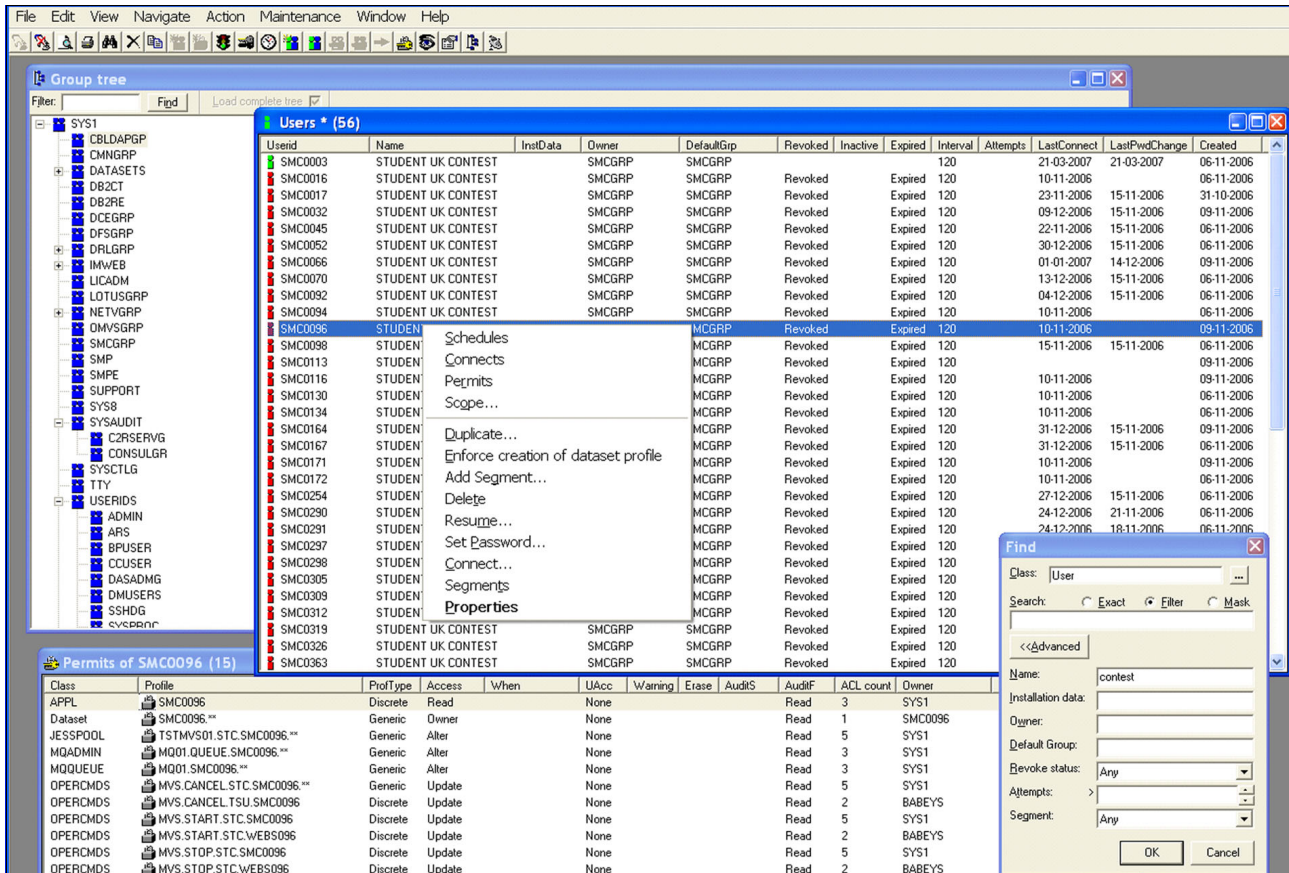


Figure 6-5 Security zSecure Visual user interface

Security zSecure CICS Toolkit

This component has two facets. It is a pre-built administrative interface that runs as a CICS transaction in a CICS region. It also provides a CICS API to allow applications to perform their own security functions. For example, if an application needs a reverification of user credentials when certain program constraints are met (such as funds transfer over a certain amount), the Security zSecure CICS Toolkit API can be used to drive this reverification.

Similarly, focusing on the same examples, the Security zSecure CICS Toolkit API also can be used to drive RACF audit capability. For example, an SMF record can be generated to log the fact that this particular user performed a funds transfer above a certain amount.

Figure 6-6 on page 133 shows a pre-built CICS RACF administration menu.


```

Session E - [24 x 80]
Termid = 0S25          IBM Tivoli zSecure CICS Toolkit      Date = 2007/247
Userid = CRMAROB          MAIN MENU                      Time = 12:21:42
Name   = VAN HOBOKEN, ROB

PF01 ADGRP  PF02 ADUSER  PF03 ALTGRP  PF04 ALUSER  PF05 CONNCT  PF06 DELDSD
PF07 DELGRP  PF08 DELUSR  PF09 LDSD   PF10 LGRP    PF11 LUSER   PF12 PERMIT
PF13 RALTER  PF14 RACLNK  PF15 RDEFNE  PF16 RDELTE  PF17 REMOVE  PF18 RLIST
PF19 USRDAT

Number ==> _

Licensed Materials - Property of IBM
5655-T05 Copyright IBM Corp. 1988, 2007. All Rights Reserved.

Use PF key or enter NUMBER for desired command. Press CLEAR to exit
MA e 16/041

```

Figure 6-6 Security zSecure CICS Toolkit

6.2.2 Security audit and compliance with Security zSecure

Security audit and compliance for z/OS systems is achieved with the Security zSecure Audit, Alert, and Security zSecure Command Verifier components of the Security zSecure suite. Using these tools, you can ensure that your system becomes compliant with the rules and regulations that you must adhere to and to ensure that your system stays compliant under these rules.

Security zSecure Audit

The Audit component of the Security zSecure suite is an ISPF-based application that uses input from the RACF database and the z/OS system to generate audit reports and provide information about the security strength of your system. The z/OS information is either extracted from SMF or a CKFREEZE file, which is a system snapshot containing the most important and security relevant information about your z/OS system. It is a safe and fast data repository on the basis of which audit and compliance reporting can be performed. Additionally, Security zSecure Audit supports reporting about major z/OS middleware, such as DB2, IMS, and CICS, and z/OS UNIX and IP information. All this information can either be accessed through the ISPF interface or by using self-written CARLa programs that are fitted to your own needs. Security zSecure Audit uses a compliance framework that is based on Common Criteria security levels, preferred practices from the industry, and a compliance reporting framework that supports industry standards for z/OS compliance and self-written system-specific standards.

The following list provides an overview of the features and benefits of Security zSecure Audit:

- ▶ Gather and analyze critical information efficiently
- ▶ Support security information and event management (for example, IBM QRadar® SIEM)
- ▶ Customize reports to meet specific needs
- ▶ Analyze RACF profiles to get fast answers
- ▶ Analyze SMF log files to create a comprehensive audit trail
- ▶ Use external file support to make reports highly usable

- ▶ Detect system changes to minimize security risks
- ▶ Track and monitor baseline changes for RACF
- ▶ Detect integrity breaches
- ▶ Perform reporting not only to RACF, but also z/OS, UNIX, and major z/OS middleware

Security zSecure Audit can be used to feed data into the QRadar SIEM solution. This support has been integrated into the product and is further described in 6.3.8, “Integration of QRadar SIEM with z/OS” on page 142

Figure 6-7 shows the record selection menu for status audits in Security zSecure Audit

```

Menu          Options          Info          Commands          Setup
-----
                                zSecure Admin+Audit for RACF          0.1 s CPU, RC=0
Command ==>
-----
Enter / to select report categories
- MVS tables          MVS oriented tables (reads first part of CKFREEZE)
- MVS extended        MVS oriented tables (reads whole CKFREEZE)
- RACF control        RACF oriented tables
- RACF user           User oriented RACF tables and reports
- RACF resource       Resource oriented RACF tables and reports

Select options for reports:
/ Select specific reports from selected categories          Audit policy
/ Include audit concern overview in overall prio order     / zSecure
- Only show reports that may contain audit concerns       - C1
- Minimum audit priority for audit concerns (1-99)        - C2
- Show differences                                         - B1
- Print format          - Concise (short) report
- Background run

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN      F9=SWAP lis F10=LEFT      F11=RIGHT     F12=RETRIEVE

```

Figure 6-7 Security zSecure Audit

Security zSecure Alert

Security zSecure Alert is a real-time security monitoring solution relying on z/OS SMF data. It is designed to monitor critical security conditions within your z/OS system and to send out alerts quickly if such a condition is detected. Therefore, it uses technology to intercept the SMF records even before they are written to the log and analyzes them on the basis of predefined conditions. If, for example, you want to be notified if there is a successful change to one of your critical system data sets, Security zSecure Alert can be configured to extract that information out of the SMF records and can notify you in various, definable, ways, including SMS, SMTP email, and UNIX syslog.

The following list summarizes the capabilities of Security zSecure Alert:

- ▶ Monitor sensitive data for misuse to enhance access controls
- ▶ Use configurable alerts to analyze events, improve security, and reduce costs
- ▶ Detect configuration mistakes or changes before others use them and audit for compliance reporting

A sample of user alerts is shown in Figure 6-8 on page 135.

```

zSecure Admin+Audit for RACF - Setu Row 1 to 13 of 15
Command ==> _____ Scroll ==> CSR

User alerts
Select the alert you want to work with.
The following line commands are available: A(Preview), C(opy), D(elete),
E(dit), I(nsert), W(Who/Where),S(elect), U(nselect), B(rowse)
-----
Alert                               Id    Sel  gECSW  C
= Logon by unknown user              1101  Yes   gE     W
- Logon with emergency userid         1102  Yes   gE     W  Y
- Logon of a userid with UID(0) (Unix superuser) 1103  Yes   gE     W
- Highly authorized user revoked for pwd violations 1104  Yes   gE     W
- System authority granted            1105  Yes   gE     W
- System authority removed            1106  Yes   gE     W
- Group authority granted             1107  Yes   gE     W
- Group authority removed             1108  Yes   gE     W
- SPECIAL authority used by non-SPECIAL user    1109  Yes   gE     W
- non-OPERATIONS user accessed data set with OPERAT 1110  Yes   gE     W
- Invalid password attempts exceed limit 1111  Yes   gE     W
- Password history flushed            1112  Yes   gE     W
- Suspect password changes            1113  Yes   gE     W

```

Figure 6-8 Security zSecure Alert - sample list of user alerts

Security zSecure Command Verifier

This component of Security zSecure catches mistakes of administrators. Administrators might mistype parameters of commands, or forget to specify a parameter. Sometimes, naming conventions or security standards must be applied to new or existing profiles. In other cases, you might not fully trust your administrators to make every change in the system.

Security zSecure Command Verifier uses RACF command screening. It verifies every RACF command that is issued before it is run, that is, before the actual change is made in the RACF database. It uses filters to specify which profiles are within scope for an administrator. It uses a granular policy that is defined by RACF profiles that can be used to specify exactly what each administrator can do and what commands the administrator cannot issue. The verification takes place before the commands are run, so inappropriate commands can be prevented, Parameters that were omitted may be added by policy rules, and inappropriate parameters may be removed or changed by policy rules.

An interesting example for a missing parameter is the **UACC** parameter in a command to add a data set profile. Depending on some conditions, the resulting profile might get a **UACC** of READ, UPDATE, or worse, instead of the preferred value NONE. Through a simple Security zSecure Command Verifier policy rule, a parameter of **UACC(NONE)** can be added to every **ADDSD** or **RDEFINE** command where the parameter is missing. Another easy policy rule is to deny all **UACC** values other than NONE. Security zSecure Command Verifier changes a command if there is a corresponding policy rule in the RACF database, and leaves the command unaffected if there is no policy rule for the profile or the parameter.

Security zSecure Command Verifier's policy is controlled by the access list on the policy profiles and applies to all RACF command issuers, including system special users. This allows you to limit the system special's ability to change profiles and settings in RACF, reducing the risk that the system special privilege poses.

The following list summarizes the capabilities of Security zSecure Command Verifier. Figure 6-9 shows a sample output from a system that is using Security zSecure Command Verifier and where forbidden commands were entered:

- ▶ Verify commands before processing to proactively monitor policy compliance
- ▶ Take control by setting policies, alerts, and default values
- ▶ Reduce database cleanup time and audit concerns
- ▶ Reduce the risk of security breaches and failed audits
- ▶ Raise alerts when risky commands are run to help reduce chances of outages

```
setr password(nohistory)
C4R751E SETROPTS PASSWORD.HISTORY not allowed, command terminated
READY
setr password(interval(180))
C4R751E SETROPTS PASSWORD.INTERVAL not allowed, command terminated
READY
permit irr.password.reset class(facility) id(ibmuser) access(update)
C4R607E ACL setting for self to UPDATE not allowed, command terminated
READY
ralter facility irr.password.reset uacc(update)
C4R600E UACC UPDATE setting not allowed, command terminated
READY
setropts noclassact(facility)
C4R754E CLASSACT not allowed for class FACILITY, command terminated
READY
permit 'sys1.parmlib' gen id(ibmuser) access(update)
C4R646E Management of locked profiles not allowed, command terminated
READY
connect ibmuser group(sys1)
C4R548E You may not connect yourself to group SYS1, command terminated
READY
-
```

Figure 6-9 Security zSecure Command Verifier

6.3 IBM Security QRadar

The over-arching value of the IBM Security QRadar is its ability to tie intelligence from the network to the broader set of data that is collected from the entire enterprise infrastructure. It provides collection, analysis, and correlation across a broad spectrum of systems, including networked solutions, security solutions, servers, hosts, operating systems, and applications. The result is security intelligence that provides a meaningful context for security professionals while radically reducing operational complexity across multiple systems.

As with all IBM security intelligence solutions, IBM Security QRadar relies on a unified architecture for collecting, storing, analyzing, and querying log, threat, vulnerability, and risk-related data. IBM security intelligence solutions are built upon the three pillars of intelligence, integration, and automation, as shown in Figure 6-10 on page 137.

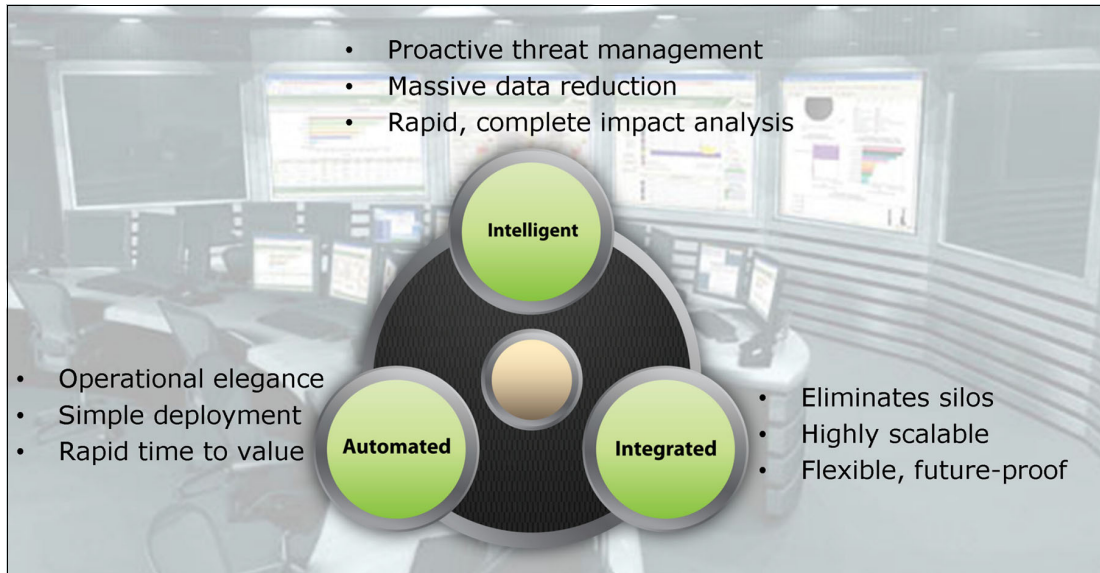


Figure 6-10 Intelligent, integrated, and automated IBM security intelligence solutions

The IBM Security QRadar security intelligence product family is based on a long-planned and carefully developed strategy to build an operating system approach to security intelligence. The IBM security intelligence operating system powers the IBM Security QRadar product family.

Organizations can benefit by having several departments and staff with various roles (such as operators, analysts, and auditors) and needs seamlessly using different modules.

Figure 6-11 shows the IBM security intelligence operating system as the foundation of the IBM security intelligence solutions.

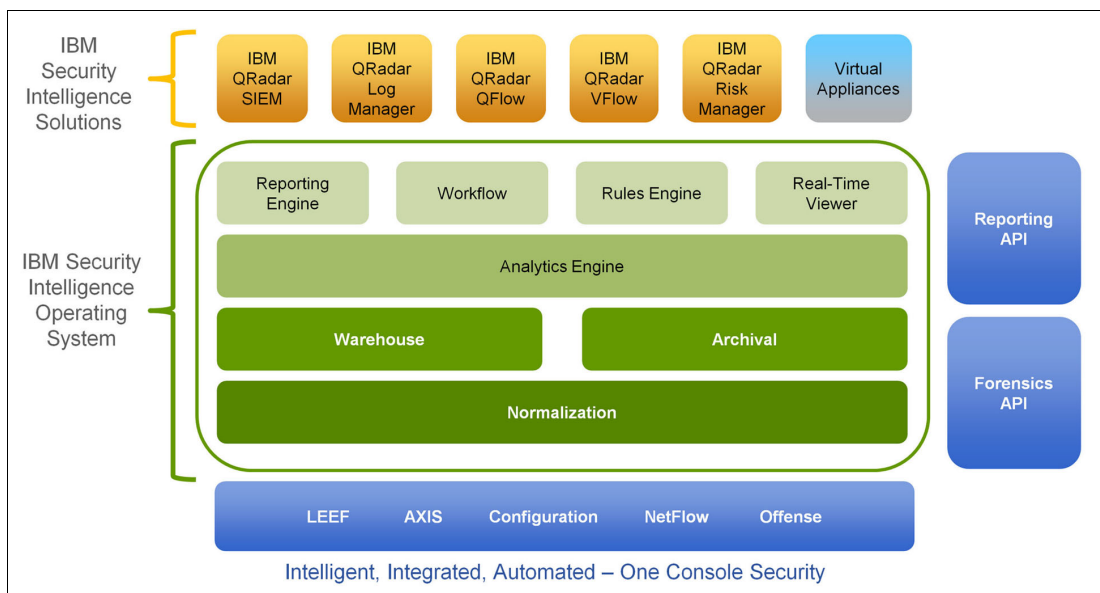


Figure 6-11 IBM security intelligence product family that is built on a common foundation

This framework is common for collecting, warehousing, filtering, analyzing, and reporting on all security intelligence telemetry. This integrated solution is the platform for risk management, security information and event management, log management, and network and application activity monitoring.

The IBM security intelligence operating system approach includes the following benefits:

- ▶ Convergence provides consolidation of previously siloed monitoring and analysis capabilities.
- ▶ Simplicity provides multiple functions that are delivered with a common user experience.
- ▶ Scalability provides simplified expansion capabilities for the largest infrastructures.

The IBM Security QRadar product family consists of the Log Manager, SIEM, Risk Manager, QFlow for network and application activity monitoring, VFlow for virtual activity monitoring, Vulnerability Manager, and Network Anomaly Detection.

6.3.1 IBM Security QRadar Log Manager

IBM Security QRadar Log Manager is a comprehensive solution for organizations that are looking to implement a distributed event log manager to collect, archive, and analyze network and security event logs. Log management has emerged as a required part of an organization's ability to deliver security preferred practices and to meet specific auditing and reporting requirements of various government regulations.

For example, it helps you meet the following government requirements:

- ▶ Federal Information Security Management Act
- ▶ Health Insurance Portability and Accountability Act
- ▶ North American Electric Reliability Corporation
- ▶ Payment Card Industry Data Security Standard
- ▶ Sarbanes-Oxley

IBM Security QRadar Log Manager provides numerous advantages, including the following examples:

- ▶ Fast and efficient deployment
- ▶ Distributed log collection and archival
- ▶ Policy-driven event log manager correlation
- ▶ Effective reporting and compliance auditing
- ▶ Reliable and tamper-proof log storage
- ▶ Possible upgrade to full IBM Security QRadar SIEM

6.3.2 IBM Security QRadar SIEM

IBM Security QRadar SIEM delivers the SIEM system solution that can give security professionals the visibility that they need to protect their networks. The advanced SIEM technology of IBM Security QRadar can protect IT assets from a growing landscape of advanced threats and can meet current and emerging compliance mandates.

The IBM Security QRadar next-generation SIEM is an intelligent, integrated, and automated SIEM system. It delivers the following benefits:

- ▶ Unified, turnkey deployments and more efficient administration and management
- ▶ Distributed correlation that can allow for billions of log entries and records to be monitored per day

- ▶ Single log archival capacity that ensures seamless reporting and comprehensive searching within the SIEM system
- ▶ Centralized command and control functions that reduce acquisition costs of the security management solution and improve IT efficiency
- ▶ Advanced threat and security incident detection that reduces the number of false positives and detects threats
- ▶ Compliance-centric workflow that enables the delivery of IT preferred practices that support compliance initiatives
- ▶ Distributed appliance architecture scales to provide log management in any enterprise network

By accepting data from various input sources, QRadar SIEM integrates security events into a unified framework for security incident and event management. These input types incorporate servers, devices, virtual machines, and other sources of information. Using Event Correlation and Anomaly Detection mechanisms, it can propose a consolidated view of possible incidents to the user. The QRadar workflow is shown in Figure 6-12.

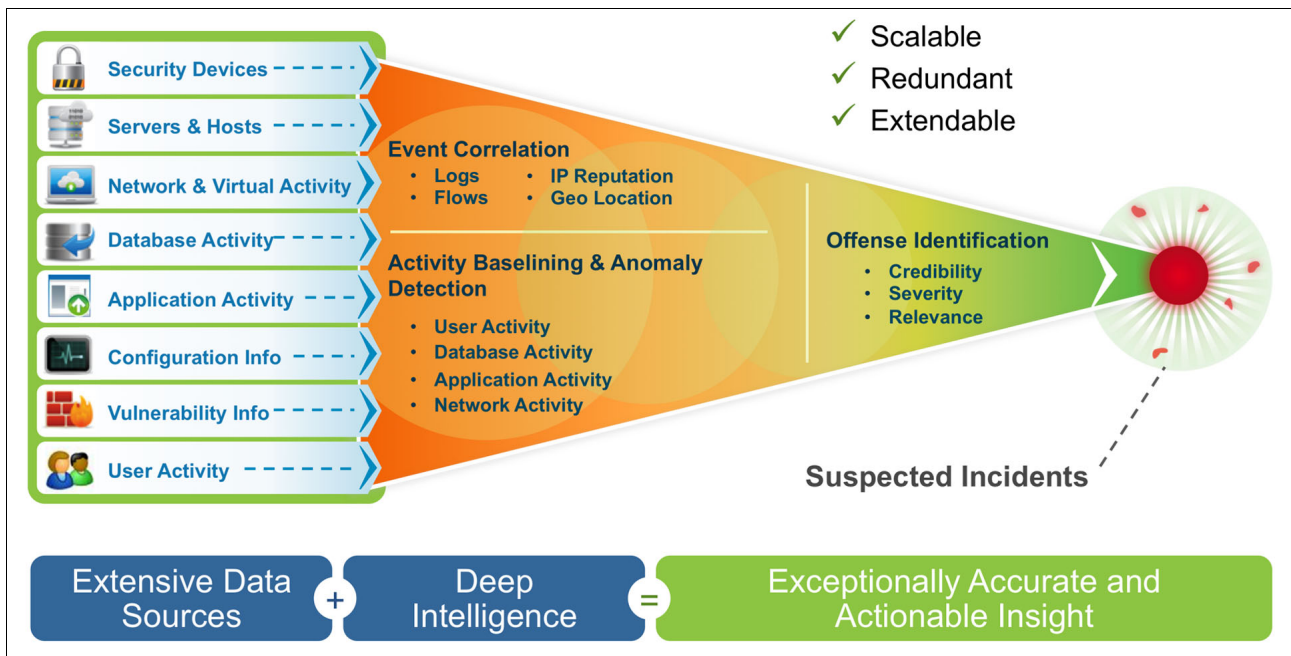


Figure 6-12 QRadar SIEM workflow illustration

6.3.3 IBM Security QRadar Risk Manager

IBM Security QRadar Risk Manager can provide organizations with a preliminary solution that network security professionals can use to assess which risks exist during and after an attack.

These network security professionals also can answer many “what if?” questions ahead of time, which can greatly improve operational efficiency and reduce network security risks. Powerful security analytics, simulation, and visualization tools can give network security professionals the ability to move away from day-to-day security fire fighting and to adopt a proactive, risk-based methodology that can improve network security and minimize the risk of exposures.

With IBM Security QRadar Risk Manager, network security teams have the tools that they need to achieve the following objectives:

- ▶ Automate compliance tasks and assess compliance risk, using the broadest set of risk indicators
- ▶ Simplify multivendor configuration audits to ensure consistency of device configuration and assess the risk of configuration changes
- ▶ See the risk impact of network changes, including new application and infrastructure deployments through enhanced security modeling and simulations
- ▶ Use powerful network security visualizations to gain insight into when traffic can and does occur on your network, helping to pinpoint security risks that make exist

6.3.4 IBM Security QRadar QFlow for network and application activity monitoring

The IBM Security QRadar QFlow appliance performs network and application activity monitoring. Network and application activity monitoring is a security fundamental that some organizations do without at their peril. Effective analysis of network session activity, or flow data, involves the collection and detailed classification of network and application behavior. It also involves the ability to correlate network and application activity with log events and other security activities across your entire network.

Application-aware network monitoring enables stateful information about all conversations at the application layer. It also provides a more thorough understanding of complex applications, including voice over IP (VoIP), multimedia, enterprise resource planning (ERP), and database.

IBM Security QRadar QFlow gathers knowledge from a deep examination of every packet within a conversation and provides a more detailed application level context. This information, when correlated with network and security events, enables a more advanced analysis of the overall security posture of the network. Furthermore, the application content that is captured can provide key forensic data and evidence for analyzing the true impact of threats, notably those threats that can include potential data leakage.

6.3.5 IBM Security QRadar VFlow for virtual activity monitoring

Because virtual servers are as susceptible to security vulnerabilities as physical servers, organizations now must define and implement appropriate precautionary measures to protect their applications and data that are within the virtual data center.

With the Security QRadar VFlow for virtual activity monitoring, IT professionals can increase visibility into the vast amount of business application activity that appears across their virtual networks. The Security QRadar VFlow offering helps organizations to better identify these applications for security monitoring, application-layer behavior analysis, and anomaly detection. IBM Security QRadar VFlow can also enable operators to capture application content for deeper security and policy forensics.

6.3.6 IBM Security QRadar Vulnerability Manager

IBM Security QRadar Vulnerability Manager includes an embedded scanning engine that can be set up to run both dynamic and periodic scans, providing near real-time visibility of weaknesses that might otherwise remain hidden. Using the passive asset discovery capabilities of IBM Security QRadar QFlow and Log Collector appliances, any new asset appearing on the network can be immediately scanned. As a result, organizations can reduce their exposure to advanced threats between regular scanning cycles and help ensure compliance with the latest security regulations.

6.3.7 IBM Security QRadar Network Anomaly Detection

IBM Security QRadar Network Anomaly Detection enhances intrusion prevention system (IPS) solutions by providing greater insight into network behavior and abnormal activity to better identify security threats. By correlating IPS alerts, vulnerabilities, network traffic, and threat intelligence, IBM Security QRadar Network Anomaly Detection helps deliver a complete, three-dimensional view of your organization's network activity and security risks.

By using IBM security intelligence solutions, organizations can achieve the following objectives:

- ▶ Detect threats that other solutions miss: Internet-based threats and fraud continue to get more sophisticated. Intelligence is hidden in an organizations' data, which can be used to detect alarming problems. Such problems range from employees stealing proprietary information to botnets trying to break in and steal credit card information or international espionage. IBM security intelligence solutions can help identify the high priority offenses against your corporate data and detect anomalies in user, application, and network behavior.
- ▶ Consolidate data silos: With many companies generating millions or billions of records of events every day, a wealth of information exists in the event and log data that is generated by existing network devices. Unfortunately, this information is frequently stored in silos, often ignored, and often underused. IBM security intelligence solutions can converge the previously distinct network, security, and operations views of the infrastructure into a sensible yet scalable intelligence solution. By using this method, an organization can quickly respond to what is important. It can distill network and security information down to the identity and application awareness level to better and more efficiently resolve network threats and policy infractions.
- ▶ Detect insider fraud: Some of the biggest threats to an organization come from the inside. Organizations often lack the intelligence that is necessary to accurately link individuals to incidents of malicious behavior. With user and application monitoring, organizations can set a baseline for normal user activity, making it easier to identify abnormal or risky behavior and weaknesses.
- ▶ Predict risks against your business: Security and IT teams are constantly challenged to better manage risk across an ever-growing spectrum of vulnerabilities before a breach occurs. IBM security intelligence solutions can provide a preliminary solution that allows for assessing what risks exist during and after an attack. It can also answer many "what if?" questions ahead of time, which can greatly improve operational efficiency and reduce network security risks.
- ▶ Exceed regulatory mandates: Companies today are under growing executive pressure to comply with mandates, such as Sarbanes-Oxley, Good Practice Guide 13 (GPG-13), Financial Services Authority (FSA), Garante, HIPAA, FISMA, GLBA, PCI, and NERC.

The massive amounts of data and events that are being generated in an organization provide the keys to the audit trail. The collection correlation and integration of all surveillance feeds for IBM security intelligence solutions yields the following information:

- ▶ More accurate data for an operator
- ▶ More granular forensics for an incident response manager
- ▶ Complete reporting for auditors

6.3.8 Integration of QRadar SIEM with z/OS

The IBM z/OS DSM for IBM Security QRadar allows you to integrate with an IBM z/OS mainframe using IBM Security zSecure Audit to collect security, authorization, and audit events.

Using a Security zSecure process, events from the System Management Facilities (SMF) are recorded to an event file in the Log Enhanced Event format (LEEF). QRadar retrieves the LEEF event log files using the log file protocol and processes the events. You can schedule QRadar to retrieve events on a polling interval, which allows QRadar to retrieve the events on the schedule you have defined.

Recommended reading: For more information about implementing the connection between QRadar SIEM and z/OS, see *IBM Security zSecure Suite, Installation and Deployment Guide, SC22-5463*.

6.4 IBM Security Key Lifecycle Manager

The IBM approach to key management revolves around IBM Security Key Lifecycle Manager (SKLM), a product that was announced in 2008. Security Key Lifecycle Manager for z/OS simplifies encryption tasks. The Security Key Lifecycle Manager for z/OS supports several storage devices:

- ▶ The IBM System Storage TS1130 Tape Drive (3592 Model E06 and Model EU61). The 3592 EU6 Tape Drive is a 3592 E05 Tape Drive canister that is upgraded to contain a Model E06 drive through the Miscellaneous Equipment Specification (MES) process.
- ▶ The TS1120 Tape Drive (3592 Model E05) can encrypt data as it is written to any size IBM TotalStorage Enterprise Tape Cartridge 3592, including WORM cartridges.
- ▶ The IBM System Storage TS1140 Tape Drive (3592 Model E07) can encrypt data on IBM TotalStorage Enterprise Tape Cartridge 3592 JB/JX and JC/JY media only in supported write format. JA/JW cartridges can read only encrypted or decrypted data.
- ▶ The IBM LTO Ultrium 4 Tape Drive (LTO Ultrium 4) and LTO Ultrium 5 Tape Drive (LTO Ultrium 5) drives can also encrypt data as it is written to any LTO Ultrium 4 and LTO Ultrium 5 Data Cartridges.

Encryption is performed at full line speed in the tape drive after compression. Compression is more efficiently done before encryption. This new capability adds a strong measure of security to stored data. It is added without the processing impact and performance degradation that is associated with encryption that is performed on the server or the expense of a dedicated appliance.

The Security Key Lifecycle Manager for z/OS is part of the IBM Java environment and uses the IBM Java Security components for its cryptographic capabilities. (For more information about the IBM Java Security components, see *System z Crypto and TKE Update*, SG24-7848.) The Security Key Lifecycle Manager for z/OS has four main components that are used to control its behavior:

- ▶ Java security keystore

The keystore is defined as part of the Java Cryptography Extension (JCE). The keystore is an element of the Java Security components, which are, in turn, part of the Java runtime environment. A keystore holds the certificates and keys (or pointers to the certificates and keys) that are used by the Security Key Lifecycle Manager for z/OS to perform cryptographic operations. Several types of Java keystores are supported that offer different operational characteristics to meet your needs.

- ▶ Configuration files

The configuration files enable you to customize the behavior of the Security Key Lifecycle Manager for z/OS to meet the needs of your organization.

- ▶ Device Table

The device table is used by the Security Key Lifecycle Manager for z/OS to monitor the devices it supports. The device table is a non-editable, binary file whose location is specified in the configuration file. You can change its location to meet your needs.

- ▶ KeyGroups.xml file

This password-protected file contains the names of all encryption key groups and the aliases of the encryption keys that are associated with each key group.

6.4.1 Security Key Lifecycle Manager components and resources

IBM Security Key Lifecycle Manager is divided into two major components:

- ▶ Key serving
- ▶ Key and certificate management

Figure 6-13 is a high-level view of the IBM Security Key Lifecycle Manager architecture.

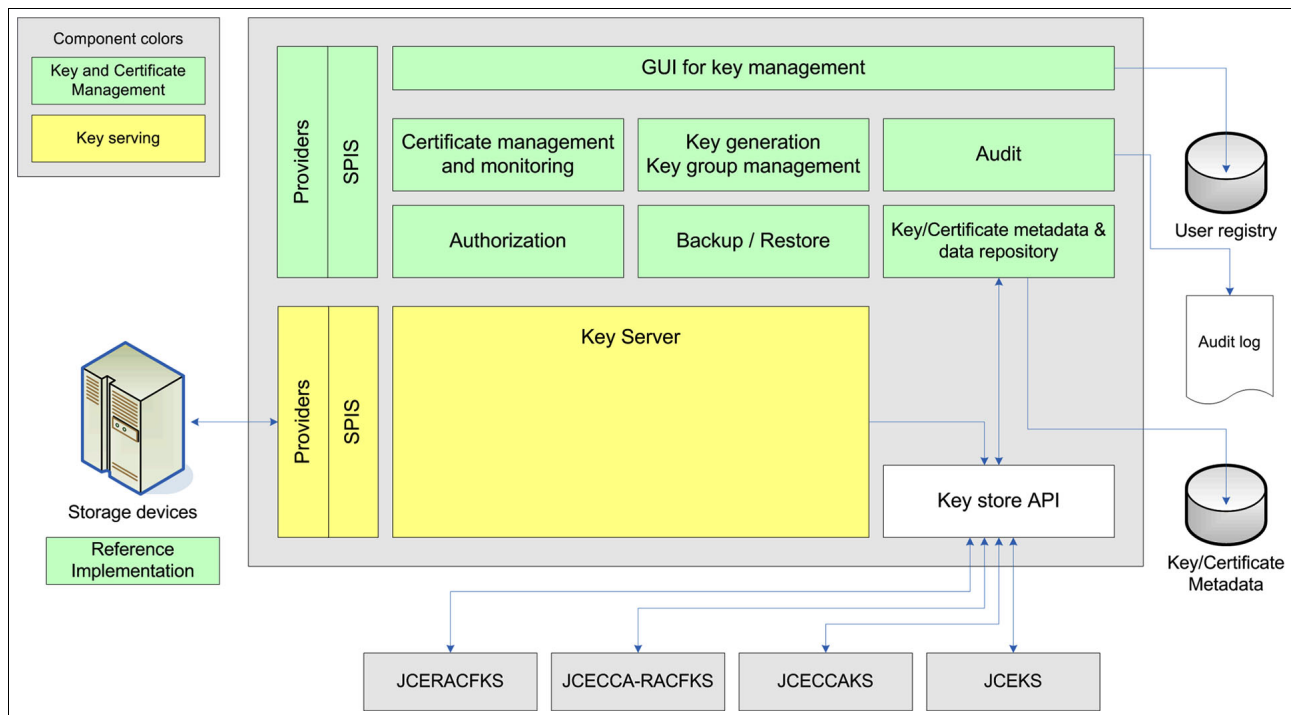


Figure 6-13 IBM Security Key Lifecycle Manager architecture

The *GUI for key management* is the primary user interface into IBM Security Key Lifecycle Manager and provides all the functions for day-to-day key management. The *audit* component creates and sends audit information to an external *audit log* and the *backup/restore* facility provides for backing up of the IBM Security Key Lifecycle Manager keystore and associated metadata to a single protected file. The *key/certificate metadata and data repository* function maintains the metadata that is associated with the keys (such as key identifier and the tape serial number) and stores it in a DB2 database. Keys are generated and key grouping is maintained by the *key generation, key group management* component, which also manages key group rotation. *Certificate management and monitoring* handles certificate management, for example, the creation of certificates or the obtaining of them from third parties, checking the validity of certificates (that they have a valid root certificate and have not expired), and managing their renewal.

IBM Security Key Lifecycle Manager can be implemented to serve keys to only predefined storage devices, and the *authorization* function validates these devices as necessary. It is the IBM stated intention to provide an API (the *providers service provider interfaces (SPIs)*) to enable communication with other key managers.

The key serving component is responsible for obtaining the correct key from the keystore using the *keystore API* and delivering it securely to the encrypting storage device through the *Providers SPIs*. In addition, by implementing the *reference implementation*, third parties can have their devices participate in IBM Security Key Lifecycle Manager key management.

6.4.2 Encryption-enabled 3592 and LTO tape drives

Security Key Lifecycle Manager for z/OS supports encryption-enabled 3592 and LTO tape drives. Drives without encryption enablement are not supported. Security Key Lifecycle Manager for z/OS supports these drive types:

- ▶ 3592 tape drives
- ▶ TS1120, TS1130, and TS1140 tape drives that are enabled to encrypt data.
- ▶ LTO.
- ▶ LTO Ultrium 4 and LTO Ultrium 5 tape drives that are enabled to encrypt data.
- ▶ Encryption is performed at full line speed in the tape drive after compression.

6.4.3 Enterprise storage: IBM System Storage DS8000 (2107, 242x)

Security Key Lifecycle Manager for z/OS supports the DS8000 Storage Controller. This support requires the appropriate microcode bundle version on the DS8000 Storage Controller, Licensed Internal Code level 64.2.xxx.0 or higher.

6.4.4 Managing encryption

The Security Key Lifecycle Manager for z/OS is a Java software program. This product assists IBM encryption-enabled devices. The product assists in generating, protecting, storing, and maintaining encryption keys. Those keys are used to encrypt information being written to, and decrypt information being read from, tape media (tape and cartridge formats) and system storage devices. This product is designed to run in the background as a shared resource deployed in several locations within an enterprise. The product can serve numerous IBM encrypting tape drives and system storage devices regardless of where those devices are. These devices can be in tape library subsystems, which are connected to mainframe systems through various types of channel connections, or installed in other computing systems. A command-line interface client provides a robust set of commands to customize the Security Key Lifecycle Manager for z/OS for your environment and monitor its operation. The product uses one or more keystores to hold the certificates and keys (or pointers to the certificates and keys). These keystores are required for all encryption tasks. The Security Key Lifecycle Manager for z/OS supports IBM keystores such as JCEKS, JCECCAKS, JCECCARACFKS, and JCERACFKS.

For more information, see “Keystore Considerations” at the following link:

http://www-01.ibm.com/support/knowledgecenter/api/redirect/tivihelp/v2r1/topic/com.ibm.tivoli.isklm.doc_11/top_EKMipug_plan_keystore.html

Attention: The product performs the function of requesting the generation of encryption keys and passing those keys to TS1120, TS1130, TS1140, or LTO Ultrium 4 and LTO Ultrium 5 tape drives and DS8000. The key material, in wrapped (encrypted) form, is in system memory during processing by the Security Key Lifecycle Manager for z/OS. The key material must be transferred without error to the appropriate tape drive so that data written on a cartridge can be recovered (decrypted). If a corrupted key material is used to write data to a cartridge, then data that is written to that cartridge is not recovered. There are safeguards in place to make sure that such data errors do not occur. If the machine hosting the Security Key Lifecycle Manager for z/OS is not using Error Correction Code (ECC) memory, it is possible that the key material can become corrupted while in system memory. The corruption can then cause data loss. Although the risk is slight, it is best to host the Security Key Lifecycle Manager for z/OS using ECC memory.

The Security Key Lifecycle Manager for z/OS acts as a background process. The product waits for key generation or key retrieval requests. The requests are sent to it through a TCP/IP communication path between itself and the tape library, tape controller, tape subsystem, device driver, or tape drive. When a tape drive writes encrypted data, it first requests an encryption key from the Security Key Lifecycle Manager for z/OS. Upon receipt of the request, the Security Key Lifecycle Manager for z/OS performs the tasks that are listed in the following sections.

For TS1120, TS1130, and TS1140 tape drives

The Security Key Lifecycle Manager for z/OS generates an Advanced Encryption Standard (AES) key and serves it to the tape drives in two protected forms:

- ▶ Encrypted or *wrapped*, using Rivest-Shamir-Adleman (RSA) key pairs. TS1120, TS1130, and TS1140 tape drives write this copy of the key to the cartridge memory. The key also is copied in three additional places on the tape media in the cartridge for redundancy.
- ▶ They are separately wrapped for secure transfer to the tape drive. The keys are unwrapped upon arrival, and the key inside is used to encrypt the data that is written to tape.

When an encrypted tape cartridge is read by a supported tape drive, the protected AES key on the tape is sent to the Security Key Lifecycle Manager for z/OS. The wrapped AES key is unwrapped in the Security Key Lifecycle Manager for z/OS. The AES key is then wrapped with a different key for secure transfer back to the tape drive. The AES key is unwrapped in the tape drive and used to decrypt the data that is stored on the tape. The Security Key Lifecycle Manager for z/OS also allows protected AES keys to be rewrapped or rekeyed. Different RSA keys can be used from the originals when the tape was written. Rekeying is useful when an unexpected need arises to export volumes to Business Partners whose public keys were not included. This method eliminates rewriting the entire tape. This method also enables the data key of a tape cartridge to be reencrypted with the public key of the Business Partner.

For LTO Ultrium 4 Tape Drive and LTO Ultrium 5 Tape Drive

The Security Key Lifecycle Manager for z/OS fetches an existing AES key from a keystore. The product then wraps the AES key for secure transfer to the tape drive where it is unwrapped upon arrival. The key is used to encrypt the data being written to tape.

When an encrypted tape is read by an LTO Ultrium 4 or LTO Ultrium 5 Tape Drive, the Security Key Lifecycle Manager for z/OS fetches the required key from the keystore. The key is fetched based on the information in the Key ID on the tape and serves it to the tape drive wrapped for secure transfer.

For DS8000

When the DS8000 starts, the device requests an unlock key from Security Key Lifecycle Manager for z/OS.

If the DS8000 requests a new key for its unlock key, Security Key Lifecycle Manager for z/OS generates an Advanced Encryption Standard (AES) key and serves the key to the drive in two protected forms:

- ▶ Encrypted (wrapped) using Rivest-Shamir-Adleman (RSA) key pairs. The DS8000 stores this copy of the key on the array in an unencrypted partition.
- ▶ Separately wrapped for secure transfer to the drive, where it is unwrapped upon arrival and the key inside is used to unlock the array.

If the DS8000 requests an existing unlock key, the protected AES key on the array is sent to Security Key Lifecycle Manager for z/OS, where the wrapped AES key is unwrapped. The AES key is then wrapped with a different key for secure transfer back to the DS8000, where it is unwrapped and used to unlock the array.

6.5 IBM Enterprise Key Management Foundation

The IBM Enterprise Key Management Foundation (EKMF) is a flexible and highly secure key management system for the enterprise. It provides centralized key management on IBM zEnterprise and distributed platforms for streamlined, efficient, and secure key and certificate management operations. The Enterprise Key Management Foundation is suitable for banks, payment card processors, and other businesses that must meet Europay, MasterCard, and Visa (EMV) and payment card industry (PCI) requirements. It includes cryptoanalytic capabilities to identify compliance issues and help key officers understand who has access to key material. The Enterprise Key Management Foundation provides a foundation that can be tailored to address the needs of multiple industry segments to identify compliance issues and help key officers enforce enterprise key management policy requirements.

6.5.1 The solution architecture

There are three basic components that make up an IBM Enterprise Key Management Foundation-based solution:

- ▶ The workstation

All key management operations that involve human interaction are carried out at the workstation. The workstation uses an IBM 4765 Cryptographic Co-processor that is installed for secure generation of the keys.

The key management application is installed on the workstation, and operators can use smart cards for authentication to the application and the IBM 4765, if two-factor authentication is required. The smart cards also can be used to store the master key of the IBM 4765. If keys must be printed, a printer can be attached to the workstation.

- ▶ The key repository

Keys and their metadata are stored in an IBM DB2 database. This database can be deployed on any server in the organization, although it is typically deployed on an IBM z/OS system when this platform is available, or on one of the servers that also provide an IBM 4765. The workstation connects to the database using Java Database Connectivity (JDBC) or an agent. The network protocol is always TCP/IP.

- ▶ The agents

The agents are installed on servers, where either the key repository is or where there are keystores to be managed. An agent that manages only keystores is referred to as a *crypto agent*, and an agent that manages the repository is called a *database agent*. The crypto agent is required to push a key to the Integrated Cryptographic Service Facility (ICSF) on z/OS or an IBM 4765 on one of the other platforms because ICSF and the IBM 4765 do not provide a network-based interface, as shown in Figure 6-14 on page 148. The agent on z/OS image 1 in the figure serves both as database and crypto agent, and is denoted as a combined agent in the figure.

Additionally, a crypto agent can provide private keys and certificates to IBM RACF key rings. The option to log in to z/OS SMF is available using either a crypto agent or a database agent. An agent is either installed as a started task on z/OS or as a service on the other available platforms. The IBM WebSphere DataPower solution exposes a *web service interface* that can be used for managing the keys and certificates; an agent is not needed in this case.

The components of this architecture are shown in Figure 6-14.

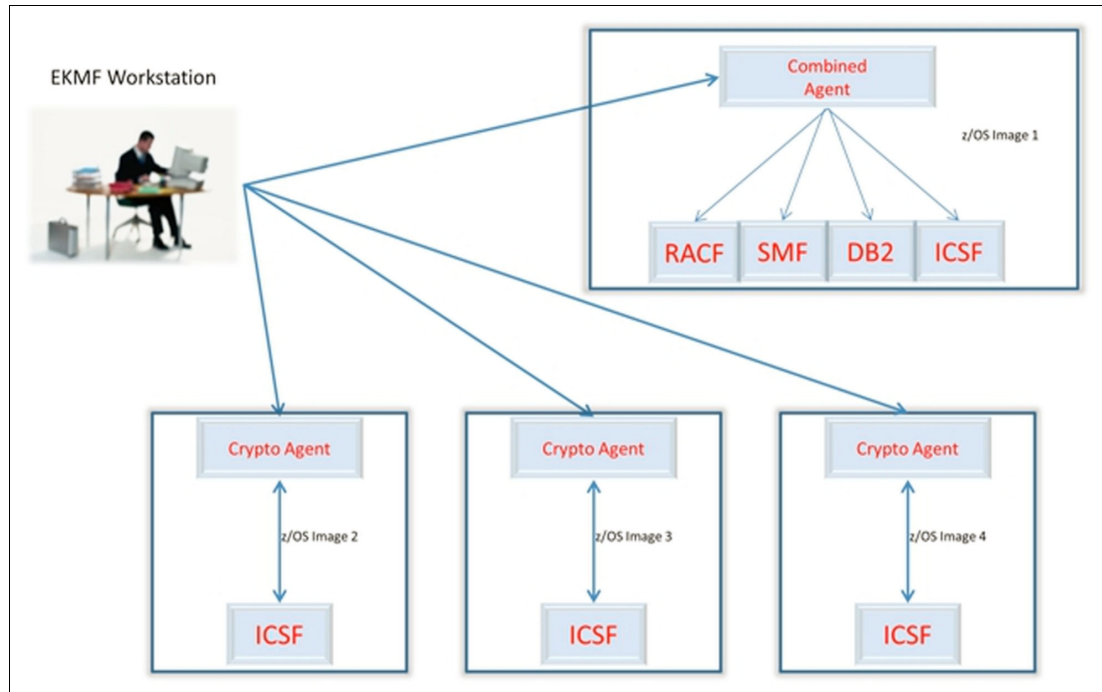


Figure 6-14 Components of IBM Enterprise Key Management Foundation

6.5.2 Centralized key management using the IBM Enterprise Key Management Foundation

The IBM Enterprise Key Management Foundation is built around a workstation that uses a key management application and an IBM 4765 Cryptographic Co-processor for secure key generation. The other main component in the IBM Enterprise Key Management Foundation is the key repository, which is an IBM DB2 database that is often deployed on an IBM z/OS system. From the workstation, keys are pushed to the supported platforms that are shown in Figure 6-15 on page 149.

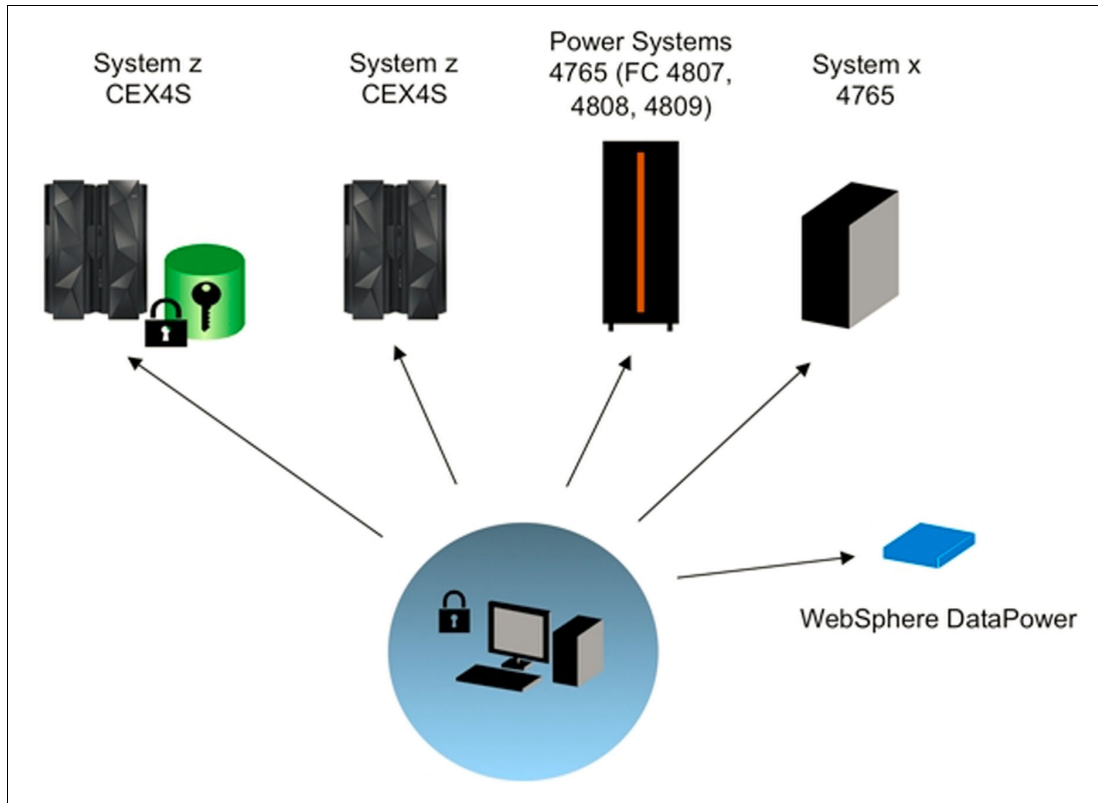


Figure 6-15 IBM Enterprise Key Management Foundation

The IBM Enterprise Key Management Foundation provides business value for all organizations that must manage and maintain encryption keys in their IT infrastructure in the following categories:

- ▶ Centralized key management

The IBM Enterprise Key Management Foundation provides centralized key management for the organization, concentrating the key management effort in a single organizational entity, with trained personnel that use a common set of procedures.

- ▶ Compliance

This IBM technology has been used for more than 20 years by financial institutions that needed to be in compliance with the card association's rules for processing PIN-based transactions. The solution is policy-driven, and an organization can configure a setup where administrators create templates for keys that can be used over and over again when renewing keys for IT applications. Furthermore, it can be configured to require dual control for all or exactly those operations that organizations require.

When dealing with key parts, the system tracks which users have handled which key parts. All key management operations are logged in the systems database and optionally in the z/OS System Management Facility (SMF).

► Business-oriented solutions

Beyond the basic key management capabilities, the IBM Enterprise Key Management Foundation offers many solutions for various business purposes:

- ATM Remote Key Loading (RKL)

For online distribution of automatic teller machine (ATM) master keys, the IBM Enterprise Key Management Foundation provides support for Rivest-Shamir-Adleman (RSA), as shown in Figure 6-16.

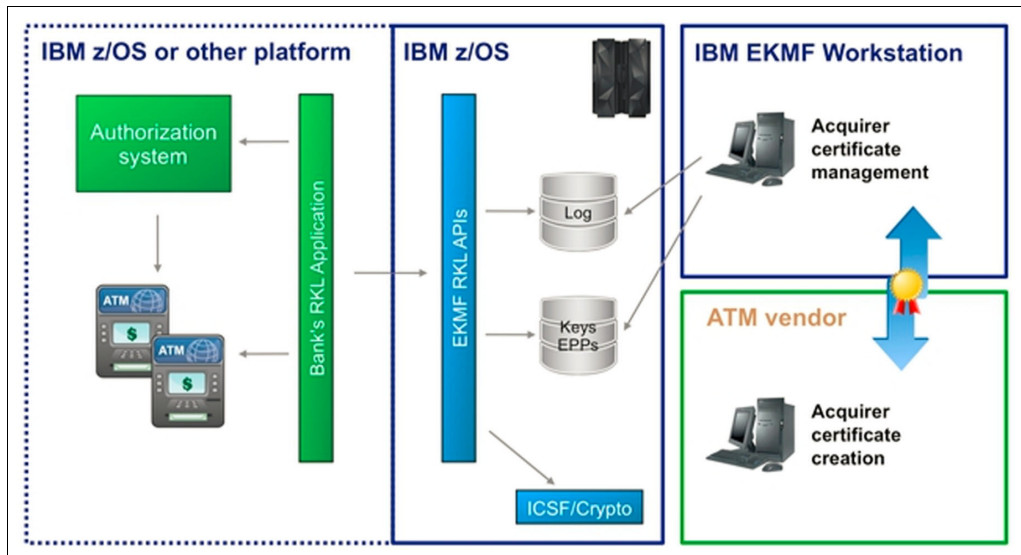


Figure 6-16 IBM Enterprise Key Management Foundation with ATM

- Europay, MasterCard, and Visa (EMV)

The EMV organization has created a set of standards for chip-based payment cards. The standards use both symmetric and asymmetric encryption techniques, as shown in Figure 6-17.

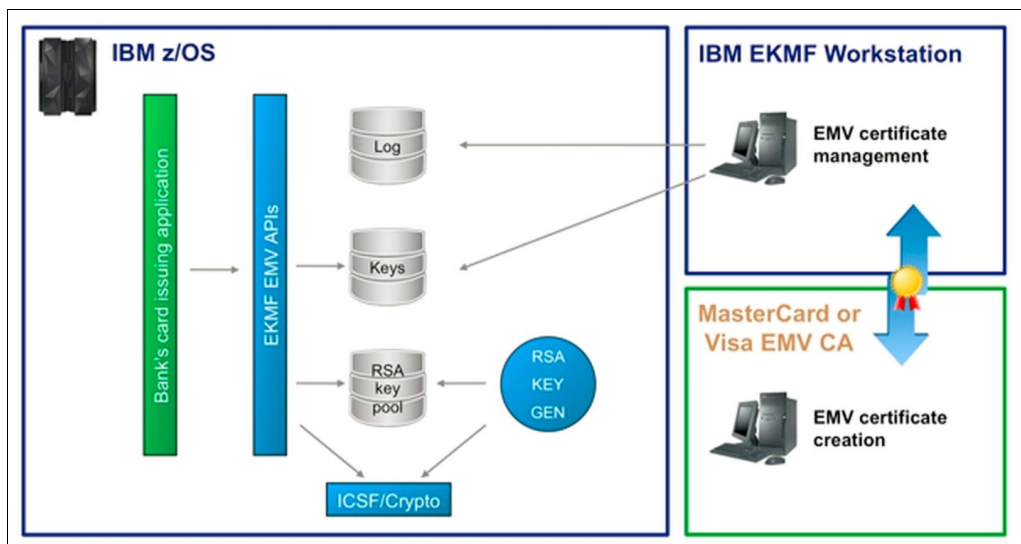


Figure 6-17 IBM EKMF with MasterCard and Visa EMV CA

The IBM Enterprise Key Management Foundation enables an organization to manage the involved certificates to generate the asymmetric keys to stock all your cards in batch processes, and to encapsulate the issuing and authorization functions in application programming interfaces (APIs). Figure 6-17 on page 150 shows this function.

6.5.3 Certificate management

The ever-growing use of cryptography for securing transactions has resulted in an intensive use of certificates. Certificates must be managed, particularly because they have a limited lifetime.

Certificate expiration can often be linked as the direct cause of IT application or server downtime.

To address this issue, a certificate management model has been developed for the IBM Enterprise Key Management Foundation. The model is shown in Figure 6-18, and consists of four phases:

- ▶ In the discovery phase, the network and servers are scanned for certificates.
- ▶ In the enrollment phase, discovered certificates are checked against the system's database and new certificates can be enrolled in the system.
- ▶ The monitoring phase is an ongoing phase, where the database is scanned for certificates that will expire in the near future. Notifications are issued for expiring certificates.
- ▶ In the management phase, the certificates are renewed and installed on the servers, the old certificate is decommissioned, and the new certificate enters the monitoring phase. The IBM solution provides the capability to issue X.509V3 certificates for internal use.

To assist applications in public key infrastructure (PKI)-related functions requiring certificates, a number of APIs can be provided to help with verifying certificate chains and building structures according to Public Key Cryptography standards PKCS#1 and PKCS#7, as shown in Figure 6-18.

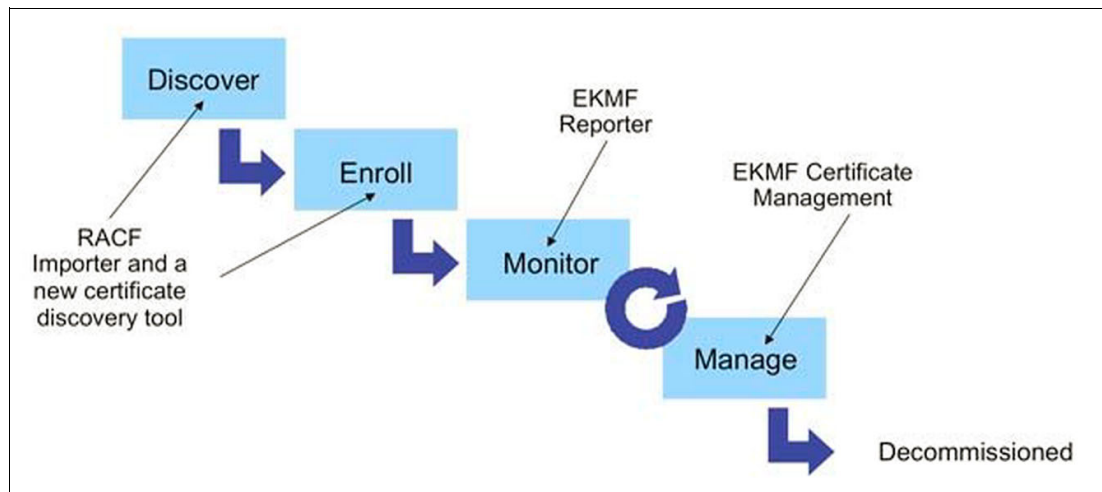


Figure 6-18 IBM Enterprise Key Management Foundation API

6.6 Encryption Facility for z/OS

The need for creating secure archived copies of business data is a critical security concern. Encrypting data that can be recovered at any time offers a high degree of privacy protection from unwanted access. Encryption Facility for z/OS provides this protection by offering encryption of data for exchange between different systems and platforms and for archiving purposes. It uses hardware compression and encryption and relies on a centralized key management that is based on the z/OS Integrated Cryptographic Service Facility (ICSF), which is highly secure and easy to use. Encryption Facility for z/OS can use ICSF to perform encryption and decryption and to manage cryptographic keys. To encrypt data files, Encryption Facility for z/OS uses the cryptographic keys of different kinds, such as TDES triple-length keys and 128-bit AES keys. For Encryption Facility for OpenPGP of Encryption Services, 128-bit, 192-bit, and 256-bit AES keys are used.

6.6.1 Encryption Services Feature

The Encryption Services feature supports both the System z format (originally introduced in Encryption Facility for z/OS V1.1) and the OpenPGP format (part of Encryption Facility for z/OS V1.2). The System z format supports hardware-accelerated compression before encryption. The Encryption Services feature includes batch programs CSDFILEN (to encrypt z/OS data) and CSDFILDE (to decrypt z/OS data), and Encryption Facility for OpenPGP to support the OpenPGP standards that are described in RFC 2440.

6.6.2 IBM Encryption Facility for z/OS Client

The Encryption Facility for z/OS Client is a no-cost, separately licensed program that is offered *as is*, with no warranty, and is designed to enable the exchange of encrypted data between z/OS systems that have the Encryption Facility for z/OS installed and systems running on z/OS and other platforms that require the supported functions. The Encryption Facility for z/OS Client consists of the Java-based Client and the Decryption Client for z/OS.

6.6.3 DFSMSdss Encryption

The DFSMSdss Encryption feature enables the encryption of DFSMSdss DUMP data sets. You can decrypt the data through the **RESTORE** command. With this feature, you can also use the DFSMSHsm memory dump class settings to encrypt data that is dumped through the **BACKVOL DUMP** command and automatic memory dump processing, and decrypt the data through the **RECOVER** command. This feature supports hardware-accelerated compression before encryption to tape.

6.6.4 Encryption Services batch programs CSDFILEN and CSDFILDE

Encryption Services batch programs CSDFILEN and CSDFILDE can use a symmetric key that is randomly generated to protect System z format data. CSDFILEN encrypts this data key and stores it with the data in a header record. Encryption Services allows you to protect the data key by using public key architecture, by providing an RSA key label, or, if an RSA public/private key pair is not available, by using a key that is generated from a password that you provide. The encrypted data contains a header with enough information to use for decryption. You can use the CSDFILDE batch program to decrypt the z/OS data. Decryption Client for z/OS, which is part of the Encryption Facility for z/OS Client, can also use CSDFILDE to decrypt data on a z/OS platform, or you can send the encrypted data to a Java platform where you can use the Java-based Client, also part of Encryption Facility for z/OS Client, to decrypt the data. You code job control statements (JCL) to control CSDFILEN encryption and CSDFILDE decryption services. You can optionally specify that the data is compressed before encryption and that the encrypted data, which is an output sequential file, is sent to tape for archiving or transfer.

6.6.5 Encryption Facility for OpenPGP

Encryption Facility for OpenPGP can encrypt and decrypt z/OS-type data for use with OpenPGP-compliant systems and OpenPGP-compliant messages and files. It includes support for the following items:

- ▶ Passphrase base encryption of session key
- ▶ Digital signatures of data
- ▶ Importing/exporting of OpenPGP certificates
- ▶ RSA, ElGamal, and DSA key generation
- ▶ use of partial data packets
- ▶ ASCII Armor for OpenPGP certificates
- ▶ Data encryption with a randomly generated symmetric session key
- ▶ Symmetric encryption of a randomly generated symmetric session key
- ▶ Asymmetric encryption of randomly generated symmetric keys
- ▶ Digest/Hash using SHA-1, MD5, MD2, SHA-256, SHA-384, and SHA-512 algorithms
- ▶ Digital Signature using DSA with SHA1 and RSA algorithm

6.7 IBM Security AppScan

IBM Security AppScan® is a suite of web application security testing products that are used to automate application scanning and vulnerability identification. The Security AppScan products scan and test for a wide range of web application vulnerabilities, including those identified by the Web Application Security Consortium (WASC) threat classification and the Open Web Application Security Project. The Security AppScan product line contains a wide variety of products, each of them adapted to the needs of a specific user.

6.7.1 Main editions of IBM Security AppScan

IBM Security AppScan is available in five different versions.

IBM Security AppScan Developer Edition

The first user that is targeted by the IBM Security AppScan product line is the developer. The most efficient way to stay ahead of application security vulnerabilities is to build software securely from the ground up. The challenge is that most developers are not security experts, and writing secure code is not always their top priority. Therefore, a good way to engage development in the process of application security is to provide them with tools that work in their environment and that generate results in languages they understand.

IBM Security AppScan Developer Edition is designed to empower developers to invoke web application security testing from within their development environment. It enables addressing the volume of security issues that can be introduced in code at an early point, streamlining the development lifecycle workflow, and helping to reduce security testing bottlenecks that can occur at the end of the release cycle. IBM Security AppScan Developer Edition uses a range of analysis techniques to accurately pinpoint security issues in your web applications, including static code analysis, dynamic analysis, runtime analysis, and string analysis

The Developer Edition is designed with automation in mind. Automating code analysis configuration for greater ease of use and accuracy is realized by using *string analysis*, which is a new technology invented in collaboration with IBM Research. String analysis presents a breakthrough in the field of static code analysis by helping to solve the biggest challenge that plagues current security code-scanning solutions, which is false positives. Until now, the most advanced technique for analyzing code security, taint analysis, tracked input values as they flow through the system. However, it relies on the developer to determine whether the data is properly sanitized by marking the sanitization functions.

IBM Security AppScan Build Edition

An additional way to use the vulnerability detection capabilities of IBM Security AppScan Developer Edition vulnerability scanning engines is to perform automated scanning during the builds. Integrating security into the build phase can be done by using IBM Security AppScan Build Edition. By integrating with multiple build management systems, such as IBM Rational® Build Forge® software, IBM Security AppScan Build Edition provides security testing coverage for scheduled builds. It includes the same set of analysis techniques as the IBM Security AppScan Developer Edition, and provides a high level of accuracy plus code coverage that helps you identify which code has been tested.

After scanning, IBM Security AppScan Build Edition routes the results back to development through defect-tracking solutions, such as IBM Rational ClearQuest® software, or through security reporting solutions, such as IBM Security AppScan Enterprise Edition or IBM Security AppScan Reporting Console. IBM Security AppScan Build Edition also includes an API and various other result formats to support the propagation of scan results to other repositories.

IBM Security AppScan Standard Edition

The next user that IBM Security AppScan product considers is the security auditor. To help this user, the IBM Security AppScan Standard Edition was created. To allow the security auditor to automate testing of the latest technologies, IBM Security AppScan Standard Edition supports the latest Web 2.0 technologies, parsing and execution of JavaScript and Adobe Flash applications, asynchronous JavaScript and XML (Ajax) and Adobe Flex-related protocols, such as JavaScript Object Notation (JSON), Action Message Format (AMF), and SOAP, elaborate service-oriented architecture (SOA) environments, and custom configuration and reporting capabilities for mashups and process-driven applications.

One of the most critical aspects of web vulnerability scanning is the quick remediation of issues. IBM Security AppScan Standard Edition provides a fully prioritized list of the vulnerabilities that are found with each scan, which enables high-priority problems to be fixed first, helping organizations to focus on what matters the most from a security perspective. Each vulnerability result includes a full description of how the vulnerability works and the potential causes. Integrated web-based training provides short training modules directly from the user interface. The software's remediation view then explains the steps that are required to remediate the issue, including examples of both secure and insecure code.

IBM Security AppScan Standard Edition can also help organizations address critical compliance requirements, such as Payment Card Industry Data Security Standard (PCI DSS), by providing a way to support an ongoing level of application security. IBM is an approved scanning vendor (ASV) with its IBM Security AppScan Standard Edition offering, making the software a perfect choice for addressing application security requirements around PCI DSS.

IBM Security AppScan Standard Edition can produce custom security reports and includes the ability to select which data points to include in each report. Users can also choose from more than 40 predefined reports and map scan results to key industry and regulatory compliance standards. These include National Institute of Standards and Technology Special Publication (NIST SP) 800-53 and the Open Web Application Security Project top 10, PCI DSS, Sarbanes-Oxley, Gramm-Leach-Bliley Act (GLBA), Health Insurance Portability and Accountability Act (HIPAA), Family Educational Rights and Privacy Act (FERPA), Freedom of Information and Protection of Privacy Act (FIPPA), and Payment Application Best Practices (PABP).

IBM Security AppScan Tester Edition

By using the same vulnerability detection capabilities as IBM Security AppScan Standard Edition, IBM Security AppScan Tester Edition, which is available as a desktop application, offers capabilities to help quality assurance (QA) teams integrate security testing into existing quality management processes, thus easing the burden on security professionals. IBM Security AppScan Tester Edition integrates with leading testing systems, thus allowing QA professionals to use its functions in testing scripts. They can conduct security checks within their familiar testing environments, facilitating the adoption of security testing along with functional and performance testing.

Given that QA organizations already know how to prioritize defects, make testing repeatable, and report on test coverage and release readiness, these teams are ideally suited to perform security testing. They can help scale security testing and find and remediate security vulnerabilities earlier in the application delivery process. They can do this while they are testing for functional and performance issues and prevent costly delays in your releases.

IBM Security AppScan Enterprise Edition

IBM Security AppScan Enterprise Edition is a web-based, multi-user web application vulnerability testing and reporting solution for organizations that must scale application scanning across the organization while maintaining centralized control of vulnerability data. IBM Security AppScan Enterprise Edition includes QuickScan, a point-and-shoot testing tool, and integrated Computer Based Training to accelerate the adoption of security testing across the software development lifecycle.

IBM Security AppScan Enterprise provides centralized control with new advanced application scanning, remediation capabilities, executive security metrics and a dashboard, key regulatory compliance reporting and seamless integration with AppScan Standard, and provides User Administration capabilities for AppScan Source.

The IBM Security AppScan scan engine operates by traversing a web application, analyzing and testing the application for security and compliance issues, and generating actionable reports with fix recommendations to simplify the remediation process. These advanced fix recommendations deliver unmatched accuracy and efficiencies for developers and security auditors to help address and remediate vulnerabilities that are disclosed by the scan. Seamless integration with leading QA testing tools (including IBM Rational ClearQuest) also is provided.

IBM Security AppScan Enterprise provides the following features:

- ▶ A scalable, enterprise architecture that enables centralized scanning of multiple applications simultaneously.
- ▶ Intelligent fix recommendations to ease the process of remediation after security vulnerabilities are identified and validated.
- ▶ Ability to scan websites for both embedded malware and links to malicious or undesirable sites to ensure that your website is not infecting visitors or directing them to unwanted or dangerous sites without their knowledge.
- ▶ Continuous monitoring and aggregation of metrics to ensure remediation and trend improvement over time.
- ▶ Addition of a web services API enabling integration with IBM Rational Insight™.
- ▶ Sophisticated dashboards and flexible reporting views to provide enterprise-wide visibility of risks and remediation progress. It offers the lowest false positive rate in the industry, while finding the most severe security issues.
- ▶ Ability to test sequential business logic, such as opening a new account or making an online purchase.
- ▶ Over 40 security compliance reports, including PCI Data Security Standard, Payment Application Data Security (PA-DSS), ISO 27001 and ISO 27002, HIPAA, GLBA, and Basel II
- ▶ Role-based reporting access and scan permissions to help enforce test policies and to centralize vulnerability scanning.

Version 8.7 focuses on addressing scalability and performance, simplifying the installation and ensuring compliance with US Federal Government requirements. IBM Security AppScan Enterprise v8.7 is easier to install, meets FIPS 140-2 requirements, and scales and performs better than the previous version of the product.

6.8 IBM Security Access Manager

IBM Security Access Manager is an authentication and authorization solution for corporate web, client/server, and existing applications. Security Access Manager enables you to control user access to protected information and resources. By providing a centralized, flexible, and scalable access control solution, Security Access Manager enables you to build secure and easily managed network-based applications and e-business infrastructure. Security Access Manager supports authentication, authorization, audit and logging, data security, and resource management capabilities.

Security Access Manager provides the following frameworks:

- ▶ *Authentication framework:* Security Access Manager provides a wide range of built-in authenticators and supports external authenticators.
- ▶ *Authorization framework:* The Security Access Manager authorization service, which is accessed through a standard authorization API, provides permit and deny decisions on access requests for native Security Access Manager servers and other applications.

The authorization service, together with resource managers, provides a standard authorization mechanism for business network systems. Security Access Manager can be integrated into existing and emerging infrastructures to provide a secure and centralized policy management capability.

6.8.1 Security Access Manager architecture

A user registry and an authorization service are the fundamental building blocks upon which Security Access Manager builds to provide its security capabilities. All other Security Access Manager services and components are built on this base.

The Security Access Manager environment requires certain basic capabilities for administrative control of its functions. Management facilities are provided through the following base components:

- ▶ The Policy Server, which supports the management of the authorization database and its distribution to Authorization Services.
- ▶ A Policy Proxy Server, which provides a mechanism for resource managers to access Policy Server functions without a direct connection to the master Policy Server.
- ▶ The pdadmin utility, which provides a command-line capability for performing administrative functions, such as adding users or groups.
- ▶ The Web Portal Manager, which provides a browser-based capability for performing most of the same functions that are provided by the pdadmin utility.
- ▶ The administration API, on which the pdadmin utility and the Web Portal Manager are built, enables performance of program-initiated level administration tasks and queries.

Security Access Manager requires a user registry to support the operation of its authorization functions. Specifically, it provides the following items:

- ▶ A database of the user identities that are known to Security Access Manager
- ▶ A representation of groups in Security Access Manager that may be associated with users
- ▶ A data store of other metadata that is required to support authorization functions

A high-level view of the Security Access Manager implementation is shown in Figure 6-19. The core part of Security Access Manager (that is, the Policy Server) does not itself protect application-owned resources; it maintains the user registry and the protected object database (also called the object/ACL database or the authorization database).

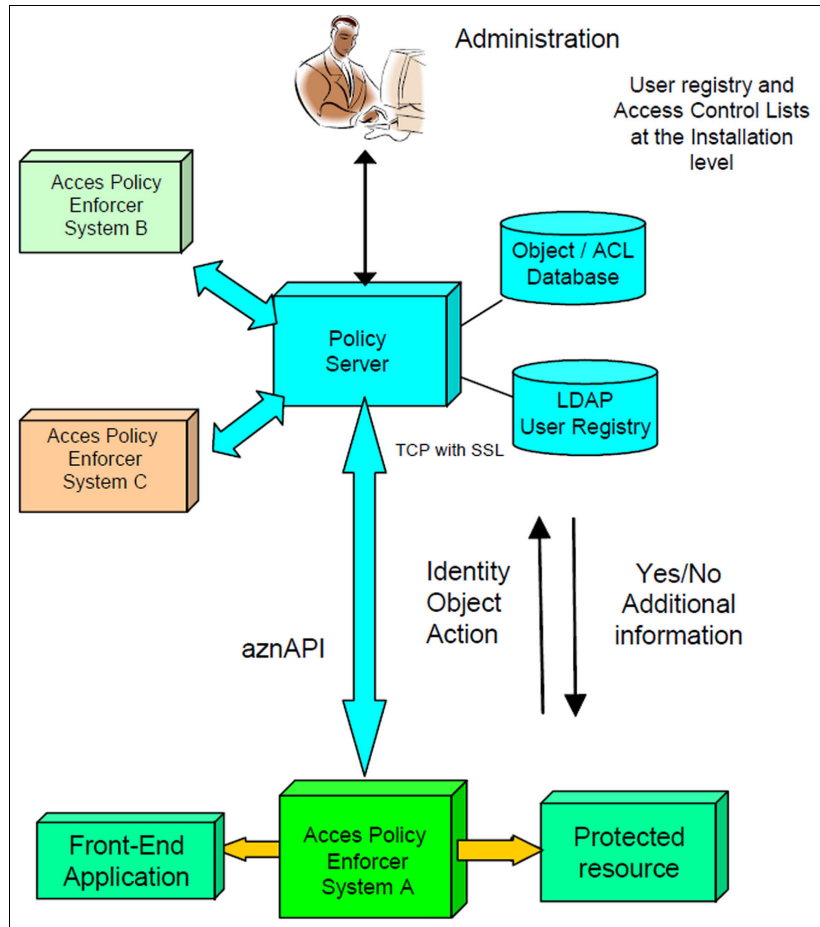


Figure 6-19 Security Access Manager architecture

Application resources are protected using *policy enforcers* that are between the front-end application that the user interacts with and the back-end application, or system that contains the resources that require protection. The policy enforcer is essentially a plug-in program that is installed in the path between the requesting application and the application or system that hosts the resource. IBM provides policy enforcers, and users also can design policy enforcers of their own.

When the policy enforcer receives a request from the front-end application, it uses the aznAPI to query the authorization database. It sends, through TCP/IP, information about the user making the request, the action being requested, and the object to be accessed. This results in a yes or no response, which indicates whether the request should be accepted or rejected. The response also can contain data pertaining to a PoP, if there is one.

The aznAPI is an Open Group standard for authorization request API that is intended to be application- and platform-neutral, which isolates the requesting applications from the complexity of the access control decision-making processes.

All communication between the Security Access Manager components can be protected with SSL/TLS.

As shown in Figure 6-19 on page 158, the Security Access Manager Policy Server can serve requests coming from policy enforcers on different systems, thus sharing the access policy in the protected object database between applications on these systems.

In some cases, installations might install a policy enforcer that acts as a proxy agent (that is, an independent entity from the front-end and back-end systems). In other cases, the front-end, back-end, and policy enforcer might be different parts of the same application.

As a summary, Security Access Manager implementations achieve the following goals:

- ▶ Maintain a central user registry: Users and groups, and authentication information.
- ▶ Maintain a model of the protected objectspace: Hierarchically organized.
- ▶ Define permitted actions on objects: Uses access control list templates. These are attached to entries in the objectspace.
- ▶ Provide an API for making authorization queries.
- ▶ Provide APIs for Security Access Manager administration.

Security Access Manager supports various LDAP-based directories, which also includes the IBM Security Directory Server on non z/OS or z/OS systems. When using the z/OS LDAP server as the Security Access Manager user registry, the schema files that are supplied in z/OS (`schema.user.ldif` and `schema.IBM.ldif`) already include the specific Security Access Manager user registry object classes and attribute types.

Resource managers are program components that provide access manager authorization support for specific application types. The resource manager is responsible for the enforcement of the security policy within an access manager environment. The resource manager uses the policy enforcer to call the Security Access Manager authorization service with the credentials of the user making the request, the type of access wanted, and the object to be accessed. The resource manager takes the recommendation of the authorization service, performs any additional verification actions, and ultimately either denies the request or permits the request to be processed.

6.8.2 IBM Security Access Manager for Enterprise Single Sign-On

IBM Security Access Manager for Enterprise Single Sign-On introduces a new level of security, authentication, and automation experience to business enterprise users on their desktop applications. On a day-to-day basis, the number of resources or applications that a business user accesses varies and is inevitably increasing. Applications that a user employs during normal daily activities might require a range of elements to authenticate or verify the user's identity before granting access to corporate information.

The classic authentication approach is the unique user name and password combination. Each desktop application might require its own unique set of user name and password credentials. The challenge that users are faced with is the need to remember every set of unique credentials for different applications.

IBM Security Access Manager for Enterprise Single Sign-On offers users an experience that eliminates the need to remember and manage multiple sets of user names and passwords. Through the ease of AccessProfiles, this solution can capture and manage credential information for a range of supported application types. This increases user efficiency by making daily activities more convenient, and it efficiently decreases costs for business organizations to address password management issues, supports the need to manage business risks, and ensures that sufficient security and regulatory compliance are in place.

IBM Security Access Manager for Enterprise Single Sign-On offers efficient sign-on solutions and automation of workflow for existing applications as they are. No modifications are required to the existing targeted systems, platforms, or applications where the product is deployed.

So, the format of user name and password logon information can differ between applications. Access Manager for Enterprise Single Sign-On uses the concept of authentication services to represent and map to the different formats that are used. In some cases, AccessProfiles require specific engineering to accommodate complex application credential structures and authentication logic.

6.9 IBM Security Identity Manager

IBM Security Identity Manager enables organizations to drive effective Identity Management and governance across the enterprise. This solution helps strengthen regulatory compliance and security by reducing the risk of identity fraud. It automates the creation, modification, recertification, and termination of user privileges and supports policy-based password management throughout the user lifecycle. It features a redesigned business-friendly user interface and reporting tools to help managers make better governance decisions. As part of IBM Security Systems portfolio, IBM Security Identity Manager helps provide intelligent identity and access assurance.

6.9.1 IBM Security Identity Manager entities

Security Identity Manager's role is to manage users and their accounts. Passwords, group memberships, and other attributes are associated with the users and accounts. These all relate to managed systems and applications. To enable management of users, accounts, and associated information, Security Identity Manager uses an organizational tree and roles, ACLs, and policies. Security Identity Manager also contains workflow, audit logs, and reports. These are described in the following sections.

Here are the entities that are managed by Security Identity Manager:

- ▶ Users, accounts and attributes
- ▶ Passwords
- ▶ Group memberships
- ▶ Managed systems and applications

Security Identity Manager manages users on many managed systems. These include operating systems, such as many versions of UNIX and Windows servers, and applications, such as databases and business applications.

Security Identity Manager deploys an adapter to administer accounts on the system or application. Some adapters are deployed to the system or application and interact locally. Others can operate remotely and be deployed anywhere in the network.

On the server side, there is a set of definitions of each type of resource to be managed; this set of definitions is called a *profile* or *service profile*. Each profile must be deployed to the Security Identity Manager Server before you can manage this type of resource.

The adapter to server communication by default does not use SSL; however, you can enable SSL communication with one-way or two-way authentication.

On the Security Identity Manager Server, WebSphere Application Server SSL support is used. Each adapter instance is defined as a service within the Security Identity Manager server.

Accounts are associated with specific services. For example, there is a service for every Linux server. The services are defined within the organization tree and can have ACLs attached to control administrative access to functions performed against the service. A service can be defined only for a pre-existing managed target.

There is a service profile for every type of service. For example, there is one service profile for Linux services. The service profile defines the account attributes for that type of service.

6.9.2 Security Identity Manager key functions and logical architecture

Here is a summary of the Security Identity Manager key functions:

- ▶ User self-service: For users to maintain their account passwords and other personal information.
- ▶ Password management: Through Security Identity Manager, users and administrators can centrally manage and synchronize their passwords across all their accounts (that is, across systems and repositories where users are registered).
- ▶ Manage people and accounts: System administrators can manage an organization's employees (people) from a central location.
- ▶ Apply policy to people and account management: Policies are used to determine and enforce compliance of people and their accounts that are managed by Security Identity Manager. They also are used as the basis for automation of account provisioning and de-provisioning, account ID creation, and password strength checking.
- ▶ Apply workflow to people and account management: Workflows are a technical representation of specific business processes in Security Identity Manager and can be used to complement account provisioning and lifecycle management activities, such as adding, removing, and modifying people and accounts in Security Identity Manager and managed resources across the environment.

6.9.3 The Security Identity Manager logical architecture

Security Identity Manager is a Java application with a logical architecture, as shown Figure 6-20.

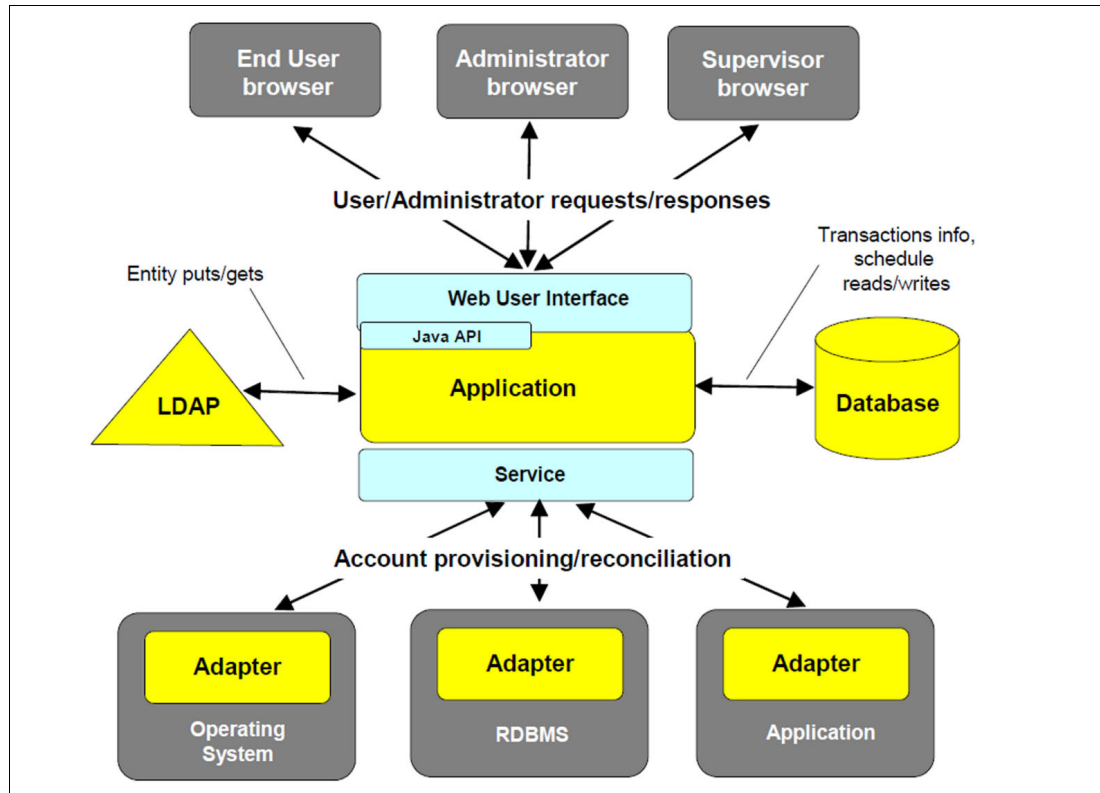


Figure 6-20 IBM Security Identity Manager architecture

It is composed of three main functional layers:

- ▶ The web user interface layer: The web user interface is the interconnecting layer between the user's browser and the Identity Management application layer.
- ▶ The application layer: This is the core of Security Identity Manager and runs on an application server. The application subsystem contains all the modules that provide provisioning-specific capabilities, such as Identity Management, account management, and policy management.
- ▶ The service layer: This core services subsystem contains all modules that provide general services that can be used within the context of provisioning, such as authentication, authorization, workflow, and policy enforcement. This also includes communication with the adapters on the managed services and to directories for storage of information.

The Security Identity Manager system uses an LDAPv3 directory server as its primary repository for storing the current state of the enterprise that it is managing. This state information includes identities, accounts, roles, organization chart, policies, and workflow designs.

A relational database is used to store all transactional, reporting, and schedule information.

Typically, this information is temporary for the currently running transactions, but there also is historical information that is stored indefinitely to provide an audit trail of all transactions that the system has run.

6.9.4 The Security Identity Manager RACF adapter

Security Identity Manager is connected by TCP/IP to the z/OS instances to which it provides Identity Management services. The Security Identity Manager commands sent to the adapter and responses to these commands are exchanged using Directory Access Markup Language (DAML) messages between Security Identity Manager and the RACF adapter that is installed in z/OS, as shown in Figure 6-21.

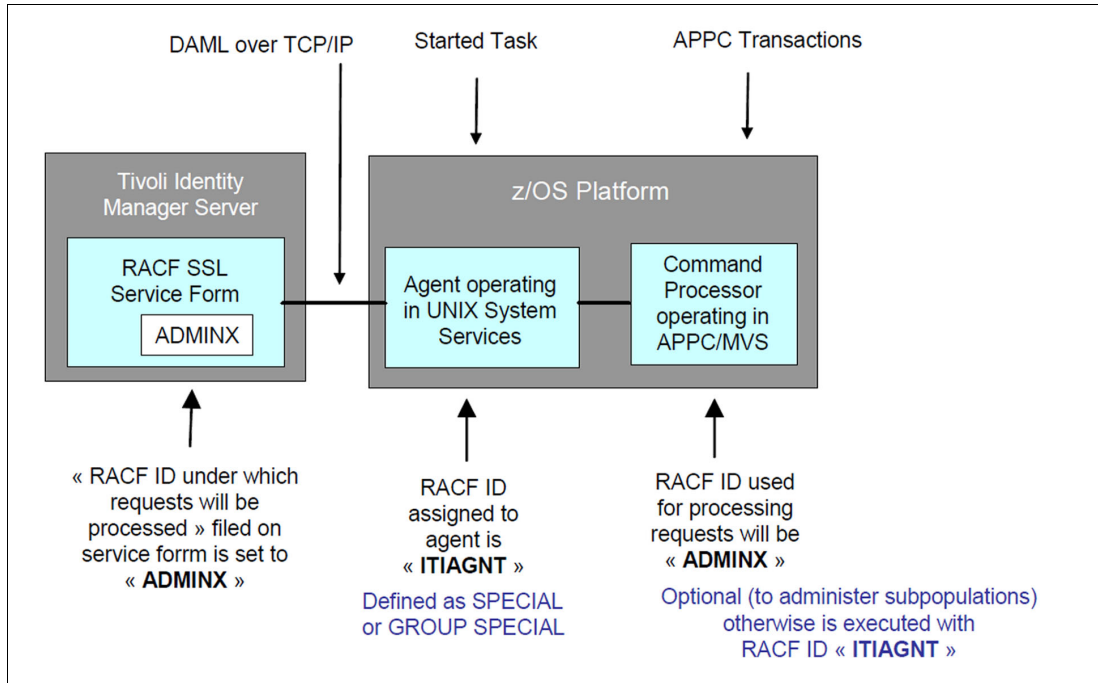


Figure 6-21 IBM Security Identity Manager RACF adapter

The RACF adapter is composed of two parts that are installed in the z/OS system:

- ▶ A z/OS UNIX-based TCP/IP agent to directly interface with Security Identity Manager
 - The Security Identity Manager agent is a z/OS started task listening for requests coming from Security Identity Manager. When such a request arrives, the agent uses Advanced Program-to-Program Communication (APPC) to trigger a command processor program in z/OS. The TCP/IP communication between Security Identity Manager and the RACF agent is protected with SSL/TLS.
- ▶ The RACF command processor
 - This is a set of APPC programs that issues RACF commands, or starts RACF utilities, and send responses back to the Security Identity Manager agent.

Figure 6-21 also shows examples of a RACF user ID being used for the running of the agent or the command processor programs:

- ▶ ITIAGNT is the RACF user ID that is assigned to the agent started task.
 - ITIAGNT is defined with the SPECIAL attribute if Security Identity Manager manages all the users in the RACF database, or with a group-SPECIAL attribute if Security Identity Manager is in charge of only certain users in the RACF database.
- ▶ ADMINX is an existing RACF user ID that can be optionally specified at Security Identity Manager.
 - In this case, the RACF commands are run under this user ID.

The purpose of this facility is to support managing subpopulations of Security Identity Manager administered RACF users, using specific administrators' user IDs. If this facility is used, then ITIAGENT does not need any specific attribute in itself. However, it should be defined as a surrogate of ADMINX. If this facility is not used, then the RACF command is run under the ITIAGENT user ID. The Security Identity Manager RACF adapter does not allow reflecting of local changes to a RACF password back to Security Identity Manager. Assuming that an installation requires that all passwords in the installation be resynchronized to the new RACF password, the infrastructure that is shown in Figure 6-22 provides this function.

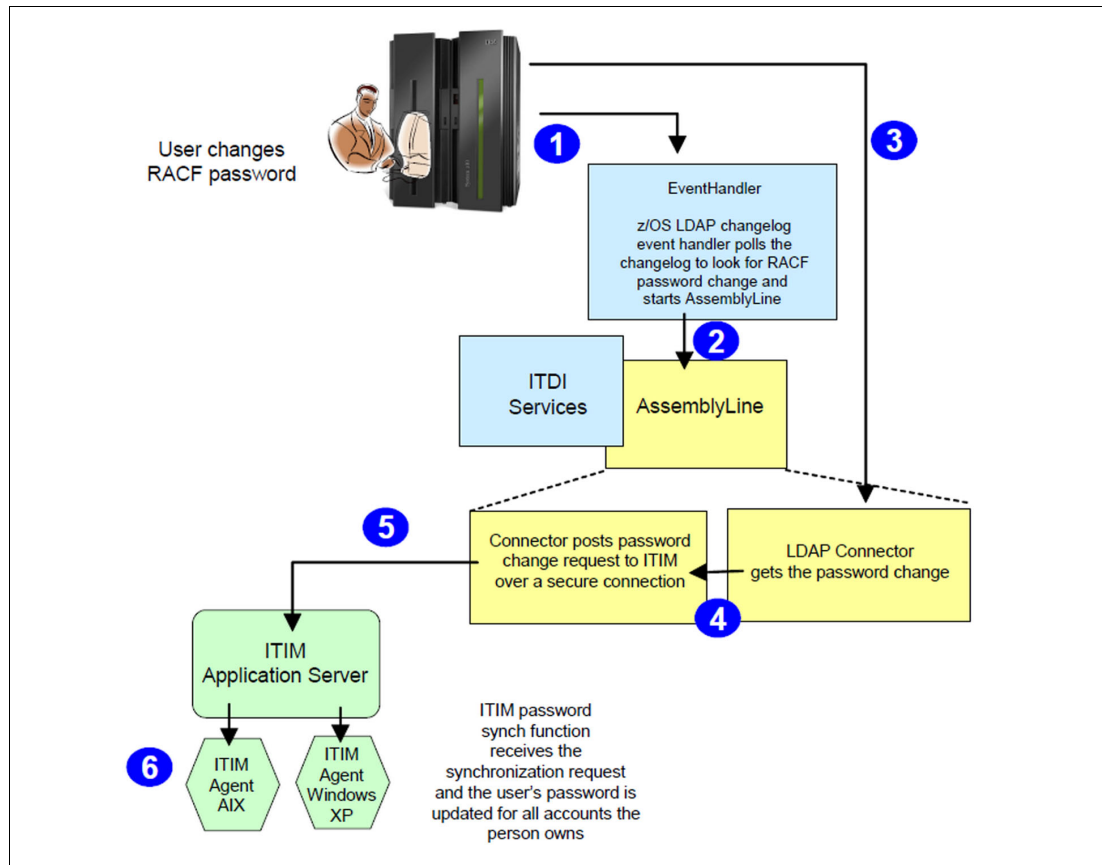


Figure 6-22 Security Identity Manager RACF password synchronization flow

Here is the event flow that is described in Figure 6-22:

- ▶ The user changes his password in RACF. This password change is recorded in the z/OS LDAP GDBM changelog and a password envelope is created. The Directory Integrator EventHandler for z/OS LDAP detects the password change (1).
- ▶ The EventHandler calls a Directory Integrator (ITDI) AssemblyLine (2) that is composed of two connectors:
 - The first connector securely retrieves the encrypted password attribute (3) and decrypts it with the supplied API in Directory Integrator. The connector then passes the decrypted password to the second connector (4).
 - The second connector posts the XML structured request to IBM Security Identity Manager to request the password change. The connector performs an HTTP post to the IBM Security Identity Manager password synchronization servlet to make the password change request (5).

- ▶ Security Identity Manager receives the password synchronization request for its use and checks to see what other accounts are eligible for password synchronization (6). Password change requests are then automatically initiated for the eligible systems, for example, AIX, Windows NT, and Active Directory.

6.10 Federated Identity Management

Federated identity technology is used to create a globally interoperable online business identity and drive relationships or affinity-driven business models between companies. The concept is nothing new, as there are real-world models for federated identities of individuals. For example, a passport is a global identity credential that vouches for one's identity in a country, an ATM card is a credential that vouches for one's bank account, and a driver's license vouches for one's ability to operate a motor vehicle and is also frequently used as a proof of identity in many business transactions.

Federated Identity Management is based on the business agreements, technical agreements, and policy agreements that allow companies to interoperate based on shared Identity Management. This helps companies to lower their overall Identity Management costs and provide an improved user experience. It uses the concept of a portable identity to simplify the administration of users and to manage security and trust in a federated business relationship. The simplification of the administration and the lifecycle management in a federation leads to the following value proposition:

- ▶ Identity Management costs can be lowered because companies are no longer in the business of managing users or identities that are not under their control, including the delegate administrator identities that are currently managed by many first-generation federation attempts. Businesses need to manage access to data but do not have to manage accounts and user account data.
- ▶ A user experience can be improved because users can navigate easily between websites while maintaining a global login identity.
- ▶ Inter-enterprise application integration within federations benefit from the end-to-end security and trust capabilities.

Integration can be simplified because there is a common way to network identities between companies or between applications. Organizations can implement business strategies that drive organic market and customer growth by eliminating the friction that is caused by incompatible identity and security management between companies.

6.11 Conclusion

System z sets up an excellent platform to establish a secured application hosting environment. Although System z is an enterprise platform, an organization still hosts various other platforms in its technology architecture and hence demands an overall solution to secure its environment.

Some of the security solutions that are described here caters specifically to System z or the z/OS. But, the intention is to provide an overview of all such solutions that present an enterprise view of managing security across all operating platforms. Thus, these products and solutions ensure to strengthen the security capabilities of System z as aligned with the IBM Security Framework.



Part 2

Guiding principles for IBM System z security

This part provides a more in-depth look at security preferred practices on the IBM mainframe.



Organizing for security

This chapter describes how organizations can ensure that security is maintained across their systems infrastructure and applications. This chapter covers the following topics:

- ▶ Describes how security must be maintained for the entire enterprise, not just the mainframe
- ▶ Describes the following factors and requirements:
 - The potential gap between security policies and the security implementation
 - The need for a security design
 - The need for a continuous improvement plan for security
 - The nature of IBM System z audits
 - The need for monitoring and alerting
 - Access recertification to avoid access creep
 - The purpose and nature of security controls
- ▶ Introduces the concept of a Security Engineering group
- ▶ Emphasizes the need for a *standards-plus* approach
- ▶ Describes conflicts that might arise in the management of security and the gaps in it

7.1 The mainframe infrastructure does not exist alone

Chapter 1, “Introduction: Why these books are being written” on page 3 established that the security of the IBM System z mainframe cannot be done in isolation from the rest of the IT infrastructure that is used by an organization. You can make the System z based applications more secure than other platforms, but you always depend on functions that are provided by those other systems.

7.1.1 Defense in depth: The castle approach

Regarding security, think of castles in the Middle Ages. Each castle had layers of pass-through to reach a *keep*, where the most valuable possessions or people were kept. This section describes how those layers are synonymous with the present day IT infrastructure.

Moat

Castles were frequently surrounded by a *moat*, which is a ring of water that ensured that there was a control over who could pass in to the castle. A *drawbridge* crossed the moat, but this was put in place only for trusted people to pass. If the drawbridge was raised, then the moat had to be crossed by other means, such as using a boat or swimming. Each of these methods of crossing the moat is relatively cumbersome because it slows down any potential attack. In some cases, crocodiles and poisonous snakes were in the moat so that no one could cross them by swimming.

The width of the moat also reduced the effectiveness of a bow and arrow attack on the castle. Long bows were less accurate, and cross-bows were less effective because of the distance.

A moat might be analogous to the modern day firewall.

Walls

Inside the moat were the castle's *walls*. The top of the walls could be patrolled so that potential invaders could be spotted as they advanced towards the castle. The walls were thick, and unlikely to be easily broken down. They had *windows* in them, which were small *slits* to allow arrows to be fired from within the castle. But external arrow attacks towards the castle would fail as the slits were small and the archers needed to be accurate to get an arrow to pass through that slit.

The walls also might have had built-in defensive mechanisms, such as the ability to direct hot oil or water onto anyone trying to scale the walls.

The walls might be analogous to application gateways and intrusion detection systems.

Keep

Within the castle itself, the *keep* might have been built on a hill or mound to make it more difficult for the invaders to attack. The keep was further defended by thick walls and the best soldiers. There were provisions within the keep to allow for long sieges.

Although the keep held so many provisions, there was still a regular need to gain access to it. However, there were gatekeepers to restrict access.

The keep might be analogous to the modern System z server.

7.1.2 The analogy

The castle approach shows the defense in-depth approach that is used in most organizations today. You do not have moats, but you do have firewalls with rules that allow some traffic, as with drawbridges. You have security zones, which are separate, with a controlled flow of data between them.

In many cases, we have a System z environment at the heart of the system, and this contains and processes our valuable data.

So, although the goal with this book is to ensure the security of the organization's data and systems, you ensure that security by using all that other infrastructure outside the mainframe, and the security systems within the mainframe.

7.2 Security policy definition and implementation

Most organizations have a security policy that typically states the rules for controlling access to data. There also are statements for data ownership, and there are rules about granting the least access that is necessary for each role.

However, there might be few instructions about the practical scenario of implementation. There might be little mention of any of the IT platforms that are involved. Thus, there might be little or no link between that policy and the security procedures that must exist.

Organizations find a great benefit in having documentation that relates the policy to the platform, and to each software product that needs security-related configuration. It should be possible to see the line from policy to procedures, and see that the policy is enforced in the implementation environment.

7.3 Security design

In the past, the System z infrastructure was secured by using the controls available with it. As new controls became available, only some of them were implemented. A few of them were deemed too restrictive and ignored. However, over the years, it has been common to *not revisit* the security controls; after all, they have been working all correctly, correct?

It is difficult to show that controls are working unless you run regular reports and relate them back to the security policy. Such reports should show what access is available and what unauthorized requests were rejected.

A frequently ignored question is “When was the last time the System z security administration changed procedures or implemented new controls as a result of a formal Security Policy change?”

Too often, security design is either ignored, avoided, or not done. It is frequently seen as the role of the z/OS security administrators, rather than the role of application designers.

Here is an example of an extreme situation, but it is not as uncommon as you might think:

This example involves a new CICS system, which uses a third-party product. Development worked on the application, but turned off most security controls because they got in the way of what development was trying to do. Development did not have a document defining what security controls were needed. When they were satisfied with the functional aspects of the development, they decided the application was ready to implement. So they passed the application to the production system.

When production support tried to run the system in production, they could not run it because RACF presented messages denying access to various resources. So, the RACF administrators granted access to the resources, and then production support tried again. They got further, but then got more denial messages in the log. They resolved those messages as well. They went on and granted access at every security restriction until eventually the product ran, and produced the correct results. At this point, production marked the security design as complete.

In this example, there was no attempt at any point to perform an application security design. No attempt was made to define what infrastructure security changes might be needed.

A security design for this application should have stated what resources needed to be accessed, what levels the access needed to be at, and what the auditing requirements for those resources are. It also should state which users need access, and whether there were any conflicts of interest if users of this system had access to other systems. It should be a document that addresses those security issues, but also can make reference to technical security matters that were encountered during development. These examples are just examples, and are *not* intended in any way to limit a security design.

7.3.1 Threats change over time

Over the years, IBM has introduced new security capabilities and controls at regular intervals. However, many organizations have decided to not implement those controls based on the assessment of the security threat level *then*. They regarded the controls as too restrictive.

As the whole threat landscape has changed since some of those *restrictive* controls were introduced by IBM, it is a preferred practice to revisit these controls and see whether there is a justifiable reason to enable those controls.

Secure infrastructure does not just happen. Secure applications do not just happen. Each one must be *designed*, and then *implemented* to that *design*.

Preferred practice: Revisit security controls regularly. If a control is available and a platform policy decision has been taken not to implement it, then this should be documented with the current justification, and revisited at regular intervals. These *revisitations* should examine the threats, and should be attended by technical staff who understand the nature of the control and the implications of enabling or disabling it.

7.4 Continuous improvement

In any quality process, there is a need for continuous improvement. This applies to security management, which is a process as well.

Too often, security management does not look for better ways to do things; it does not look for *more secure* ways to do things. If security is to improve, it must change. Have you ever heard someone say, “I do not want anything to change, I just want it to get better?”

Henry Ford said “If you always do what you've always done, you'll always get what you've always got.” Henry Ford knew that change was necessary for improvement.

Preferred practice: Institute a continuous improvement process for security management and security procedures. Make *change* an accepted part of the environment.

7.5 System z audits

Audits that are performed on many System z environments are done with a *light touch*. Most management teams are happy about this approach. They have a natural desire to pass audits. A passed audit tells them they are *apparently* doing well. However, this approach engenders a false sense of security.

Preferred practice: If your organization has a System z audit that comes back with no exceptions and you know this is not correct, act to ensure that it is fixed.

It is true that many auditors trust the System z security administrators that it is a highly secured platform, and that the auditors do need not worry about it.

7.6 Monitoring, alerting, and forensics

Some System z installations often do not pay attention to alerting and monitoring capabilities within their environments. This is a consequence of the trust that is placed in the platform and its capabilities. However, all too often this trust is misplaced when the processes and procedures around the System z mainframe are not as strong as they must be.

The consequence of this lack of monitoring controls is that any security breach on System z is either unnoticed or noticed too late.

After the alert is raised, you then must consider the forensic investigation of the event. The z/OS has a high performance and extensive logging system that is known as the System Management Facility (SMF). However, some logging is often disabled to reduce costs and maintenance. So, after a breach is discovered, the data that might have given details of what happened and when it happened is already lost or unavailable.

Preferred practice: Make sure that you have the capability to detect changes to your operating systems, and security settings that are critical for your business. Enable logging with proper retention. Use security intelligence products and capabilities so that you are alerted when any potential breach occurs.

7.7 Access creep

Do you have a recertification process to ensure that any access that is granted is based on a continuing need? This is a basic security process that is frequently missing from many System z environments.

Do you have a mechanism to determine whether the access your functional groups have is the minimum that is needed? Often, the decision whether infrastructure support groups need access is based on what they say. Do you have any way to determine whether they use the access they have?

The need for minimum access to do a job might be in your organization's security policy, but how can you be sure that you have implemented that minimum access? Minimum access is there to minimize the exposure if your logical identification becomes compromised in any way.

Preferred practice: Have a process for access recertification and continued business need verification. Institute a program for access reduction.

7.8 The purpose of security

Now, it is relevant to determine what is the purpose of the security. This question might seem to be a question with an obvious answer, but there are in fact several subtly different aims, especially for the mainframe environment.

7.8.1 Maintaining the integrity of the operating system

The prime aim of any security system is to maintain the integrity of the operating system. As described in 3.6.3, "Statement of integrity" on page 79, the z/OS and z/VM operating systems have statements of integrity that refer to the use of RACF as the External Security Manager (ESM).

So, the "security system" in this case is RACF, which has an important role in securing the controls of the operating system. In return, the operating system maintains the integrity of those RACF controls.

The operating system consists of many components. Some of these are software components (load libraries and other executable code), and other components are runtime components. Examples of runtime components are data sets, such as configuration data sets (SYS1.PARMLIB and extensions), page data sets, spool data sets, and logging data sets.

If the integrity of the operating system can be compromised, then it opens the gateway to compromising the security controls. Hence, the operating systems controls are the most important set of controls.

Subsystems

In System z, major software products or components are referred to as *subsystems*.¹ Examples of these subsystems are IMS, DB2, CICS, and others.

¹ In the heart of z/OS is a capability that is known as the subsystem interface. The use of this facility requires a formal definition of a subsystem. In this chapter, this term refers to major software or middleware components, which use APF-authorized z/OS facilities.

These subsystems use operating system level authorities, so they must be secured in the same manner as the operating system. A different group of personnel than the group that manages the operating system itself might manage those subsystems. Nevertheless, these subsystems need the same important controls applied to them, if you are to maintain the overall integrity of the operating system.

7.8.2 Maintaining the security of application data

The operating system hosts applications that contain data. The next important role of security is to maintain the integrity of those applications and their data.

Every organization has rules to determine who is allowed access to the data. The role of the security system is to implement those rules.

7.8.3 Maintaining the separation of applications, virtual machines, and networks

One of the most significant advantages of a System z over other platforms is its ability to run different operating systems on the same server with multiple applications running concurrently on each operating system.

Many standards for IT security require that there is a level of separation of one application from another one. This separation is achieved by requiring each function to be implemented on its own server.² System z, however, allows multiple applications to be running simultaneously on the same physical server.

z/OS maintains separation at run time by running each application in a distinct address space. z/VM maintains separation at run time by running each virtual server in a separate virtual machine. The concepts of address space and virtual machines are described in 3.3, “Virtualization” on page 47.

However, each application uses data sets and possibly other physical and logical resources that must be aligned to the runtime environment. It is the role of the security system to maintain this separation. This is *horizontal separation*.

² That server might be a virtual server running under a virtualization manager.

Figure 7-1 shows an operating system running four applications. Each application is shown in a different color, and the operating system is shown in black. The role of the security controls is to maintain separation between the applications and the operating system, but also between the various applications.

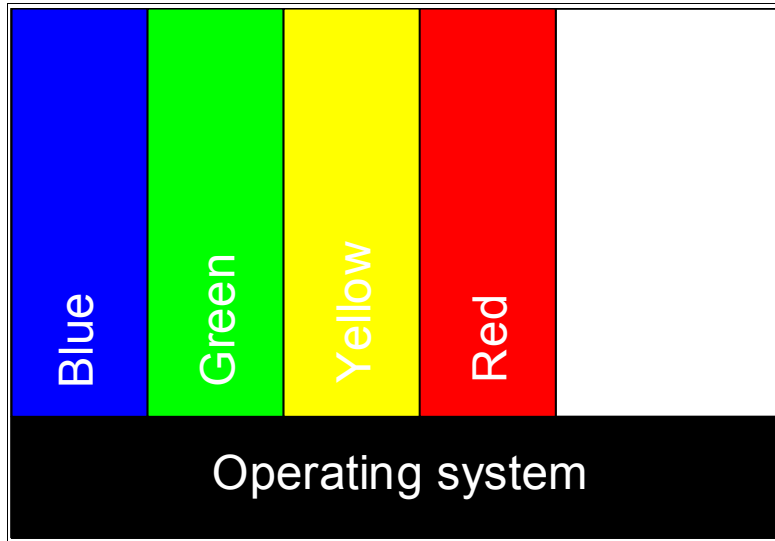


Figure 7-1 Security separation

More granular approaches are possible, and are frequently used. Each application might have different roles within it, which might require a Role-Based Access Control (RBAC) system. With this type of access, every user is associated with a business role, and the access that is granted depends on the role rather than the individual.

In addition, there will be interfaces between systems and so a need to share data feeds between them.

7.9 Types of security controls

Security controls differ in type. Some controls are designed to prevent actions by individuals on objects, and other controls might monitor actions that occur and record them. They might produce an alert.

7.9.1 Preventive controls

Preventive controls prevent unauthorized actions, where an ESM such as RACF denies access to a resource by a specific user.

Preventive controls are preferred over other controls because they can stop any undefined or unauthorized access to a resource.

7.9.2 Monitoring controls

Monitoring controls keep a record of access that has been granted. ESMs such as RACF have settings that determine what levels of access are monitored. The monitoring is performed by SMF, which is a secure mechanism within the operating system.

Sometimes the actions that take place are not related to a single specific event, for example, a “many” normally authorized events take place within a short period. For example, a user might be accessing a database in an unusual pattern. This pattern might indicate an attempt to copy data from a specific category of records. Consider the following example:

Mary Smith at XYZ bank normally accesses customer records to adjust credit card limits, and she is authorized for it. However, on a specific day, a monitor might detect that she has adjusted about 10 times the number of records than is normal for a normal day. This might be an indication that actions are being taken that are contrary to the security policy.

In this case, each individual action was part of Mary’s job and would be authorized. However, the sheer volume of changes that are made during a period causes a monitoring alert to be raised.

7.9.3 Detective controls

With a detective control, whenever an unauthorized change is made by a user or a process, an alert is raised within a short span of time. Such a near-real-time alert can then be used to revert the change, or to take appropriate actions to apprehend the person making the change.

This type of control can be implemented on mainframe systems by using IBM Security zSecure Alert. Alerts can be raised for various different actions, such as changing a configuration parameter or running a RACF command. Those rules for alerts can also filter them by time of day, so that, for example, only changes made outside of a normal time window raise such alerts.

7.9.4 Forensic controls

Forensic controls refer to capabilities for analyzing security data after an event to determine the root cause. For example, if a change was made within a certain period, a forensic control might be able to determine when and by whom the change was made.

7.9.5 All controls

A mixture of all of these types of control is always advisable. A preventive control might fail, and in such cases, it is useful to know that a change was made even though it could not be prevented. If detective controls also have failed, it still might be possible to determine when an event occurred and who caused it to occur.

However, if a choice must be made, a preventive control always takes precedence over a detective or monitoring control.

7.10 Standards-based approach

There are many security control standards for the various System z platform components. Some companies maintain their own standards, and others use external standards. For example, one such external standard is known as the Security Technical Implementation Guides (STIGs), which are produced by the US Department of Defense (DoD).

For example, there is both a z/OS STIG and a z/VM STIG, each of which can be obtained from the US DoD at the following website:

<http://iase.disa.mil/stigs/os/index.html>

However, STIGs are only one example of security standards for System z mainframes. This book does not endorse any particular standard. Industry preferred practices, technology advancements, threat levels, and business needs are some of the common factors that drive the security policy for any organization. Most of the standards are built on the aggregation of all these factors for the best possible protection.

7.10.1 Need for a standards-based approach

Having a standard to measure your own security controls is a good idea. It enables a measurement process that shows that security is improving, or possibly getting worse. Without such a standard to measure against, it is not always easy to determine whether improvements are happening.

Using a standard from an external body as a measure for security has several advantages:

- ▶ It is not necessary to develop the standard yourself.
- ▶ It is usually a standard that has been subject to several reviews by others.
- ▶ It is likely to be a standard that will be maintained in the future, and so will cater to future technology and developments.
- ▶ If problems are found in the standard, they are likely to be fixed.
- ▶ There is a greater likelihood that any problems with the standard will be discovered.

7.10.2 Challenges with a standards-based approach

However, there are problems with a standards-based approach to security controls that must not be overlooked:

- ▶ There is a natural tendency to believe that if your System z environment is compliant with a standard that then you are secure, which is not the case.
- ▶ If there are gaps in the standard, and the standard is published, then people who want to attack your system have access to the documentation and can seek out those gaps.
- ▶ Standards are reviewed periodically and updates are applied. However, at any one time, there are likely to be several updates pending, which leaves gaps in the standard during the review period.

7.10.3 “Standards-plus” approach

Choose a standard for *your* System z security environment. After you choose a standard, the system should be carefully measured against the standard and a gap analysis should be performed.

However, your goal should never be to simply meet the standard. Your goal is to meet your own security requirements with the restrictions that are set by your own business needs.

Assume for the present that the agreed business needs require a higher standard of security than what is defined by the standard you measure against. You can now explore the “standards-plus” concept, which is based on the standard that you choose, but your set of controls are augmented by extra security controls to reduce the risk to the level that your business can stand.

Your aim here is for your system to be secure. Being *compliant* is good, but being *secure* is better.

7.11 Security engineering

Although many organizations have a security policy and processes derived from them, only a few have a separate technical department with a role that is dedicated to managing IT security across their business. However, there are considerable advantages to such an approach.

In many organizations, there are multiple computing platforms in use. Each platform has its own processes for managing security. These processes and procedures differ from one platform to another.

You might have applications span across those different platforms. Each group is responsible for their own security, but no one is responsible for the gaps in security processes between two platforms, and it is no one's specific responsibility to ensure that such gaps are fixed.

A dedicated security engineering department with cross-organization responsibility and authority is a good way to address such issues.

Figure 7-2 show the relationships that a security engineering group might have. It must be linked to all of the following groups:

- ▶ Business and application development
- ▶ Audit and external compliance
- ▶ Security operations and management
- ▶ Technology infrastructure and capabilities

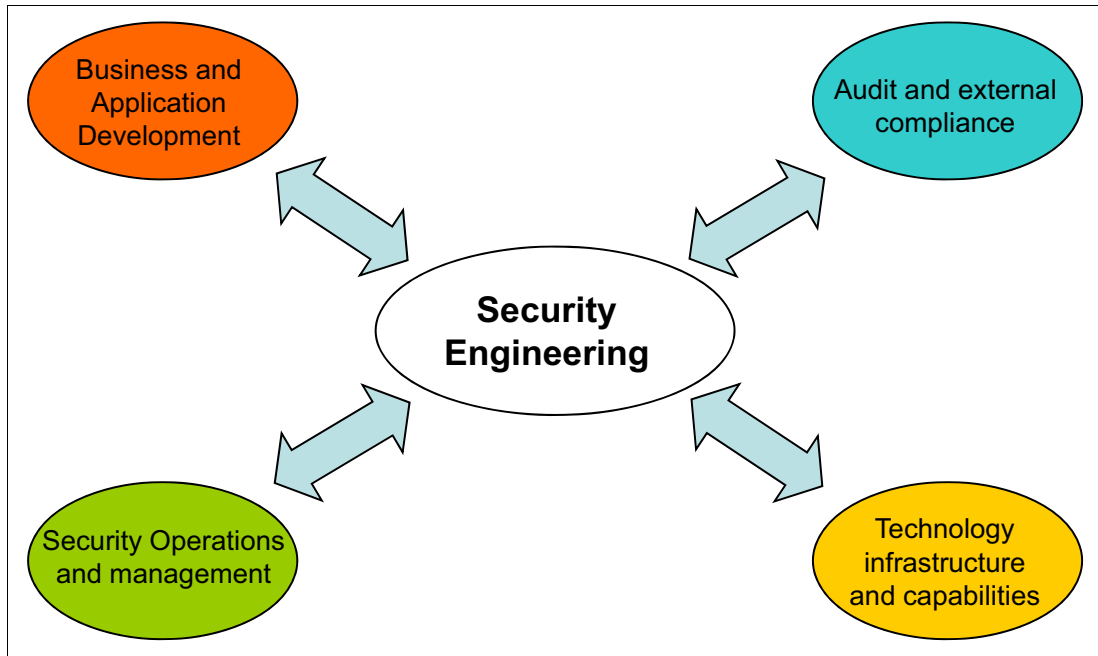


Figure 7-2 Security engineering connections

7.11.1 Business and application development

Security engineering is aware of the security needs of the business and works to ensure that those needs are met by the IT infrastructure and applications. In addition, the security engineering department emphasizes those needs in application development. The aim here is that application development addresses those security needs of the business during the development cycle. Too frequently, application development ignores security needs and assumes that these needs are the responsibility of a security management group. Such an approach frequently leads to badly configured security.

In application design terminology, the security requirements frequently are referred to as *non-functional requirements*. In today's modern systems, the security behavior is a true function of the application. If the security requirements are not met, then the system should *not be implemented*.

7.11.2 Audit and external compliance

The security engineering department should be aware of the compliance needs of the applications and infrastructure. Such an awareness, alongside the security needs of the business, tends to ensure that money that is spent on compliance results in improved security and compliance.

Compliance to external standards also must ensure that there are some end-to-end security controls on applications. Security engineering takes an overall view of the application compliance.

7.11.3 Security management and operations

Security management has a view of the patterns of requests for security changes to IT resources. They see problems as they appear and if there is a security intelligence solution in place, they see trends in security issues.

Security engineering should have strong links to this organization because they are expected to respond to business security needs in a timely manner.

7.11.4 Technology and infrastructure capabilities

The security engineer must understand all of the security technologies being used in the organization. There is a need to understand the security characteristics of the following items:

- ▶ Servers
- ▶ Operating systems
- ▶ Middleware
- ▶ Applications
- ▶ Network devices
- ▶ Encryption devices
- ▶ Certificate management
- ▶ Key management
- ▶ Security processes and procedures

The department has a view on the future of security, and sees trends in security-related activities in other parts of the IT infrastructure over the world. They are in touch with external bodies regarding new security standards and technologies.

7.11.5 Other responsibilities

In addition to those four core departments, the security engineering department looks outside the organization to see what the threat landscape is like. On this basis, the security engineering department might recommend that extra security measures be employed.

7.11.6 Security engineering and data ownership

In many organizations, the concept of data ownership is formally recognized, which is a good thing because decisions over who gets access to data can be made by the department who owns the data and is responsible for it.

However, you often see situations where the owners of the data grant access far too easily, with lack of understanding of what the intended use is for the data, or the risks that are associated with granting access.

Although security engineers are not the correct authority for making such decisions, they can have a role in the process in the following ways:

- ▶ They can understand better than the data owners the risks that are associated with having access to the data in question.
- ▶ They can understand what joint risks are associated with having access to this data and some other data.
- ▶ They can take an overall view of data access rulings, and make recommendations for remediation to reduce business risk.

This is not just a matter of data classification levels. It is a matter of ensuring the necessary separation of duties in terms of data management.

7.12 Conflicts of interest

Conflicts of interest occur when there is a single control point, possibly a manager, who has one or more conflicting sets of objectives. When one of these objectives is to maintain and monitor security, it can often fall into conflict with others.

Within IT security, the concept of separation of duties is understood. In some organizations, separation of duties is used to ensure that the implementor of a change cannot be the same person as the requester.

Separation of duties also is often used within encryption key management departments, where the authority to a key and the ability to change a key are assigned to different people.

Does your organization have conflicts because of a lack of separation of duties? It is not uncommon in System z management, where a reduction in the number of personnel has placed multiple responsibilities on the same person. This is often true of security administrators, who end up with no one to audit their tasks.

Recommendation: Ensure that security administrators are not given responsibilities for auditing and security management. There is a strong reason for separation of duties in security policies.

However, there are other conflicts that arise within security departments.

7.12.1 Conflict of availability versus security

Security operations is a common name for the department that is responsible for making security changes. These changes include the management of the joiners, movers, and leavers process for staff (sometimes called as the JML process), implementing access level changes for resources, providing security implementation services for new systems, and much more.

This department often reports to an operations manager, who also has a responsibility to maintain service availability. Frequently, the manager sees his prime aim as service availability, and the security part of his role has been “tacked on” because of a staff reduction.

Conflicts can arise when there is pressure to ensure continued service availability, but this can be done only by compromising security. Because of the way many mainframe shops are run, this type of organization is often used. Such a situation can lead to risk-based decisions being made in the wrong place in the organization on behalf of the whole business, by managers who should not have the authority to take such risks. Although a carefully designed risk-based approach is used to design the security system by using the IBM Security Framework and the IBM Security Blueprint, it can be easily compromised by an ill-advised reporting structure of this nature. Thus, careful attention is required to ensure that these two conflicting responsibilities are brought together only at the level of the enterprise-wide Chief Information Officer (CIO).

It is assumed that the CIO has the authority to understand the risk-based policy set in place in the organization. However, others might want the risk-based decisions to involve the business lines. This approach is reasonable if this is what the business deems appropriate. However, what you want to avoid are security compromises being made by managers who ought not to be in a position to “bet the business”.

7.12.2 Conflict of cost versus security

A second type of conflict arises between cost controls and security controls.

This is a difficult subject to address and for which to provide concrete rules. Every organization has a different view about how to resolve such conflicts. So, all you can do here is explore some ways to think about the issues.

A risk-based approach usually is how such matters are handled. Assessing the risk that is associated with a given security gap must be made based on at least two factors:

- ▶ You must determine the likelihood of the security gap being used in some way.
- ▶ You must determine the effect or impact of that security gap being used by unknown persons.

The impact is affected by all sorts of issues, including the breadth of use of the systems, the nature of the data being processed, the quantity of data that might be exposed, and the nature of the potential exposure.

The security gap also can be mitigated by other mechanisms. For example, a security gap in a preventive security control may be mitigated by a detective control on the same security gap.

Ultimately, if closing the gap produces a large financial cost, a cost of availability for a short period, or a cost in some other way, such as a loss of capacity or a loss of time, then some kind of balance must be struck.

There are numerous documents on risk assessment in IT. The National Institute of Standards and Technology (NIST) has published a number of useful guides on the subject. One of those can be found at the following website:

http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf

There is also an older but useful guide on carrying out risk assessments that can be found at the following website:

<http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>

Authority to spend

Ultimately, the cost that is associated with the risk must be taken by the organization, and must be taken at that level of management who can authorize the cost that is associated. Here is an example of such a situation:

- ▶ In the XYZ supermarket chain, a security gap is identified. A software solution is available to correct this gap. However, the solution costs \$70,000 to implement.
- ▶ The situation is shown to the Security Operations Manager. He has the authority to sign off spend at \$5,000. On examining the report of the security gap, he makes the decision that the risk is too small to worry about.
- ▶ However, he does not have authority to sign off for \$70,000 of spend. So this decision should ideally be referred to an appropriate senior-level manager who does have the authority for that level of spending.

The danger in such a situation is the manager is taking a high risk for his own authority. Just because the risk does not materialize into a problem does not make his decision correct or justified.

7.13 Proving that security controls work

Many organizations put significant effort in to demonstrating that their business resiliency and disaster recovery plans are effective. Those organizations usually do not test their security measures. The rapid rate of change in many organizations means that security risks and exposures can arise in unexpected places overnight. Hence, it is necessary for each organization to demonstrate that there are effective security controls in place.

7.13.1 Testing individual controls

Each security control needs some testing process before implementation. The tests that are used and the results of those tests should be retained so that they can be repeated in situations where a relevant change has taken place.

Like all tests, the materials should include the expected results.

7.13.2 Regular security control scans

Regular scanning for exposures or risks should take place. This task can be automated with tools, such as that provided by Security zSecure Audit.

7.13.3 Penetration testing

Penetration testing can be of value, but is not always the best method of providing assurance of security.

z/OS systems are frequently tested for penetration from within the enterprise. This type of testing is the last type of testing to be performed after you are confident that all exposures have been addressed.

PCI-DSS and possibly other standards require regular penetration testing, but do not always state where from where the test is to start.

7.14 IBM services to help you

There are several IBM services that can be used in improving the security of your system.

7.14.1 Preventive maintenance

IBM regularly finds and fixes security vulnerabilities. Make sure that you are up to date on preventive maintenance, which includes all the latest security APAR fixes.

Keeping the *old* around: Organizations sometimes preserve old versions of IBM product libraries for various reasons. If you do this, and those libraries are no longer at a supported level where preventive maintenance is provided, consider making those libraries non-executable. It does not make sense to keep your main library up to date on security patches if an attacker simply needs to know the name or location of the alternative library where security patches are no longer being applied.

7.14.2 The IBM System z Security Portal

The System z Security Portal is the only place where SMP/E HOLDDATA for security issues is available. The portal also contains CVSS scoring for z/VM security and integrity APARs. Each IBM customer can sign up to receive this vulnerability information. However, the details of how the vulnerability can be exploited are *not* given. For more information, see the following website:

http://www.ibm.com/systems/z/advantages/security/integrity_sub.html

7.14.3 IBM Emergency Response Services

If you have a breach in security that results in data loss or other major issues, IBM has a service that can assist you in the management and resolution of the breach. For more information, see the following website:

<http://www.ibm.com/services/us/en/it-services/emergency-response-services.html>

7.15 Decision time

Before moving forward to assess or remediate any IT environment, whether it be System z or any other platform, choose an appropriate standard to use as a base. This standard should be chosen as the most applicable to your corresponding industry segment.

The necessary additions to that standard are chosen regarding the following items:

- ▶ The threats that are currently pertinent to the industry in question
- ▶ The “risk posture” of the organization

If there is a requirement for risk reduction and a conflict arises in terms of cost, then the gap should be documented and mitigated in some other way. *However, the standard should remain.* A lack of funds is a valid reason to take a business risk. It is still not a valid reason to alter the standards of security that are required for the enterprise.

7.16 Conclusion

This chapter has described multiple items regarding security.

- ▶ System z within the enterprise

The security of IBM mainframe systems is not an issue that stands alone. The IBM System z mainframe must be secured within the IT infrastructure of the organization.

Organizations have multi-layered security, which is comparable to the rings of defense that are used in castles.

- ▶ Security design

Infrastructure and application security must be *designed*.

- ▶ Continuous improvement

Continuous improvement is important for security systems.

- ▶ System z audits

System z audits are not always as thorough as they should be.

- ▶ Monitoring, alerting and forensics

There is a need for monitoring and alerting, and to have some forensic capability.

- ▶ Access recertification

There is a need for access recertification and access reduction.

- ▶ Role of security

The security of System z fulfills multiple roles:

- To protect the integrity of the operating system
- To provide protection for applications and their data
- To provide separation for one application from another

- ▶ Types of security controls

There are various types of security controls:

- Preventive controls
- Monitoring controls
- Detective controls
- Forensic controls

Preventive controls are preferable whenever a choice must be made.

- ▶ Standards-based approach

A standards-based approach to security controls should be used. However, to ensure that systems are secure rather than merely compliant, use a “standards-plus” approach, and augment the standards with extra levels of security or stronger settings.

The publicly available security standards should be regarded as a non-acceptable minimum.

- ▶ Security engineering

A security engineering approach across the organization is a useful way of ensuring that the organization is secure within the constraints that are chosen. This is a reasonable and effective way to ensure that the business risks the organization is prepared to accept are the only ones to which they are exposed.

- ▶ Conflicts

Conflicts can arise in terms of security responsibility, where clashes can be found with availability management.

Too often, risks are taken at too low a level within an organization.



IBM System z hardware

System z boasts of a robust hardware technology that provides reliability, high availability, and a highly securable platform. With the capability of a complete data center in a box, the way System z is configured defines these characteristics in practice. Thus, this chapter covers the security aspects of the System z hardware components. This chapter covers the aspects of configuring System z hardware components and functions, and the processes and procedures that are required to keep those configurations intact.

This chapter considers the latest models of System z at the time of writing, such as the IBM zEnterprise EC12, and the technology it provides. With the technology advancing every day, expect to see new features, threats, and protection mechanisms.

This chapter describes the security preferred practices of the following items:

- ▶ Physical access controls
- ▶ Support Element (SE) and Hardware Management Console (HMC)
- ▶ Input/output configuration
- ▶ Terminals and printers
- ▶ Storage devices
- ▶ Tape and removable media
- ▶ Network adapters

You will see the considerations that help you use the features of System z to secure your computing environment, supplemented by robust processes and procedures.

8.1 Physical access controls

When you secure System z hardware, physical access protection is the first thing to be considered. It applies to all the hardware components of System z that are installed in a data center.

Because the impact of unauthorized access to System z is considerably higher in a large IT infrastructure, protecting it is deemed critical, much like the logical access controls that you put in place. Physical access is not limited to securing just the hardware, but also includes all remote devices or terminals that connect to System z and things such as the removable media that carries backup of mainframe data. Every connection to the mainframe must be secured because any single vulnerability can lead to disruption of services on a large scale.

Access controls for devices making connections to the mainframes from remote locations, such as mobile devices or remote terminals, are usually controlled by logical access controls. Still, there are few things to be considered about protecting such devices from the physical controls perspective, which are explained in the next few sections.

Why is physical access control more important for mainframes compared to other platforms? System z runs most of the business-critical applications and has a wider scope of coverage of applications in the organization. It serves as the back end for most IT processing, so a disruption of its service has a chain effect on all the other platforms that run functional parts of every application. To put it simply, the mainframe is a data center by itself, and there is a huge impact if its services are disrupted.

For example, Customer ABC has a huge data center with its System z hosted in it along with many other servers and devices. There is a planned electrical maintenance activity inside the data center, which is expected to be nondisruptive to any IT operations. It was reviewed at all levels with strict change control mechanisms, and resources were deployed with clear instructions.

Two electrical engineers are working on this maintenance activity. One is providing instructions remotely from the control panel through a walkie-talkie to the second engineer who is working close to an electrical junction in the false flooring where the System z is hosted. Misinterpreting an instruction that he received from the control engineer, the second engineer flipped the power switch of the System z, bringing it to an abrupt halt. This action causes a tremendous amount of damage.

When the second engineer asks the ABC data center manager about what happened, the data center manager replies, "You just got us ready to do a power-on-reset!" The second engineer still could not understand what the data center manager said.

What went wrong here? When electrical maintenance activity is in physical proximity to the System z and other IT servers and equipment, the steps in the activity should have clearly identified the exact electrical connections and markings that are involved. Every electrical point in the data center can be labeled so that they can be used in such identification, preventing an error as in this case.

Thus, physical access controls are important from a wider perspective than just the System z and its related components. Even an unrelated activity can be a potential threat to the security and availability of the System z or the applications that are hosted on it. A thorough review and planning of all activities on or from the physical space that is shared by System z or its components from all angles helps reduce the risk of such threats.

This example does not mean that nothing can be done to prevent such errors. It is still important to evaluate every possible situation that can disrupt the service while reviewing such changes being performed, even though it might not be directly related to IT services. Such a review might not prevent an error from occurring, but the review can greatly reduce the probability of such errors and the impact to service. Obviously, such errors become a learning experience and provides a mechanism to strengthen the processes and procedures.

8.2 Hardware Management Console and Support Element

The Hardware Management Console (HMC) and Support Element (SE) are the functions that provide management functions for the System z hardware and its microcode, including the configuration for Process Resource/System Manager (PR/SM), timers, or the Coupling Facility (CF). The HMC and SE capabilities are described in 4.2.3, “Controllers” on page 93.

Although the SE is attached within the central processor complex (CPC) frame in modern mainframes, the HMC has transformed itself into a remote function that is not restricted to the data center. With web enablement, an HMC application can be accessed remotely over the network if configured so.

The SEs are two integrated notebooks that are supplied with the CPC, for example, the zEC12. One is always the *active* primary SE and the other is the *backup* alternative SE. The SEs are closed systems, just like the HMCs, and no other applications can be installed on them.

With the HMC and SE, a system operator can perform logical partitioning functions, service functions, and various system management functions using either the *classic* HMC user interface or the new *tree-based* user interface. The HMC also provides a comprehensive set of APIs for advanced automation of data center systems.

Recently, the HMC and SE have started to provide more advanced capabilities for managing both the physical and virtual systems across multiple architectures and environments. They provide a unified approach to platform management, which is designed to lower IT operational costs and increase operator and IT staff productivity. The HMC allows viewing and managing multi-nodal servers with virtualization, I/O networks, service networks, power subsystems, cluster connectivity infrastructure, and storage subsystems through the Unified Resource Manager. A task, *Create Ensemble*, allows the Access Administrator to create an ensemble that contains CPCs, images, workloads, virtual networks, and storage pools, either with or without an optional zBX.

The HMC communicates with each CPC through the SE. When tasks are run at the HMC, the commands are sent to one or more SEs. The SEs then issue commands to the respective CPCs or the controlled zBX, if any. CPCs can be grouped at the HMC so that a single command can be passed along to all the CPCs that are defined to that HMC. One HMC can control up to 100 SEs, and one SE can be controlled by 32 HMCs.

8.2.1 Considerations for multiple Hardware Management Consoles

The SEs and HMCs provide functions that are disruptive to the virtual or physical resources and operations of the System z hardware. Often, multiple HMC instances are deployed to manage an overlapping collection of systems. Before the announcement of zEnterprise systems, all HMCs were *peer consoles* to the managed systems. Furthermore, all management actions are possible to any of the systems that are reachable while logged in to a session on any of the HMCs, subject to access control.

With the definition of an *ensemble*, this resource management paradigm changes. Management actions that target a node of an ensemble can be done only from the primary HMC for that ensemble. But, this is applicable in relatively large environments with more than one zEnterprise servers with zBX to be managed as a single logical virtual system. Still, in such an environment, any action on an ensemble impacts the whole group of resources that is defined to it, and so demands huge caution in defining responsibilities for those roles and resources that are authorized to perform those actions. A clear definition of access control rules is required with proper change control mechanisms to validate the actions that are performed on an ensemble through the HMC.

8.2.2 HMC guiding principles

In many small and medium environments, or even large data centers with System z predominantly for mainframe workloads, SE and HMC act as a critical function for regular operations. So, why must you keep them secure? Consider the following example:

Customer ABC deploys a System z for some of their applications, which are active only six days a week. So, the customer shuts down their System z at the end of every week as a housekeeping routine and starting it again at the beginning of the week. As part of their start procedure, they also have a step to start the Activity Monitor in the HMC and keep it running to display the real-time resource use. This is at the end of their start procedure, and is a preferred practice. All these steps are clearly documented with clear roles and responsibilities, and accesses granted to corresponding roles.

One day, the engineer who runs the start procedure was ready to start the activity monitor. A running monitor is indicated by an icon that says *Activity* in the HMC. Instead, the engineer overlooked it and double-clicked the Activate operation for the CPC. He even acknowledged it on the confirmation dialog that is presented for any such disruptive action. Within a second, the entire system was abruptly down and started activating the CPC again.

What went wrong here? The customer clearly had defined procedures, roles, responsibilities, and accesses defined. The HMC also was physically protected only for authorized personnel. So, this error easily can be argued as a human error. In any situation, with all the control mechanisms in place, there is always the involvement of humans to perform certain tasks. But, you should strive to prevent the possibility of such errors by keeping your processes and procedures strong and secure.

In this case, all disruptive actions should be clearly identified and authorized only for certain user IDs that are restricted from performing other normal operations. The process also can schedule different times or resources for critical tasks to eliminate the possibility of overlap or confusions during subsequent actions by the same resource. Such a mechanism of scheduling of events and assignment of different resources can reduce the probability of human errors.

There are some basic capabilities of the HMCs that help you configure it for high level of security:

- ▶ Any communication over the HMC is encrypted by SSL.
- ▶ HMCs have two isolated network adapters, one for the dedicated communication with the SE network and another for the external communication for remote access.
- ▶ HMCs have an internal firewall for filtering inbound traffic.
- ▶ HMCs are configured not to route any network traffic through them.
- ▶ HMC supports enabling or restricting access for each user ID at every function, task, or object level.

- ▶ HMC supports LDAP authentication for its user IDs.
- ▶ HMC supports Secure FTP using the SSH protocol.
- ▶ Group of HMCs within the same environment can be isolated by using a feature that is called Domain Security.

Here you can examine a collection of basic guidelines and preferred practices for securing your HMC and SE:

- ▶ As SEs are physically enclosed within the CPC frame, they are usually covered for physical security as defined for the System z server. Have audit-controlled physical access restrictions for the System z hardware and SE that is attached to it.
- ▶ Although most of the operational tasks for hardware management can be handled remotely from the HMC, there are few actions that can be performed only from the SE. Have such actions performed only by authorized specialists (IBM CEs) with clearly defined procedures and change control mechanisms. The specialists or IBM CEs usually recommend precautionary measures for such critical changes to recover or fall back in case of a problem. Preserve the access to user IDs for such tasks in a secured way within the organization and ensure that they are used only during an authorized business need.
- ▶ Maintain audited controls on your SE to HMC and HMC to SE associations in case of large environments where multiple SEs and HMCs are active.
- ▶ If you are using locally attached HMCs for regular operations, ensure that it is physically secured with the same standards as for the System z server itself.
- ▶ Disable the capabilities of your locally attached HMC to *boot* from any external device. Data copy or transfer from locally attached HMCs should be restricted only to IBM approved media and tools that are defined for HMC operations.
- ▶ Although the HMC to SE connectivity is SSL encrypted, always have the connection in an isolated network that is dedicated for this communication to provide better security.
- ▶ use of the HMC Remote Security Facility (RSF) is suggested in place of System z Call Home facility to eliminate exposing the System z server or the SE to the Internet. HMC RSF provides a secure, isolated, and controlled gateway to the Internet for reporting System z problems and First Failure Data Capture (FFDC) information to IBM RSF servers. With this, the communications between CPC to SE, SE to HMC, and HMC to Internet are isolated at a physical level with different etherNet adapters. RSF connections are encrypted by SSL.
- ▶ Although HMC has its internal firewall to track and control inbound connections, use of additional proxy servers is suggested to audit all outbound connections from HMC to the Internet, wherever it is applicable. This proxy can also limit HMCs access only to IBM RSF servers, thus cutting it off from the wider Internet.
- ▶ Disable or remove default user IDs from HMC that are used during initial installation. Change passwords for IDs such as Access Administrator (ACSADMIN), which cannot be deleted.
- ▶ Define roles, actions, and access levels before designing your access profiles or user IDs. It is seen as a preferred practice to share HMC user IDs between multiple resources, which must be strictly avoided.

SEs contain configuration information that runs the System z. This information is so critical that it could be targeted with malicious intent. For example, it holds the Input/Output Control Data Set (IOCDs) file, which is created from the IODF configuration from z/OS. This file can also be read from an HMC after converting it by using a disassemble task from an SE or HMC. The readable form displays the IO configuration of the entire CPC, thus exposing it to attackers. It also can be edited from a SE or HMC, thus disturbing the integrity of the configuration.

Although such tasks are protected from the z/OS perspective to allow only authorized users to perform them, it also demands the correct level of protection from SE or HMC. In this case, an SE can even restrict specific images to write to its IOCDs, which defines an I/O configuration that can strictly be isolated only to specific images in a multi-image environment. The only way to prevent such issues is to define appropriate roles and restrict access at both functional and task levels.

- ▶ Maintain the same user profile information on all production HMCs by synchronizing them through the data replication feature
- ▶ If your organization's user authentication server can provide LDAP-based authentication services, configure your HMC to use this LDAP server for authentication instead of its local authentication for the user IDs. User templates and patterns can be used with LDAP authentication for grouping user roles and profiles.
- ▶ Align the HMC user ID provisioning and password rules with the organization's security policies and standards.
- ▶ Enable remote access only for the specific user IDs that require it. Restrict remote capabilities of those user IDs strictly to operational tasks. Disruptive or global task, such as a restart or LIC upgrade, should be restricted to limited IDs or completely turned off from remote access. Performing such tasks from locally attached HMCs enforces greater scrutiny and control. Also, limit users to read-only capability for operating system messages wherever appropriate to restrict them from issuing commands that disrupt the entire operating system.
- ▶ Use certificates that are signed by a certificate authority (CA) for remote users of the HMC, instead of HMC self-signed certificates.
- ▶ Enable logging for all user accesses and operations that are performed on HMC. Ensure that timely backup and retention of those logs are defined for debugging and investigations.
- ▶ API support should be activated only when it is critically necessary. If enabled, it should be restricted to specific user IDs or IP addresses. The automation script or operation must be clearly validated in a test environment before enabling them on production HMCs.
- ▶ Enable Security Event Notification and Audit reporting and establish a process to review them at regular intervals. Safely retain the HMC logs and audit records according to the organization's security and retention standards.
- ▶ Use Secure FTP for any necessary data transfer to and from HMC, such as downloading logs or reports.

8.3 Input and output configuration

All System z servers require a description of their I/O configuration. The I/O configuration is the set of all I/O devices, control units, and channel paths that are available to the CPC. By assigning control units, devices, and I/O paths to LPARs, the I/O configuration defines access rights for these entities.

This description is stored in a data set named I/O definition file (IODF) within the z/OS operating system. An IODF can define multiple hardware and software configurations to several operating system images. An IPL requires that you identify the IODF that contains the definition of your working configuration.

The IOCDS is used at power-on reset (POR) to load the IO configuration to a common memory area that is accessible by all images running on that CPC. This IOCDS is generated from an IODF data set in z/OS and is pushed into the SE by the Hardware Configuration Manager (HCM) function. Also, the IODF data set that is used by the operating system for initialization is a VSAM data set that is created from a readable IODF source within the HCM.

The IOCDS contains the configuration for a specific processor or CPC, and the IODF can contain I/O configuration data for multiple processors. The System z firmware acts as a driver to control the access to input and output devices through the channels for the respective images, as configured in the IOCDS.

Here are the considerations for protecting the IO configuration:

- ▶ Securing the IOCDS in the SE is already described in 8.2, “Hardware Management Console and Support Element” on page 189. Ensure that reading or disassembling the IOCDS from the SE or HMC is prohibited to restricted user IDs who are authorized for that purpose.
- ▶ It is a preferred practice to limit the images that can create or over-write the IOCDS from the IODF. Thus, IO management can be isolated to certain images that are accessible only to system programmers or hardware planners.
- ▶ Strictly restrict the access to export or copy the IOCDS file out of the SE or HMC to authorized IDs and channels. Use of external media with locally attached HMCs must be restricted and audited for authorized uses. Encrypted channels are a must for any such authorized use.
- ▶ Within z/OS, it is preferable to maintain the IODF source file only in those images with restricted access. After the production IODF is created from this image, it can be made accessible to all other images in the complex. Thus, the IODF source can be isolated and protected from unintended or unauthorized access.
- ▶ Production IODF data sets should be protected by using ESM rules. There is no need for access to any user IDs that access the IODF file, except for the operating system. All accesses to the IODF data set should be audited and reviewed regularly.
- ▶ Operator commands that target I/O functions, such as dynamic changes or activation of IODF configuration, should be strictly restricted to limited IDs. The IDs should be authorized through strict change control processes and justified with business needs before implementation. All such commands should be protected by ESM rules with conditional access profiles only. Even the **DISPLAY** commands for I/O configuration should be restricted only to those users who really provide justification for business need. The same protection criteria is also applicable to all MVS utilities that provide facilities to operate on I/O configuration or IODF data sets.
- ▶ System configuration data sets and members that point to the active or production IODF, such as the PARMLIB or IPLPARM, should be protected at equal or stricter levels as for the IODF production data set itself. This prevents easy identification of the active configuration for unauthorized people.
- ▶ All backups of IODF, both production and source data sets, should be encrypted and properly secured from unauthorized access.

8.4 Terminals and printers

Terminals and printers still serve as an effective means of communicating with the System z platform as input and output devices. Even with dumb terminals almost extinct and emulated terminals in wide use even for conventional or modern 3270 applications, printers are still extensively used for large-scale printing from mainframe applications.

With emulated terminals, most of the security aspects have moved to either the mode of network communication or the configuration parameters of the corresponding emulation software's security features. There are still a few organizations using the IBM 2074 console controllers to emulate non-SNA terminals that are used as master or secondary consoles. Recently, many of these console controllers were replaced OSA Express - Integrated Console Controllers or other forms of non-SNA remote consoles, but some IBM 2074 console controllers still exist. These console controllers are usually placed in proximity to System z or the system operators to provide the console management function using emulated non-SNA terminals. The console controllers connect to System z using ESCON and to the external network using high-speed Ethernet. Although physical security controls for these devices must be replicated, as with the System z itself, the emulated terminals are fully authorized master consoles that can control the z/OS or a sysplex through various commands, most of which can be disruptive. Again, you need protection either by using ESM rules that enforce controls on operator commands that are entered through master console, or by moving to new technologies that provide better access management for console terminals. There are ISV/OEM products that are available for such console management functions.

The basic security need for remote 3270 terminals of TSO, CICS, or IMS users is to ensure that the connection is encrypted. This feature is available in all available terminal emulation software products, but some configuration must be performed at both ends. Use some session management products as a gateway for accessing terminal applications because such products come with extensive security features that can be customized.

Mainframes are the most voluminous in terms of printing hardcopy information, and they are usually connected to high-speed large printers for this purpose. Originally, these printers were locally attached, but now often are controlled over wide area networks. Because there are multiple applications running in a sysplex that access these printers in a random fashion, there are also virtual printers that provide a transparent view to connect to physical printers. These services and functions are offered through various software products that run on z/OS. Still, as the printers are physically in various geographically dispersed locations, there should be a regular audit of active physical printers to avoid routing of critical information to incorrect printers. A defined housekeeping process for both physical and logical printers helps eliminating such errors and potential data theft.

The security of logical printers is usually managed by the features that are offered by the respective software product that manages the printers. This security is ensure the product's internal security mechanisms or through ESM. As JES also manages printers, it is necessary to implement effective controls over JES commands that operate printers.

Because the information is out in the open after it is printed, physical security becomes important for monitoring the dispatch of printed outputs to the correct destination or people. Unauthorized access and distribution of printed data leads to major security breaches and should be controlled through proper governance mechanisms.

8.5 Storage devices

Mainframes are known for their tremendous amount of data processing. In the past, auxiliary storage was expensive and there was less processing demand. In recent decades, new technologies gave rise to massive and efficient storage devices that can accommodate terabytes of data and provide faster access to them. Enterprise storage devices, such as the IBM System Storage DS8000, provide excellent reliability, availability, and serviceability (RAS) capabilities for System z. These devices are now shareable between both System z and other distributed platforms.

These enterprise storage devices usually have full disk encryption capabilities to protect the data that is stored on them. This encryption is managed by software products such as IBM Security Key Lifecycle Manager for z/OS. Most of these devices are RAID controlled, so the logical data is striped across different physical disks to provide recovery capabilities and performance enhancements. The access controls of the logical data that is stored within the disks is managed by the ESM installed on z/OS. The encryption capabilities secure the data in physical form in the controller.

These storage devices also come with their own HMC, which is used to configure and manage the disks and controllers. These HMCs, which are low-end servers with the appropriate software installed, are placed close to the respective hardware. These HMCs hold critical information, such as the configuration of the controllers or the encryption recovery keys. This setup demands that the HMCs be controlled by the same level of physical security and audit controls as for the devices themselves. As these HMCs can perform disruptive tasks that might impact the availability of data, access to these HMCs is strictly restricted only to authorized engineers with a clear technical and business justification. Roles on these HMCs, such as the storage administrator and the security administrator, should be defined to different individuals to serve their purpose. The policy for setting the password for other users of these HMCs should also align with the organization's security policies and password rules.

These storage devices usually allow monitoring through standard protocols such as SNMP. Proper care should be taken to ensure that the access to these devices through SNMP is restricted only to those authorized devices that are used for monitoring them. Regular audits are required to ensure that this is enforced throughout the life of the device as aligned with technological changes and organizational policies.

8.6 Tape libraries and removable media

Because there is a need to transport data from auxiliary storage for backup and recovery purposes, tertiary storage was introduced as removable tape devices. These devices have a long association with mainframe computers because mainframes were processing huge amounts of data compared to other platforms, and backup of that data was critical because of limited and expensive auxiliary storage.

As data storage needs increased, the need to back up data to tapes increased proportionally. This situation turned the focus towards ensuring the safety and security of data that was physically transmitted or transported between different locations. With many tape media in existence, many organizations have tape management products on z/OS to manage the tape inventory and scratch pools. These tape management products have a mechanism to prevent the reading of any tape that is "alien" to the system, which means that unless the tape label is known to its inventory, which is managed by the storage administrator, it cannot be read by the system, thus preventing any unauthorized data being loaded into the system.

These tape labels also are associated logically to the respective tape at the z/OS level and represent the same tapes within the operating system as well. But, there are ways that a storage administrator can override such a protection and an unlabeled tape can be read outside the tape management system, which increases the risk of data being stolen and read by unauthorized people on a different system. To avoid this problem, ESMs can protect data on tape so that it is protected from being read on a different system without the appropriate rules or passwords.

Here are few considerations and suggestions for protecting tape volumes and data:

- ▶ Protect tape data both at the volume level and at the data set level. Use either ESM or a tape management product to secure data on tape. Although tape management products provide flexible options and methods, small organizations with stand-alone tape drives might consider using ESM facilities to secure tape data. In some cases, ESMs such as RACF can work with products like DFSMSrmm to provide enhanced security for tape data and volumes.
- ▶ Tape data sets can be password-protected to ensure that they cannot be accessed on a different system without a valid password in its password database.
- ▶ Limit utilities such as IEHINITT to authorized users to restrict initialization of tape volumes.
- ▶ The authority to bypass label processing should be highly restricted to storage administrators and be used only with a valid business justification.
- ▶ Have all the tapes labeled and protected within the inventory for better management and protection.
- ▶ A policy can be put in to place to define retention for tape data so that the data is appropriately managed and the tapes can be effectively reused.
- ▶ Do not turn off security checking for any program in the program properties table because this allows bypassing of the authorization checking for tape data.

Although the aspect of security for tape data within the environment is a priority because of tape's mobile nature, it is also necessary to ensure that the data is not accessible in a different environment than which it belongs. Some of the above suggestions meet this requirement. Still, it is preferable to enable encryption for all tape data. Recently, tape encryption has become relatively simpler and efficient, which ensures that the data on tape cannot be tampered with or stolen without the appropriate encryption keys.

Some environments have virtual tape systems that emulate the tape devices and provide multiple levels of hierarchy for auxiliary storage. With ESMs such as RACF, the data set protection rules are applicable to any data set regardless of where they are, which makes it easier to apply or validate the same data set security policy for removable media with just a few additional considerations in terms of their mobility.

8.7 Network adapters

The network adapters, or the OSA cards, are fixed to the System z hardware. Apart from the security requirements that are being applied to the entire System z hardware, the only additional requirement that applies for the network adapters is about securing the cables running from them to the switch or the router. You must ensure that the premises of the System z and the switch or router is restricted physically to authorized people and the cable is secure enough that it cannot be inadvertently removed from its designated port at both ends.

8.8 Other peripheral devices

There are many other peripheral devices that are connected to System z. There were massive controllers for SNA communication, such as the IBM 3745 or IBM 3746 devices. These devices are no longer in operation, as they were replaced by technologies such as Enterprise Extenders. There were IBM 3174 cluster controllers that were used for telecommunication, mainly for terminals or printers. These devices were later replaced by IBM 2074 console controllers and IP-based terminals and printers. You can attach a blade extension to System z to accommodate distributed platforms and enjoy the RAS features of System z hardware.

Most of these hardware devices are placed in close proximity to the System z itself and so demand the same level of physical security as the System z. Other security aspects of the respective hardware must be reviewed based on their corresponding functions, and are mostly managed by their logical controls, such as their firmware or operating system. Ensure that the security configuration and controls on those devices align with the overall System z operation and functions.

8.9 Logging and auditing for hardware

Part 1, “Direction and architecture” on page 1 described the extensive logging and auditing capabilities of System z and z/OS. With such complex hardware in operation that manages multiple operating systems that host critical business applications, you must ensure a high level of monitoring to investigate any potential issues or security breaches.

System z hardware works with the z/OS and reports any anomalies to it immediately. System z has a robust *First Failure Data Capture (FFDC)* technology for its components that captures any issues with the hardware and report them directly to the IBM Service Center and the system operators. Hardware errors and issues also are reported to the *Environmental Record Editing and Printing Program (EREP)*, which is a diagnostic program on z/OS, z/VM, or z/VSE, which helps maintaining the error recovery process of the operating system with respect to data processing.

The System Management Facility (SMF) can record many system events that relate to hardware. Some of these events might expose a gap in security and need immediate attention. Hence, it is critical to ensure that such SMF records that capture this critical information are active and regularly reviewed to report any anomalies.

The HMCs and SEs regularly highlight hardware errors, including the processor interrupts and wait states. It should be a preferred practice to review and analyze regularly these errors to capture the reasons and actions for them. These errors are a good clue to discovering security breaches and potential threats on the system that should be acted upon appropriately. Such a review should be performed by authorized engineers and a report submitted to the organization’s security engineering department.

It is a common practice to disable the logging or auditing features of IT infrastructure components to benefit performance. However, this leads to missing audit or log information that is needed if there is a serious breach, which leads to a much higher cost than is warranted by the performance benefits of turning off logging or auditing. The impact of security breaches is more wide-ranging today, especially for System z, which hosts most of the business-critical applications.

Although the possibility of security breaches is not zero, the ability to investigate the cause of a breach and provide the appropriate action to fill the gap lies in the availability of proper diagnostic information that is provided by these logging and auditing features. Recently, these functions have advanced, and they do not consume as much processing power as they did. These functions are proving to be more efficient and beneficial in reducing the impact of a security breach than the performance benefits that are associated with them.

8.10 Conclusion

The System z hardware is designed to deliver new levels of performance and capacity for large-scale consolidation and growth. It can provide support for the next generation of digital signature security support, cutting edge pattern recognition analytics for smart monitoring of system health, and new environmental capabilities. System z also extends its advanced technology to the security functions. Even the other hardware components that are associated with System z align themselves with the security levels that are provided by System z.

Still, the responsibility lies with the organization to enable and use the facilities that are offered by System z to secure their environment. Compared with logical security controls that can be enforced mostly by policies and rules, hardware security has much to do with physical security aspects in addition to processes and procedures that are associated with physical access controls. This is also supplemented by a good monitoring system for logging and auditing of any exceptions. System z helps achieve these things by being the most flexible platform that uses robust and advanced technology.



IBM z/OS security

This chapter examines many of the security-related controls that exist in z/OS and provide advice about how to choose settings that have a bearing on security and integrity. RACF settings are also examined and the significance of settings is highlighted.

This chapter looks at major components of z/OS and provides many recommendations to bring the reader's attention to some matters that were found to be issues in the past. Many of these issues are related to an older style of management of IBM mainframes that reflects how they were managed in previous decades. For more information about this older style, see Chapter 3, "Mainframe security architecture in the enterprise" on page 29.

This chapter covers the following topics:

- ▶ Introduction
- ▶ z/OS settings
- ▶ z/OS system routines
- ▶ z/OS component protection
- ▶ The IBM Health Checker for z/OS
- ▶ RACF settings
- ▶ JCL procedures
- ▶ UNIX System Services
- ▶ DFSMS
- ▶ JES and SDSF
- ▶ TSO/E
- ▶ Integrated Cryptographic Service Facility
- ▶ Started procedures
- ▶ Special procedures for privileged users
- ▶ Multiple LPARs and shared DASD

9.1 Introduction

Chapter 7, “Organizing for security” on page 169 notes that it is necessary to specify or choose a set of standards for security settings.

These standards might follow internal specifications for the organization, or they might be based on external regulatory controls. Several points are important at this stage:

- ▶ A set of standards exists.
- ▶ The standards are appropriate to the platform and the nature of the business.
- ▶ Some kind of analysis of the gap between your current system and the standard has been performed to ensure that you understand where there is a need for change.

This chapter highlights some of the things that the standards should be addressing, and also some things that they might not address, but that are important for the security architect to be aware of. Examination of many of these matters can lead to the identification of security exposures that might not have been addressed simply by conforming to the standards.

This chapter does not give you all the settings and values that are needed to configure a z/OS system for high security. Sometimes we, the authors, are specific; at other times, we direct you to an approach, and possibly away from an alternative approach that is more problematic.

You are expected to do more research to reach the definition of the necessary set of security controls for your z/OS system.

9.1.1 Finding security gaps

You might well ask how some of these security gaps can be found in your z/OS installation.

In some cases, this task requires insight into how z/OS behaves, insight into how security is managed in your organization, and insight into the nature of the applications that you run for that organization.

However, in addition, there are tools that can determine whether you have such security gaps. One such tool is IBM Security zSecure Audit.

Security zSecure is introduced in 6.2, “IBM Security zSecure Suite” on page 128. Security zSecure Audit has components that can be used to find what are *concerns* where security improvements can be made. Security zSecure grades these *concerns* so that a relative importance can be given. There are internal knowledge bases within Security zSecure so that the concerns can be associated with one of several security targets. The targets that are available are Security zSecure, C1, C2, and B1. The last three are associated with the US Department of Defense Orange Book, which is a standard available at the following website:

<http://csrc.nist.gov/publications/history/dod85.pdf>

use of the Security zSecure Audit component can identify system components that are exposed, or show system settings that can be far tighter.

Security zSecure Audit also includes a compliance framework that assists in the definition of standards, and can assist in the measurement against those standards.

IBM a program that is called TASID0. This program may be obtained at the following website:

<http://www.ibm.com/support/docview.wss?uid=swg24009131>

TASID0 can display some of the settings that are needed for an analysis of z/OS.

There are other places within this chapter where we do not specify exactly what value should be chosen, as this is typically the role of the security standard in use.

Disclaimer: The authors believe the recommendations that are given in this chapter are all sensible measures for securing your z/OS operating system. However, we do not claim that this is *all* the measures that must be taken. Each z/OS system is different, and requires distinct analysis.

9.1.2 The magic ingredient

Each z/OS system is different. It hosts data that has a different sensitivity or classification. It is run differently. Many configuration parameters are different.

However, whenever a change is made, those responsible for security must make a judgment about whether this change alters the risks being taken by the owners of the system and data.

For each configuration value, and particularly for each configuration change, it is necessary to determine what someone with malicious intent might do if they knew about this setting or could exploit this setting.

You also might want to think about whether restricting access to a bare minimum is a way of gaining an early view of attempts to gain system access. If CICS users do not need access to TSO or batch-related resources, then you can generate access violations when someone tries to use a CICS user ID to access TSO or batch. So, if an attacker gains access to a CICS user ID and attempts to use it for TSO access or attempts to use other libraries in batch, you are alerted by the failed access attempts.

Thinking this way does not come easily to some systems programmers and administrators. It is a skill that must be cultivated. It is needed if you are to understand where your system has weaknesses and vulnerabilities.

So the magic ingredient is *thinking like your enemy*. Each configuration value must be evaluated as to whether it helps or hinders someone with malicious intent to gain access to your system, or if it helps or hinders those with some access to your system to gain more access.

It sometimes takes several people to think about settings in different ways. A discussion is frequently required to bring out all the security aspects of a change. Above all, remember that security settings are not limited to RACF changes. There are many z/OS settings that have a bearing on security.

Thinking like a security person achieves only so much. To progress, you must think like the opposition. At each stage, think how you can use a setting or control.

Change is something that we all must live with. With each change, there is an opportunity for an error to occur. Consider therefore how *little* must change for a protected item to be exposed.

This book does not tell you all the things that you must do to make your system secure. Every z/OS installation differs in some way from every other z/OS installation. As each installation has a different set of security requirements, it is necessary to *think* carefully about what might constitute an unacceptable risk to you.

9.1.3 Third-party software

Most organizations that use z/OS use one or more software products that are not supplied by IBM. Some organizations have software of their own that they maintain.

If the software uses an APF-authorized function or needs routines added to z/OS libraries, such as system exits, then there is a possibility that the integrity of the operating system can be compromised.

For code that is maintained by staff at the enterprise, they are expected to follow the correct programming rules that are documented in *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608. Chapter 21, “Protecting the system”, of *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608 has many rules about how authorized programs should be coded to protect system integrity.

For code that is supplied by a third party, there must be an element of *trust*. The code that is supplied by that third party must be trusted by the organization. If there is reason for the organization not to trust code from any source, then it needs to be identified as a *risk*.

If software is supplied free, then the source code should be carefully examined. If the source is not available, then the risk you are taking should be carefully assessed according to the degree of trust that is placed in the supplier.

Preferred practice: Ensure that you know the provenance of all APF-authorized and system code that you install. If possible, get statements of assurance from the suppliers.

9.2 z/OS settings

This section highlights settings in z/OS that can have an impact on security or integrity. In some instances, these settings are selected through programming interfaces, and others are selected through settings in SYS1.PARMLIB¹ or some other mechanism.

This section does not explain everything about the various members of SYS1.PARMLIB. For the full details of syntax and use of the members, see *z/OS MVS Initialization and Tuning Reference*, SA22-7592-22.

9.2.1 Consoles

z/OS uses consoles for controlling access to its operator commands. These commands should not be confused with TSO commands or with machine instructions.

In general, z/OS requires a console of some type to enter commands and to display messages (possibly in response to those commands). Under some circumstances, commands can be run from within authorized code and can appear to have come from a designated console. The commands are known as MVS commands and are described in *z/OS MVS System Commands*, SA22-7627-26.

¹ It is possible to concatenate libraries to SYS1.PARMLIB or otherwise alter the library that is designated as SYS1.PARMLIB. This chapter always refers to SYS1.PARMLIB and presumes that it to the set of libraries, unless we need to specifically refer to members of the concatenation.

There are many z/OS commands that are used for controlling devices and for displaying the progress of work through the system. There are also many commands that can be used to alter the state of the system, and alter settings that affect system integrity or system security. It is essential, therefore, that access to consoles and commands is tightly controlled.

Unlike TSO sessions, it is not necessary for users to identify themselves when accessing a physical console. However, this is a configurable setting within the CONSOLxx member of parmlib. In times past, access to physical consoles was controlled through physical controls. The consoles were kept within a secured computer room and only those within the computer room had console access. However, there are many types of console in z/OS now, and some of them need physical controls, and others need logical controls.

Preferred practice: The authors have observed z/OS installations where physical consoles are now used in open offices, with little or no control over who can access them. Be sure that you have accountability for the use of consoles. If consoles are in open offices, then they should require a logon to use and should never be left unattended while logged on.

Here are the types of consoles that are available:

▶ Multiple Console Support

Multiple Console Support (MCS) goes back to times when there was only one physical console that was attached to the operating system. MCS consoles were then brought in as a feature to allow multiple consoles on the same system. Today, this term refers to consoles that are physically attached to the mainframe through a local controller or an OSA-ICC link.

▶ SNA MCS

SNA MCS (SMCS) is accessible through an SNA link into console support. SMCS consoles require that each user log in by supplying a user ID and password. Thus, commands are attributable to individuals.

▶ Hardware MCS

The Hardware MCS (HMCS) console is accessed through the HMC. It can be configured to require logon, to allow logon, or to have automatic logon.

▶ Extended MCS

An Extended MCS (EMCS) console is a logical console that can be used within a program to route commands and receive messages.

▶ Printer

Consoles can be defined as printer consoles. They display messages, but do not take commands.

▶ Subsystem

Subsystem consoles were an older method of achieving what is now done with EMCS consoles. Each subsystem console needs a definition in the CONSOLxx member.

▶ System console

A system console is associated with the System z hardware.

Preferred practice: Anyone outside of the day-to-day operators should be forced to use a logon when using a console. This enforces accountability for commands that are run. If practical, a logon should be enforced on all consoles.

9.2.2 System Management Facility

System Management Facility (SMF) provides a high performance logging capability for recording events that occur during z/OS operation. Some of these are events that are associated with work running and the consumption of resources. Other types of records hold information about data set use and other items.

SMF records each have a *type*, and many have a *subtype*. Records that are written by RACF are types 80 and 81. SMF also has a series of *system exits*, which are described in 9.4.10, “SMF data sets” on page 225.

Controls for the use of SMF are in the SMFPRMxx member of SYS1.PARMLIB. In this member, you can suppress SMF record types or subtypes. For security purposes, it is important that many record types are *not suppressed*.

Many security standards for SMF in z/OS systems specify which record types and subtypes must be recorded, so we do not specify anything absolute here. However, there are some suggestions in 9.4.10, “SMF data sets” on page 225.

Preferred practice: Record types that are associated with creating, reading, modifying, renaming, and deleting of data sets should not be suppressed. *All* SMF records that are written by RACF should *not be suppressed*.

During normal system running, SMF records are collected either by using the z/OS component that is called the System Logger, or by using the older method of storing them in VSAM data sets that are allocated to z/OS. However, after they are unloaded, they must be stored for archive purposes. These archive data sets need careful management.

9.2.3 MVS commands

Consoles are the main method of running MVS and JES commands. You can use these commands to perform many different types of operations. Most commands fall under one of the following categories:

1. Displaying and altering the status of devices or tasks
2. Displaying and altering the status of jobs' scheduling and the status of spool data sets (using JES commands)
3. Altering the configuration of the operating systems by changing items, such as subsystem definitions, console definitions and authorities, linklist configurations, APF list definitions, and by introducing new exits and Link Pack Area (LPA) modules
4. Running and configuring trace components, such as SLIP

It should be clear that many of these capabilities must be secured. Even the normal operators do not need access to many capabilities that can be used to alter or subvert operating system security.

So, the ability to perform many of the operations that are listed in items 3 and 4 should not be given to those day-to-day system operators.

All MVS and JES commands can be protected by using the OPERCMDS class. The RACF profiles that are used to protect these devices are described in *z/OS MVS System Commands*, SA22-7627-26.

However, when protecting these commands (and all of them should be protected), consider the following principles.

- ▶ Divide the commands in to those commands that your day-to-day operators must use and those commands that are more systems programming oriented. So, commands that alter operating system configurations, such as **SET**, **SETSMF**, and **SETPROG**, should be restricted to a small group of systems programmers.
- ▶ Most commands that display resources fall into the operators group. However, do not grant everyone access to these commands. Giving all users access to these commands (which can be entered by multiple means) simply makes it easier for an attacker to gain knowledge of your system. An example is the **DISPLAY PROG,APF** command, which shows an attacker all the APF libraries on the system.
- ▶ Pay careful attention to the **SETPROG** command. The profiles that are necessary to secure the various functions of this command are shown in *z/OS MVS Planning: Operations*, SA22-7601-13.
- ▶ Pay careful attention to the access *levels* that are required for access to each command. These levels vary by command. The authors regularly see access levels that are set too high for commands, sometimes at a level that makes no sense, while at other times **ALTER** access is granted to a discrete profile. This allows modification of the access list of the resource, and is rarely what is required.
- ▶ Examine the following JES2 and JES3 manuals to ensure that commands for those subsystems are similarly controlled. The following manuals contain similar tables:
 - *z/OS JES2 Commands*, SA22-7526-12
 - *z/OS JES3 Commands*, SA22-7540-11

SETPROG

The **SETPROG** command is secured by using a set of profiles in the FACILITY class.

APF list protection example

To secure the ability to define or remove an APF library named SYS2.LINKLIB, check access to a resource named CSVAPF.SYS2.LINKLIB in the FACILITY class. UPDATE access is needed to add or remove such a library from the APF list.

Dynamic Exits protection example

To secure the ability to define system dynamic exit IEFUTL, check access to a resource named CSVDYNEX.IEFUTL.DEFINE in the FACILITY class. UPDATE access is needed to add or remove such an exit definition.

9.2.4 Program Properties Table

Another member of SYS1.PARMLIB that is significant is the member SCHEDxx. This member contains a table that is known as the Program Properties Table (PPT).

Using this table, it is possible to assign special properties to the programs that are specified within it. However, these properties are only accepted if the program is loaded from an APF-authorized library.

There are other programs in the PPT besides those that are specified in the SCHEDxx member. These programs are hardcoded in the z/OS module IEFSDPPT. This module is in the IEFSDPPT load module. It is stored in SYS1.LINKLIB.

The entire PPT is built by merging the entries in the SCHEDxx member with those in the IEFSDPPT module.

If you run the DDLIST TSO command, it is possible to see the IEFSDPPT module. While in TSO and ISPF, run the following command:

```
DDLIST;LOAD IEFSDPPT2
```

This command shows a panel stating that the module already is loaded. Press Enter and then a panel similar to Figure 9-1 is displayed. On the right side, the panel shows the programs and their special properties. A hex display of each 16 bytes entry is shown on the left side.³

```

BROWSE   IEFSDPPT JPA Start:0008E948 Size:000006B8 Line 00000000 Col 001 080
Command ===>                               Scroll ===> HALF
***** Top of Data *****
+0 (0008E948) C9C5C4D8 E3C3C1D4 6060FFFF 20800000 * IEDQTCAM--...Ø.. *
+10 (0008E958) C9E2E3C9 D5D4F0F1 EA60FFFF 20800000 * ISTINM01?-...Ø.. *
+20 (0008E968) C9D2E3C3 C1E2F0F0 D860FFFF 00800000 * IKTCAS00Q-...Ø.. *
+30 (0008E978) C1C8D3C7 E3C64040 E800FFFF 20800000 * AHLGTF Y....Ø.. *
+40 (0008E988) C8C8D3C7 E3C64040 E800FFFF 20800000 * HHLGTF Y....Ø.. *
+50 (0008E998) C9C8D3C7 E3C64040 E800FFFF 20800000 * IHLGTF Y....Ø.. *
+60 (0008E9A8) C9C5C6C9 C9C34040 D800FFFF 00800000 * IEFIIC Q....Ø.. *
+70 (0008E9B8) C9C5C5D4 C2F8F6F0 EE00FFFF 00800000 * IEEMB860Ø....Ø.. *
+80 (0008E9C8) C9C5C5E5 D4D5E3F2 C800FFFF 00800000 * IEEVMNT2H....Ø.. *
+90 (0008E9D8) C9C1E2E7 E6D9F0F0 C810FFFF 00800000 * IASXWROOH....Ø.. *
+A0 (0008E9E8) C3E2E5E5 C6C3D9C5 6800FFFF 00800000 * CSVVFCREÇ....Ø.. *
+B0 (0008E9F8) C8C1E2D1 C5E2F2F0 ED10FFFF 00800000 * HASJES20Ø....Ø.. *
+C0 (0008EA08) C4C6E2D4 E5D9C3F0 6870FFFF 00800000 * DFSMVRCOÇø....Ø.. *
+D0 (0008EA18) C9C1E3C9 D5E3D240 ED10FFFF 00800000 * IATINTK Ø....Ø.. *
+E0 (0008EA28) C4E7D9D9 D3D4F0F0 6870FFFF 00800000 * DXRRLM00Çø....Ø.. *
+F0 (0008EA38) C1D7E2D7 D7C9C5D7 6C10FFFF 20800000 * APSPIEP%....Ø.. *
+100 (0008EA48) C1D2D7C3 E2C9C5D7 6C10FFFF 20800000 * AKPCSI EP%....Ø.. *
+110 (0008EA58) C9C1E3C9 D5E3D2C6 6C10FFFF 00800000 * IATINTKF%....Ø.. *
+120 (0008EA68) C4E2D5E8 C1E2C3D7 6870FFFF 00800000 * DSNYASCPÇø....Ø.. *
+130 (0008EA78) C4E2D5E4 E3C9D3C2 4070FFFF 00800000 * DSNUTILB ø....Ø.. *
+140 (0008EA88) C9C5C1E5 E3C4E2E5 5A00FFFF C0800000 * IEAVTDSV!...{Ø.. *
+150 (0008EA98) C9C6C1E2 D4C64040 FC00FFFF 00800000 * IFASMF Ü....Ø.. *
+160 (0008EAA8) C3E2E5D3 D3C3D9C5 6A00FFFF 00800000 * CSVLLCRE!....Ø.. *
+170 (0008EAB8) C1E5C6D4 D5C2D3C4 E830FFFF 20800000 * AVFMNBLDY....Ø.. *
+180 (0008EAC8) C5D9C2D4 C6D4C6C3 2C80FFFF 00800000 * ERBMFMFC.Ø....Ø.. *
+190 (0008EAD8) C5D9C2F3 C7D4C6C3 2C80FFFF 00800000 * ERB3GMFC.Ø....Ø.. *
+1A0 (0008EAE8) C9C7C7F0 C3D3E7F0 FC00FFFF 80800000 * IGGOCLX0Ü...ØØ.. *
+1B0 (0008EAF8) C9C7C4E2 E2C9F0F1 EA50FFFF 00800000 * IGDSSI01?&...Ø.. *
-GIMQIX1 *DDLIST

```

Figure 9-1 Display of the IEFSDPPT module

The attributes that can be assigned to a program include many that are related to performance, such as whether they are treated as swappable, and whether they are timed.

² We assume that the command delimiter character in ISPF is the semicolon character. If this is not true, then you should replace this character with whichever character is the delimiter character in your ISPF session.

³ If required, the meanings of the hex display can be deciphered with the aid of the IEFZB610 macro.

There is one property that we are concerned with for security. This is the **NOPASS** parameter. **NOPASS** specifies that RACF should not be consulted when data set operations take place. Each installation should take careful note of all the entries in the PPT that specify **NOPASS**. The provenance of each entry should be established, and steps taken to ensure that the correct properties are applied.

If you are using z/OS V2.1, you use the new MVS system command **DISPLAY PPT**. It can be easy for a malicious person to alter the entry for a program to add the **NOPASS** operand, thus granting access to all data sets, without RACF access controls or auditing. The SMF records for opening and processing data sets are still written if not suppressed in SMFPRMxx.

Preferred practice: The authors have noted several abuses of the PPT. A common abuse is the use of a program called SMFDUMP. This program has been distributed on the Internet, and has instructions to place an entry in the PPT with the **NOPASS** property and copy the program to a linklist library. The program is designed to assist with managing the unloading of SMF data.

As SMFDUMP may be used with all sorts of other data sets than SMF data, it can be used to overwrite almost any data set on the system if it is given the **NOPASS** property. Remove the **NOPASS** property from this copy of SMFDUMP.

The entire PPT can be displayed with Security zSecure Audit, as shown in Figure 9-2. The program that is named DODGYPGM has been added to the program properties table with the **NOPASS** property.

Program Property Table											
Command ==>										Line 1 of 14	
										Scroll==> HALF	
29 Aug 2013 20:49											
Complex	System	Count	Audit	concerns	Priority						
SYS1	SYS1	102		89	21						
Pri	Program	Key	Bypass	NoDSI	Modif	NonSwap	NonCan	Priv	Systask	IEFUSI	Audit
21	DODGYPGM	8	Bypass		Modif					IEFUSI	Bypas
Program name (must be APF)				DODGYPGM							
Job step storage key				8							
Bypass password / SAF				Yes							
No data set integrity				No							
Default entry IEFSDPPT				No							
Non-swappable				No							
Non-cancellable				No							
Privileged (no SWAP)				No							
System task not timed				No							
IEFUSI Region limit honored				Yes							
Audit concern											
Bypasses SAF, Modified from IEFSDPPT											

Figure 9-2 PPT entry that is displayed from Security zSecure Audit

The PPT can be altered dynamically with the `SET SCH=nn` MVS system command. The command sets the PPT from the SCHEDnn member. Thus, to alter the current PPT, you must have the rights to run this command. To set any arbitrary program name to have the `NOPASS` parameter, it is also necessary to have UPDATE authority to SYS1.PARMLIB. However, for the `NOPASS` property to be accepted, it also is necessary for the program to be stored in an APF-authorized library.⁴

9.2.5 Subsystem definitions

Within the heart of z/OS is the *subsystem interface*. Unlike the rather loose use of the term that was used in Chapter 7, “Organizing for security” on page 169, we have a far tighter definition here that is taken from *z/OS MVS Using the Subsystem Interface*, SA22-7642. This definition is as follows:

“A subsystem is a service provider that performs one function or many functions, but does nothing until it is requested. Although the term *subsystem* is used in other ways, in this section a subsystem must be the master subsystem or be defined to MVS in one of the following ways:

- ▶ Processing the IEFSSNxx member of SYS1.PARMLIB during IPL

You can use either the keyword format or positional format of the IEFSSNxx member of SYS1.PARMLIB. IBM recommends that you use the keyword format, as this allows you to define and dynamically manage your subsystems.

- ▶ Issuing the IEFSSI macro
- ▶ Issuing the SETSSI system command”

The subsystem interface in MVS is a formal mechanism that is used to communicate between two or more software components that might exist in different address spaces.

Every MVS system contains a Master Subsystem and a Primary Subsystem. The Master Subsystem is called MSTR while the Primary Subsystem is normally JES2 or JES3.

Associated with each subsystem is a set of control blocks that are in common storage (CSA, ECSA, SQA, or ESQA). Within those control blocks are settings that determine which types of *subsystem call* are accepted by the subsystem. The subsystem calls are designated with numbers that are known as *SSI numbers*.

Risks are associated with subsystems that have some or all of their control block structure unprotected.

Preferred practice: The authors often have seen subsystem definitions that have a Subsystem Communications Table (SSCT) that contains pointers to fields SSCTSUSE and SSCTSUS2, each of which points to a field that is in unprotected common storage area (CSA) storage. This type of structure is open to abuse because *any* address space can modify the contents of those two fields. If you discover that you have this type of structure, you should contact the vendor of the software that created it to determine why this has been done. In our experience, this type of problem can be fixed in a secure manner.

This type of faulty structure is not easy to find. Fortunately, Security zSecure Audit can detect such situations and report them.

⁴ A malicious user might choose to alter the PPT entry of a system utility, such as IDCAMS, thus enabling the updating of almost any data.

Figure 9-3 shows a Security zSecure Audit display of a subsystem definition that uses SSCTSUSE to point to an unprotected area in the CSA. The display shows that the storage is in subpool (SP) 241, which is defined as CSA. The storage key is shown as 8. These fields are highlighted by the red loop. Key 8 storage can be modified by any address space.

```

Line 1 of 10
Subsystem Communication Vector Tables
Command ==> Scroll==> HALF
                                current settings
Complex System Count Audit concerns Priority
██████████ 136 2 20
Pri Name Org Type MFn FIB PSS ARD SUSE dat SUS2 dat Description
20 IDIS 84 1 No Yes No ██████████
Pointer Address Where Key SP Eye catcher
SSCT 00A90F80 CSA 0 241
SSCTSUSE 00A19E90 CSA 8 241 ██████████
SSCTSUS2 00000000
SSCTSSVT 1B68A0A0 ECSA 0 241
Function Address Where Key SP Program Description
254 1B8CBDA0 ECSA 0 241 Installation-defined
Audit concern
User pointer in user-key common storage

```

Figure 9-3 Subsystem definition showing SSCTSUSE pointing to unprotected storage (some fields redacted)

9.2.6 PROG00

The PROGxx member of SYS1.PARMLIB contains many different lists that are important to the management of z/OS integrity:

- ▶ List of libraries that are APF-authorized
- ▶ List of libraries that form the system LINK LIST
- ▶ List of libraries that form the system LPA LIST
- ▶ Settings for the management of system exit routines that are associated with dynamic exit points

These lists all have a significant bearing on system integrity.

The authors' experience shows that management of APF lists is a typical area that customers should focus on. It is a known area for examination for those who want to break in to systems, and is frequently not well-managed. Good management requires co-operation between systems programmers and security administration staff.

APF-authorized list

This is a list of program libraries that are treated as APF-authorized.

An APF-authorized program is a program that can use the MODESET SVC (SVC 112) to alter the program's *state* (from problem program state to supervisor state, or vice versa) or the PSW key. The PSW key is normally 8, but may be altered to some other value 0 - 15.

Let us look at a recap of the keys:

- ▶ Key 0 is recognized by the System z hardware as being capable of accessing and modifying storage in any key.
- ▶ Keys 1 - 7 are treated by z/OS as system keys, and programs operating in these keys are treated as authorized. By convention, these keys are associated with specific products and components. For example, Key 6 is used for communications software and networking.
- ▶ Key 8 is the normal key for programs that have no special authority. Programs that are started by z/OS start running in Key 8 unless some special actions are taken.
- ▶ Key 9 is a storage key that, when associated with a memory page, allows any other key to change it.
- ▶ Keys 10 - 15 are reserved for programs that run *real*.⁵

Any program that is started by an APF-authorized program normally must come from an APF-authorized library.

Here are the full set of programs that are treated as APF-authorized by z/OS:

- ▶ SYS1.LINKLIB.
- ▶ SYS1.SVCLIB.
- ▶ Programs in the APF list of libraries.
- ▶ If LNKAUTH=LINKLIST is set in IEASYSxx, programs that are in the libraries comprise the system linklist. This applies only when those programs are accessed as part of the linklist.
- ▶ If LNKAUTH=APFTAB is set in IEASYSxx, programs that are in the libraries SYS1.LINKLIB, SYS1.SVCLIB, SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE are included.
- ▶ Programs within the Link Pack Area.

The set of libraries in the APF list can be modified by using operator commands.

The APF list also can be specified in the IEAAPFxx member. This is an older technique, and the preferred practice is to use the PROGxx method.

System linklist

The system linklist is a set of libraries that act as a default place to be searched when seeking a program to run. By default, the programs in this set of libraries are treated as APF-authorized, but a setting of LNKAUTH=APFTAB can be used to override this authorization.

TSO commands are normally placed in the system linklist, as are many other programs.

You must ensure that libraries are placed in the correct order. If two programs of the same name are in different libraries in the system linklist, then normally only the program in the higher library is run.

The set of libraries that make up the system linklist can be changed by using operator commands.

⁵ For more information, see a description of the IEASYSxx VRREGN parameter and the ADDRSPC=REAL parameter on the JOB card in JCL.

System LPALIST

The programs that exist in the set of libraries that is known as the LPALIST are loaded into z/OS memory during system initialization. These programs are then available to all address spaces.

The LPA is the name that is given to the storage area where these programs are. Programs that are in the LPA must follow specific programming rules. In particular, they must be reentrant and refreshable.⁶ Programs in the LPA are treated as coming from an APF-authorized library.

SYSLIB

A **SYSLIB** statement in the PROGxx member may be used to change the default name of some specific libraries that are a normal part of the system linklist:

- ▶ LINKLIB data set (defaults to SYS1.LINKLIB)
- ▶ MIGLIB data set (defaults to SYS1.MIGLIB)
- ▶ CSSLIB data set (defaults to SYS1.CSSLIB)
- ▶ LINKLIBE data set (defaults to SYS1.SIEALNKE)
- ▶ MIGLIBE data set (defaults to SYS1.SIEAMIGE)
- ▶ LPALIB data set (defaults to SYS1.LPALIB)

EXIT

EXIT statements in PROG00 are used to associate program modules with dynamic system exit points.

Care with these lists

To maintain the integrity (and the security) of z/OS, it is necessary to control access to these lists. Thus, the security of SYS1.PARMLIB is essential.

Security also is required to restrict access to the commands that are used to modify these lists.

Preferred practice: Security administrators need to work with systems programmers to ensure that the contents of these lists of program libraries are maintained correctly. Update access to these libraries to be defined for each library, independent of the RACF controls.

Preferred practice: Define a RACF profile for each library within these lists. Monitor all update activity to these libraries.

Some of the worst security mistakes the authors have seen are associated with the management of APF-authorized libraries. Among various situations, we have seen the following scenarios:

- ▶ APF list entries for data sets that do not exist
- ▶ APF list entries for data sets with a UACC of **ALTER**
- ▶ APF list entries for data sets that are unprotected by RACF
- ▶ APF list entries on volumes that are accessible by other systems using a different RACF database

All the above situations place a high risk on the security of all data in a z/OS system. It is the responsibility of every installation to securely manage its APF lists. Failure to manage these lists properly is a major cause of exposure.

⁶ A reentrant program may be executed simultaneously by two tasks or threads without interference. A refreshable program may be reloaded from pageable memory at any time. It is never paged out.

Preferred practice: Manage your APF lists with great care. Double-check entries. Do not leave dead entries in the list for simplicity or ease of use. Use a formal checker for the lists if possible.

Security zSecure Audit can find badly managed APF library situations. You should use this facility if you have Security zSecure Audit.

Figure 9-4 shows a panel from Security zSecure Audit of a situation where library is defined with a UACC of **UPDATE**. This allows all users (even those users who have no RACF identity) to update this library, which might compromise security.

```

Sensitive data trustees
Command ==>
Line 1 of 1
Scroll==> PAGE
5 Sep 2013 11:18
Pri Complex System Trust relations
49 SYS1 SYS1 6529
Pri Sensitivity Class Resources Trust relations
49 APF library DATASET 9 401
Pri Resource VolSer Trust relations
49 TEST.USERAPF USER00 1
Pri Userid Name From Audit concern
49 ???????? SYS1 May change APF program that can b
Pri Privilege
49 UACC

```

Figure 9-4 Security zSecure Audit report

The message is truncated in Figure 9-4 because the logical display is too small. The full message reads May change APF program that can bypass security, even outsiders may exploit.

9.3 z/OS system routines

This section provides details about z/OS system routines.

9.3.1 Supervisor calls

Supervisor calls (SVCs) routines are called to perform some system service. These routines can be started from code in any state (supervisor or problem program), but they always commence execution in the supervisor state. They are carefully coded to ensure that the service that they perform is well-defined. It is the responsibility of the SVC routine to ensure that the caller has the correct authority to perform the action that is requested.

For example, the OPEN SVC is SVC 18. It is the routine that is used to open a file or data set. It is the responsibility of this routine to ensure that the user (that is, the requester) has access to the data set being opened, at the level requested (for example, READ or UPDATE).

In other cases, it might be that an SVC is allowed to be started only if the caller is already APF-authorized. If so, the SVC makes this check and then terminates in some way if the required APF authorization does not apply.

Most SVCs are supplied as part of the z/OS operating system. Some of the major subsystems (such as CICS and IMS) also use SVCs. Each SVC is a number 0 - 255. Numbers 0 - 199 are reserved for z/OS, while numbers 200 - 255 are reserved for users. Some user SVCs are used by third-party software products. They also may be used by locally developed code.

SVCs are defined to an operating system by using the IEASVCxx⁷ member of SYS1.PARMLIB.

An SVC routine can be used to subvert the security of z/OS. This possibility arises when the SVC routine starts running in the supervisor state and key zero, and therefore can issue most system control instructions and modify most storage without referencing storage key controls that are placed upon other storage keys. This is an example of *privilege escalation*. Thus, it is important that SVC routines are coded with the *utmost care*.

Each installation should have clear documentation of which user SVCs are in use, and know the provenance of those routines.

Some badly coded SVC routines can be discovered by using Security zSecure Audit. Security zSecure can detect certain sequences of machine instructions, and can highlight short SVCs. If an SVC is short, it is highly unlikely that it has been able to perform all the necessary checks that are required to ensure that the caller is correctly identified and is in the correct system state.

It is possible to write an SVC that alters the APF authorization state of the caller. Normally, this is not something that should be done. Using such a routine, it is possible for a program from outside an APF-authorized library to gain APF authorization, and from there move into key zero and change security settings. Such SVCs are sometimes called *magic SVCs*. The presence of such a routine in a z/OS configuration represents an integrity exposure.

⁷ SYS1.PARMLIB might contain many members that are used to configure z/OS. These members typically have a two character suffix that can be specified. To represent many of these members, we use the suffix xx. Thus, IEASVCxx represents the currently active IEASVC member.

Figure 9-5 shows a panel from Security zSecure Audit. It shows an entry for a magic SVC that has been deliberately coded. Security zSecure Audit has noted the presence of potentially dangerous instruction sequences and the small code size.

```

Supervisor Call Audit Display
Command ==>
scroll right for more info
28 Aug 2013 21:53
Complex System Routines SVCs ESRs Audit concerns Priority
SYS1 SYS1 140 107 33 131 37
Pri SVC ES# APF Where K SP Program U Sf InstrSc Function
37 221 No EPLPA IGC0022A 1 00 A
Idx Where Key SP Program InstrSc Eye catchers
CN I EPLPA IGC0022A A ..
0 ENUC RO IGCERROR ..4x4..j 0..7j ..
Appl Result

SVCUPDTE Sf Last update Caller Where Module
1 00 00074E48 PVT
Index Typ APF ESR Att Locks
Current: 3/4 No No
Old: 2 No No
Expect: ??? ???
Instruction/Str/SVC scan results
FakeAPF No
Audit concern
Instruction scan hit, Dangerous code size, Updated using SVCUPDTE,
Installation-defined SVCno, Caller may be unauthorized
First 256 bytes of SVC
0000. 582040B4 960120EC 07FE0000 00000000 *.. .o.....*
```

Figure 9-5 Panel from Security zSecure Audit showing a suspect SVC

The TASID0 program can display the SVC table, but does not perform analysis of the SVC routines.

Preferred practice: If your z/OS system contains SVCs that are 200 - 255 or user SVCs below 200, you should ensure that you understand the provenance of each of them. If you have SVCs that are modified or overlaid using software from non-IBM suppliers, you should understand what each one is doing.

9.3.2 Program Call routines

Program Call (PC) routines were designed to provide access to operating system services in a more efficient manner than SVCs. SVCs gain control following an SVC interrupt and so require a path that flows through the SVC interrupt handler and the scheduler. PC routines are given control in a synchronous manner without this extra impact. However, similar to SVC routines, PC routines may be given control in supervisor state and with the PSW Key Mask (PKM) enabled for system keys. PC routines also have an Authorized Key Mask (AKM) that controls under what circumstances they may be started.

PC routines have an area that is known as the Latent Parameter Area, which can be used as a pointer to parameters for this PC routine.

A PC routine can be used to gain control in a state that is different from the caller's state, and so can be used for unauthorized privilege escalation, if coded badly.

PC routines are used by subsystems and other third-party programs that use advanced z/OS facilities.

PC routines are normally defined from within programs by using assembly language macros. There is no entry in any library that controls how PC routines are defined, and there is not any mechanism to display them in z/OS. The tables that hold their definitions can be interrogated by products such as IBM Security zSecure. Security zSecure can report on potentially bad PC routines.

However, each installation should have a mechanism to determine what PC routines are in use and be sure that they know the provenance of the code in question.

The PC routines requiring the most audit attention are the routines that do the following actions:

- ▶ Can be run from any address space (that is, routines that are in a system Entry Table).
- ▶ Can be called by unauthorized programs (the AKM contains key 8).
- ▶ Run authorized (that is, routines that run in the supervisor state add a system key to the Entry Key Mask (EKM), or have a system key as entry key (EK)).
- ▶ Are in globally readable common storage or libraries (allowing uncontrolled disassembly).
- ▶ Are in private storage (PVT or EPVT), but are not space-switching. Each caller can run a different version of the Program Call. If the Program Call runs authorized and the caller can be unauthorized, a Program Call in the PVT or EPVT might allow any user to gain authorization.
- ▶ Have a Latent Parameter Area with user-pointers pointing to writable common storage.

Figure 9-6 shows a Security zSecure Audit display of a PC routine. It shows a PC routine that is Globally Callable, which means it may be called by anyone. This PC routine may be started by any unauthorized program, and starts running in the supervisor state.

```

Program Call Audit Display
Line 1 of 34
Command ==> Scroll==> HALF
scroll right or select for more info          current settings
Complex System Systemwide PCs Audit concerns Priority
██████████ Systemwide 1046 1046 44
Priority ET-owner Connects PCs Audit concerns
44 ██████████ 12 12
Pr PCnumber A Caller AKM Runs in M K oEKM T Where Program Eye catche
44 00003B01 N FFFF| S 6 o0000 s EPVT ██████████
PCnumber LX-seqno Caller ASID Dorm -LX- Syst LX-owner ASID SFT Description
00003B01 00000000 0000 No 59 Yes ██████████ 005C
PSW Mode Key Type -EX- Syst ET-owner ASID Connects
SUPERVISOR 6 STACKING 1 Yes ██████████ 005C
Authorized Key Mask
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Entry Key Mask / use
OR
Storage area Address Where Key SP Length AM Module
PC routine: 1BC20FAA EPVT 31
Latent Parm: 04D7D150 ESQA 0 245
User parm 1:
User parm 2:
Instruction/Str/SVC scan results
No
Audit concern
In caller's private area, Executes authorized, Caller may be unauthorized,
Globally callable
First 256 bytes of PC routine
0000. 90ECD048 5810F056 58F0F052 0A0D0000 *..}...0..00.....*

```

Figure 9-6 PC call highlighting problems (some parts redacted)

This is a poor definition of a PC routine. If a malicious user uses this routine, he can gain control in the supervisor state and then elevate his authority.

It is often difficult to spot issues with PC routines that have been delivered as part of a third-party program. However, products such as Security zSecure Audit can often spot rogue PC routines and flag them to the user. In such a case, the software supplier should be contacted to fix the exposure.

9.3.3 System exits

System exits are points within the operating system where control is given to customer-supplied code to perform actions that are specific to the customer. These exit points are sometimes used for third-party software functions. If so, the systems programmers must ensure that either the exit point is unused, or that there is a mechanism in place to allow the two exit routines to coexist and function correctly together.

Badly coded exit points can cause integrity and security exposures.

Some system exits can now be coded in System REXX rather than assembly language code.

9.3.4 I/O appendages

I/O appendages are specified in the IEAAPPx member of SYS1.PARMLIB.

I/O appendages are really exit points in the I/O supervisor. As this part of z/OS is central to the way that z/OS works, they are rather sensitive. Only APF-authorized programs may use arbitrary I/O appendages. However, I/O appendages that are specified in this member may be used by unauthorized programs.

I/O appendages receive control in the supervisor state.

9.4 z/OS component protection

There are many components of z/OS that require protection. Many of these are data sets that are used by z/OS. However, there are also commands, consoles, and other elements.

In the past, it might have been acceptable to allow everyone using the operating system to have READ access to the various configuration libraries. This full access is no longer a preferred practice.

Preferred practice: Do not grant READ access to any configuration libraries except to those users with a *defined business need*.

There are some preferred practices for defining UACC values for various z/OS system data sets, which are described in 9.4.5, “System data sets” on page 220.

In all situations, the access that is being granted must be considered in the light of whether it helps someone with *malicious intent* to gain further access, or show what resources are useful to access.

Too often, we do not consider this *malicious intent* question, and those that raise it are considered *paranoid*. However, assuming that there is someone who wants to access your data and then ensuring your risk profile is as low as possible probably serves you better.

9.4.1 UACC

The meaning of UACC is not always clear to everyone. This profile setting was used at a time when not all data sets were thought to need RACF protection, and not all users of systems were thought to need identification. This was the type of system many mainframes had in the late 1970s and early 1980s.

The UACC of a resource or data set is the access level that is granted to anyone who is not named explicitly through group or user access in an access list. It also applies to any user who is *not even known to RACF*.

It is long past the point where it is acceptable for unidentified work to progress through a z/OS system. *All work needs RACF identification*. However, there are still a few cases where work unidentified to RACF can be created.⁸

⁸ The use of user definitions solely in SYS1.UADS may allow work to enter the system without a RACF identity. There is also a possibility for batch work to arrive through RJE and NJE if **BATCHALLRACF** is not set. If a user ID is not assigned to a started task, this too can run without RACF identification.

So the question then arises: why would you want to grant any access to a user that is unidentified to RACF? The answer is that there is no reason to do this. If there is a need for a UACC to be other than NONE, it should be carefully noted.

There are few cases where a UACC other than NONE is needed:

- ▶ One case is for profiles in classes where the UACC is used to mean something different from its normal meaning. Examples of this are the NODES class (which is used for defining the TRUST level for an NJE connection), and the DIGTCERT class, where the UACC field is associated with TRUST levels for the certificate in question.
- ▶ Another case that is relevant is for a resource that is covered by a GLOBAL profile with an access level of READ. In this case, it makes sense for the UACC access level in the profile to match the access level that the GLOBAL profile grants. This ensures a common level of access if the GLOBAL profile fails. However, the GLOBAL access table should not be used as an access control. It should be used as a performance tool. Use of GLOBAL profiles is a security compromise for performance.

For a data set profile, this is the only case where a UACC other than NONE should be used. A UACC of UPDATE should not be used, even for a catalog.

The access levels on system and user catalogs have different meanings than those for other data sets. An access level of UPDATE does not allow uncontrolled changes to the contents as is the case for some other data set. UPDATE allows entries to be appended to the catalog.

It is therefore reasonable to have some profiles for which you might have an access level of UPDATE granted to the ID *. (The ID=* term has no formal name, so we use the term *all-users access* in this book.)

Preferred practice: Avoid any use of UACC that is other than NONE. Use the all-users entry in access lists in preference to UACC. If there is a need for everyone to access a resource at a given level, then use a UACC of NONE and grant the all-users entry the access that is needed, which is READ in nearly all cases.

Preferred practice: Avoid any use of all-users access above NONE for data set profiles. Use all-users access of READ only for data sets that are needed by all users. For example, consider whether your CICS users need access to ISPF libraries.

Preferred practice: Restrict the use of all-users access at UPDATE to profiles for CATALOG protection that really need it. Ensure data set naming standards isolate CATALOG names from other data sets.

In the light of recent attacks on z/OS environments, the authors believe that restricting access to the barest minimum is preferred. So, although some people might advise that placing a UACC of READ or allowing all-users access to the ISPF program libraries is preferred practice, *the authors disagree*.

For example, it is possible that the user population of your z/OS installation is primarily composed of CICS users. Perhaps the TSO users comprise only 5% of your users. If this is your case, then there is no need for those CICS users to have access to the ISPF libraries. If one of those CICS user IDs become compromised, then to allow access to ISPF assists the attacker. If a role-based approach to security is used, then granting READ access only to those libraries for TSO users is straightforward to manage.

For those installations that currently have a UACC of READ on such libraries, the Security zSecure Admin product has a component that is called the Access Monitor, which has mechanisms to determine who needs access to those libraries, and can be used to replace a UACC with specific access list entries.

9.4.2 RACF database

One of the most widely misunderstood areas is the need for privacy for the RACF database. Some have argued that this is not a problem because the password values are encrypted within the database. Although this is true,⁹ there is a host of other information in the RACF database that can be extracted and then analyzed to determine loopholes or weaknesses in local security definitions.

There is other data in the RACF database that also must be kept secret. Read access to the RACF database also makes passwords vulnerable to dictionary attacks.

Preferred practice: Restrict all access to the RACF database to those who have an absolute need to know. Ensure that it has UACC=NONE and does not have an all-users entry in the access list. Apply these rules to all backup copies of the RACF database.

Preferred practice: Ensure all access to the data sets comprising the RACF database is audited. Apply these rules to all backup copies of the RACF database.

9.4.3 Master catalog

Access to the master catalog at READ level might be required by many people. TSO/ISPF users receive an access violation when listing their own data sets *if they do not* have this access because ISPF 3.4 attempts to read the ALIAS pointer in the master catalog that points to the user catalog for the user in question.

So, on this basis, it might be reasonable to grant all-users access to the master catalog at READ level. However, consider that many users of IMS and CICS need no such access.

All other master catalog access should be limited to those users with a defined need, which includes handling the definition and removal of TSO users, and handling of system programmers needs.

Access to catalogs uses multiple interfaces. use of the catalog to *locate* a single entry (such as for a single data set) uses a system interface that is known as LOCATE. This interface does not require READ access to the catalog; it returns details of the volume and device type where the data set is. However, if a catalog is searched for multiple entries, then READ access is required.

Preferred practice: Never have an all-users access level for the master catalog that is UPDATE or higher. If it is feasible to run without any all-users access, then do so.

⁹ In fact, the value that is stored in the password field is an encrypted version of the user ID, with the password being used as the source of the encryption key.

9.4.4 SYS1.PARMLIB

In the past, many customers have chosen to provide READ access to SYS1.PARMLIB. The authors believe that this is no longer acceptable. This library (and each of the members of the parmlib concatenation) should be protected by a specific profile that has a UACC of NONE and does not have an all-users entry in the access list.

If there is a need for some members of this concatenation to be read by all users in the installation, then separate these members in to a separate library that can be defined as part of the concatenation. The libraries that make up the PARMLIB concatenation are defined in the LOADxx member. This member is normally kept in a library on the same volume as the IODF.

Here is an example of this PARMLIB concatenation:

Product XYZ uses a member of SYS1.PARMLIB, and users of this product must see the configuration parameters.

This installation uses SYS1.IPLPARM, which is on the IODF volume. It contains a member that is called LOAD00, which contains the following PARMLIB statements.

```
PARMLIB  USER.PARMLIB                ZDSYS1
PARMLIB  SYS1.PARMLIB                 ZDRES1
```

A new library definition is added to the list. This library, USER.XYZ.PARMLIB, may be used to hold parameters for the XYZ product. This library now may be given a different access list to the other libraries in the parmlib concatenation. Here are the relevant statements:

```
PARMLIB  USER.PARMLIB                ZDSYS1
PARMLIB  USER.XYZ.PARMLIB           ZDRES1
PARMLIB  SYS1.PARMLIB                 ZDRES1
```

Thus, a set of users can be allowed to READ the library USER.XYZ.PARMLIB while you deny that same set of users any access to the other two libraries.

In this example, giving that set of users UPDATE access to USER.XYZ.PARMLIB can cause potential security issues because the library is higher than SYS1.PARMLIB in the set of libraries. So, USER.XYZ.PARMLIB can be made to contain a parmlib member, which overrides a corresponding member in SYS1.PARMLIB. This configuration then can be used (for example) to add a library to the APF list.

Strict controls should be placed on all members of the PARMLIB concatenation, and on the SYSn.IPLPARM library that defines the set.

Security zSecure Alert can be used to raise alerts whenever a library is changed. If such an alert is received for a change outside of normal change management procedures, then this might indicate an attempted security breach.

9.4.5 System data sets

This section contains some guidelines for defining UACC values for system data sets.

Table 9-1 lists the UACC values that we recommend be assigned to many of the system data sets. Your security policy might require a UACC of NONE for some data sets. For example, you can specify UACC(NONE) for macro libraries if you give READ access to programmers who need to assemble or compile programs that use those libraries. For a more detailed description of protecting system data sets, see “Security for system data sets” in the *z/OS Security Server RACF Security Administrator’s Guide*, SA22-7683. You can find the latest version of this guide in the following online Knowledge Center:

<http://pic.dhe.ibm.com/infocenter/zos/v1r13/topic/com.ibm.zos.r13.icha700/sds1c.htm#sds1c>

Table 9-1 UACC values for system data sets

Data set	UACC	Comments
APF libraries	NONE	Authorized program facility libraries.
Checkpoint data sets	NONE	
Distribution library data sets	NONE	
ISPF panel libraries	READ	Panel definitions, skeletons, CLISTs, and so on. Specify UACC(NONE) if access must be restricted.
JES2 offload data sets	NONE	
Load libraries	READ	
Master catalog	READ	
Page data sets	NONE	Includes PLPA, common, and local page data sets. For more information, see <i>z/OS MVS Initialization and Tuning Guide</i> , SA23-1379-02
PSF secure font data sets	NONE	
PSF secure overlay data sets	NONE	
PSF secure page segment data sets	NONE	
RMF data sets	NONE	VSAM data sets.
Security definitions data sets	NONE	
SMP data sets	NONE	
Swap data sets	NONE	
SYS1.AMACLIB	READ	
SYS1.AMODGEN	READ	
SYS1.ASAMPLIB	READ	
SYS1.BROADCAST	READ	
SYS1.CMDLIB	READ	
SYS1.DAE	NONE	

Data set	UACC	Comments
SYS1.DUMPxx	NONE	For more information, see <i>z/OS MVS Initialization and Tuning Guide</i> , SA23-1379-02.
SYS1.HELP	READ	TSO online help.
SYS1.IMAGELIB	NONE	
SYS1.JESPARM	NONE	
SYS1.JES3LIB	READ	
SYS1.LINKLIB	READ	
SYS1.LOGREC	NONE	
SYS1.LPALIB	READ	UACC can be NONE or READ depending on your installation's security policy.
SYS1.MACLIB	READ	
SYS1.MANx	NONE	SMF data sets. For more information, see <i>z/OS MVS Initialization and Tuning Guide</i> , SA23-1379-02.
SYS1.MIGLIB	READ	
SYS1.MODGEN	READ	
SYS1.NUCLEUS	READ	
SYS1.OVERLIB	READ	
SYS1.PARMLIB	READ	UACC should be NONE if any members contain passwords, or other sensitive information, such as the ACBPW password in the TSOKEYxx member.
SYS1.PROCLIB	READ	
SYS1.RACF	NONE	Includes data sets that comprise the active and backup RACF databases, split data sets created with the IRRUT400 utility, and archival copies. Your installation might use other data set names.
SYS1.SAMPLIB	READ	
SYS1.SAXREXEC	READ	System REXX library, or any libraries that are defined in the REXXLIB concatenation. UACC can be NONE or READ depending on your installation's security policy.
SYS1.STGINDEX	NONE	
SYS1.SVCLIB	NONE	
SYS1.TELCMLIB	READ	
SYS1.UADS	NONE	
SYS1.VTOCIX...	NONE	
SYS1.VVDS...	NONE	
SYS1.VTAMLIB	READ	
SYS1.VTAMLST	NONE	
Trace data sets	NONE	

Data set	UACC	Comments
User catalogs	UPDATE	
User dump data sets	NONE	

Consider creating a generic profile to protect system data sets by using the following statement:

```
ADDSD 'SYS1.*' UACC(NONE) SECLABEL(SYSHIGH)
```

Specifying a UACC of NONE for the SYS1.* profile protects any system data sets that are added to the system by new products. If new system data sets need a UACC higher than NONE or a SECLABEL of SYSLOW, you can create more specific profiles for them.

You also should create specific profiles for particular data sets (or groups of data sets, such as SYS1.DUMPxx data sets) using the information in Table 9-1 on page 221.

For any data set that is listed with a UACC of READ or higher, you should consider creating an entry in the *global access checking table*. For more information, see “Global access checking table”.

For system data sets that are listed in the table with a UACC higher than NONE, you might prefer to specify UACC(NONE) and create an access list entry of ID(*) ACCESS(access-authority). This entry prevents restricted users and users who are not defined to RACF from accessing the data sets.

Restricted users enter the system with a user ID that is defined with the **RESTRICTED** attribute, and might be shared by many users. Restricted users are prevented from gaining access to protected resources through the global access checking table, UACC, or the ID(*) entry on the access list. User IDs that are defined with the **RESTRICTED** attribute must be authorized with sufficient authority on the access list of any protected resource that they must access. Therefore, to allow restricted users to access any data set that is listed with a UACC of READ or higher in Table 9-1 on page 221, each user ID with the **RESTRICTED** attribute must be authorized at the level of access that is indicated by the UACC value.

Global access checking table

You can use global access checking to improve the performance of RACF authorization checking for selected resources. Global access checking should be used for public resources that are accessed frequently. For example, an entry in the global access checking table can allow all users on the system to have READ access to the SYS1.HELP data set.

The global access checking table is maintained in storage and is checked early in the RACF authorization checking sequence. If an entry in the global access checking table allows the requested access to a resource, RACF performs no further authorization checking. This can avoid I/O to the RACF database to retrieve a resource profile, and can result in substantial performance improvements.

Global access checking is used for authorization processing that is started by the RACROUTE REQUEST=AUTH macro. It is not used for authorization processing that is invoked by the RACROUTE REQUEST=FASTAUTH macro.

9.4.6 System page data sets

System page data sets have three types:

- ▶ PLPA pages data set
- ▶ COMMON page data set
- ▶ Local page data set.

The PLPA page data set is the first one that is named in the **PAGE** parameter that is specified in the IEASYSxx member of SYS1.PARMLIB. It is used to contain pages representing the Pageable Link Pack Area.

The COMMON page data set is the second named page data set, and is used for data that is in the common area of storage (that is, common to all address spaces), but that is not in the PLPA.

All other page data sets are for local address spaces. They may contain almost *any* data that is in *any* address spaces. This data includes passwords that have been entered, private keys, and any other data that might be processed by the application running in the address space.

Preferred practice: Page data sets should be protected from being viewed. They should have audit settings that show who accessed them, whether for READ access or anything else.

9.4.7 System dump data sets

System dump data sets may contain data from anywhere in storage, including all private address spaces. The data is dumped there (according to various policies, held in members of SYS1.PARMLIB, and some RACF controls) typically when error situations are encountered.

Steps should be taken to ensure that the system dumps have access controls that are placed on them.

Preferred practice: System dump data sets should be protected from being viewed from almost all users. Those staff requiring access for system dump analysis should be the only users accessing these data sets. The data sets should have audit settings that show who accessed them, whether for READ access or anything else.

System dump data sets are often archived. If this is done, then those archives also should be protected in a manner similar to the original system dump data sets.

User dump controls

Two RACF profiles can be used to protect data within address spaces from being written to user dump data sets (SYSUDUMP, SYSMDUMP, and SYSABEND).

Both of these profiles are in the FACILITY class:

- ▶ IEAABD.DMPAUTH

If an address space does not have READ access to this resource, then the user dump is not taken if the address space contains RACF controlled programs.

If the address space contains programs that have been fetched from a library to which the user has only EXECUTE access, then UPDATE access to this resource is needed, or the user dump is not taken.

► IEAABD.DMPAKEY

If the address space does not have READ access to this resource and the program is running in a system key with a value less than 8, the user dump is not taken.

9.4.8 SYS1.STGINDEX

The SYS1.STGINDEX data set (or an alternative name that is assigned through the **VIODSN** parameter in IEASYSxx) holds journaling information about VIO data sets that are used across multiple job steps, or across IPLs.

There is no reason to grant any access to this data set to any one who is not concerned with the running of the z/OS operating system.

Preferred practice: The SYS1.STGINDEX data set (or the alternative name that is assigned by the **VIODSN** parameter in IEASYSxx) should be protected from being viewed from all users. It should have audit settings that show who accessed it, whether for READ access or anything else.

9.4.9 SYS1.LOGREC

The SYS1.LOGREC data set (or the alternative name that is assigned by the **LOGREC** parameter in the IEASYSxx member of SYS1.PARMLIB) contains analysis information about hardware and software errors that are encountered during system operations. We shall refer to this information as LOGREC data. LOGREC data might contain user data, which is of unknown data classification.

There is no reason to grant any access to this data set to any one who is not concerned with the running of the z/OS operating system.

Preferred practice: The data set SYS1.LOGREC (or the alternative name that is assigned by the **LOGREC** parameter in IEASYSxx) should be protected from being viewed from almost all users. Only those administrators that are associated with problem analysis should have READ access. It should have audit settings that show who accessed it, whether for READ access or anything else.

Preferred practice: The **LOGREC** function also can be managed by using the high-performance **LOGSTREAM** function. If this option is chosen, then all data sets that contain data from **LOGREC** should be similarly protected from all users.

Preferred practice: If **LOGREC** data is archived, then this data should be protected the same way that the original SYS1.LOGREC data set is protected.

9.4.10 SMF data sets

SMF data sets (or the SMF logstream) contain logging information from z/OS and from many of its major subsystems. It might contain data from third-party software. It might contain data that the installation has specified.

It is a high-speed, high-capacity logging system, and should be highly secure.

The interface for writing records to SMF is protected with APF. The system macro that must be started (SMFWTR or one of its derivatives) requires that the caller be APF-authorized. Thus, the installation should have complete control over the contents of the SMF data.

There also is a UNIX System Services callable service called BPX1SMF (or BPX4SMF if operating in 64-bit mode). use of this service requires either APF authorization or access to a SAF resource called BPX.SMF in the Facility class. These services can be used to write arbitrary SMF data.

A member of SYS1.PARMLIB called SMFPRMxx is used to configure options of SMF. Each SMF record has a record type and potentially a subtype. The set of SMF records that are associated with z/OS is in *z/OS MVS System Management Facilities (SMF)*, SA22-7630-23.

The SMFPRMxx member may specify records that can be suppressed by type or by subtype. It can specify that in some environments¹⁰ different sets of records can be collected.

Some enterprises suppress SMF records to save on the CPU time that is associated with writing them, and to reduce the amount of data to be managed. This is a clear example of the conflict between cost and security. The authors think this judgment must be made by using a risk-base approach.

At a bare minimum, record the following records for security purposes: 7, 14, 15, 30, 42, 80, 81, 83, 90, 92, 102, 118, and 119. A more conservative list also includes 6, 17, 18, 20, 26, and 60 - 68. Your local standards probably have mandates about these settings.

It should be fairly clear that allowing access to SMF records that contain details of auditing successes and auditing failures is useful to any attacker who is trying to negotiate his way around security controls. So, the access to SMF records (particularly those related to security auditing) needs careful attention.

Preferred practice: Only those people that are concerned with auditing and those that are concerned with the management of SMF records should have access to them. Restrict all other access, both to the live SMF data sets and also to all archives of SMF material.

9.4.11 System linklist

The system linklist is a set of libraries that can be accessed unconditionally by any address space to load a load module or program object for running or examination.

However, this access is allowed only if the library is accessed *through the linklist*. If one of the libraries in that set is accessed simply by using a **STEPLIB** statement or some other DDname, then the access is controlled by using the access list corresponding to the data set.

The system linklist is held open by z/OS. If you use your own DDname and access the library, then you must perform an *open operation*. The open operation is performed by using SVC 18 or SVC 21. Each of these SVCs issues a call to SAF and RACF to determine authority to open the library.

If access is given to read the libraries in the system linklist, then this access allows the copying of those libraries, or the members within them. This might be unwanted.

¹⁰ The term that is used by SMF is *subsystem*, but this is not exactly aligned with the subsystem definition that is used in MVS for a formal subsystem.

Some libraries might be required at run time for some programming languages and these libraries also might be needed at *bind time* during application development. If so, then READ access to the library in the linklist might need to be given to the application development staff.

The libraries that are supplied as components of z/OS Language Environment do not fall into this category. Libraries for Language Environment are designated either as runtime libraries (good candidates for the system linklist) or application development (AD) libraries, but not both.

Preferred practice: Restrict READ access to the system linklist libraries to those who have a defined business need to read them.

Preferred practice: Give UPDATE access to system linklist libraries only to those personnel who are trusted with that responsibility.

9.4.12 System LPALIST

The system LPALIST is similar to the system linklist, except that the load modules and program objects in question are loaded into common memory, which can be read from any address space.

The LPA (or Pageable LPA) is in 24-bit memory and 31-bit memory. It also is in the PLPA page data set. This page data set is built or rebuilt at IPL time if the CLPA parameter is used.

The same access restrictions apply to the system LPALIST that applies to the system linklist.

Preferred practice: Restrict READ access to the system LPALIST libraries to those personnel who have a defined business need to read them.

Preferred practice: Give UPDATE access to system LPALIST libraries only to those personnel who are trusted with that responsibility.

9.4.13 APF-authorized libraries

Access to APF-authorized libraries can be confusing. Of utmost importance is recognizing that any user who has UPDATE access to an APF-authorized library can run arbitrary code at the operating system level and therefore can bypass security controls.

For example, the NETBILL is the user ID of Bill Jones, who is a network and VTAM systems administrator. NETBILL has UPDATE access to the library USER.VTAMLIB, which is normally used to contain UNIX System Services tables for 3270 screens.

However, this library must be concatenated with the VTAM program library, SYS1.VTAMLIB, in the VTAM started procedure. If USER.VTAMLIB is not APF-authorized, then the VTAM address space cannot start because USER.VTAMLIB is part of a VTAMLIB concatenation of libraries. If any member of the concatenation is not authorized, then the entire concatenation is treated as not authorized for this job step.

So, the STEPLIB concatenation would be as follows:

```
//VTAMLIB DD DISP=SHR,DSN=USER.VTAMLIB
//          DISP=SHR,DSN=SYS1.VTAMLIB
```

Bill Jones can update USER.VTAMLIB with the UNIX System Services tables, which is part of his job. However, if he is so inclined, Bill also could write a program and place that program into USER.VTAMLIB and then run the program from that STEPLIB by using simple JCL. If he marks the program with the code AC(1) at the binder stage, then it commences running with APF authorization. Such a program can then alter its system key to zero and then modify the system control blocks that are associated with security controls.

This example shows that although Bill Jones was given the UPDATE access only to manage UNIX System Services tables, he might subvert that use to compromise system security. This situation might be acceptable at your installation. However, you must recognize the effective authority that has been given to Bill Jones.

Preferred practice: Give UPDATE access to APF-authorized libraries only to those personnel who are trusted with that responsibility.

Preferred practice: Monitor the contents of all APF-authorized libraries and use an alerting function to ensure that you are quickly notified of any changes. Changes should always be tied back to your formal change management processes.

Whether READ access should be granted to APF-authorized libraries depends on how the programs in the library question behave. If those programs are written to check the authority of the caller to perform some specific operation, then it might be reasonable to grant READ access to that person or job role. However, some APF-authorized programs are not designed this way. If so, then care should be used to ensure that only those people with a need to access the program library get READ access.

9.4.14 System log

The system log contains a great deal of information that can be useful to someone attacking a z/OS system. Access to this log should be restricted only to those who need access. Broadly speaking, this does not include application development staff or any users. So, these data sets should have a UACC of NONE and no ID(*) entry.

Whether the system log is accessed by using SDSF or some other mechanism, access to the data should be restricted.

9.5 The IBM Health Checker for z/OS

In z/OS V1.7, IBM introduced the IBM Health Checker for z/OS as a part of z/OS. The Health Checker consists of two parts:

- ▶ The health checker *infrastructure*, which manages the initiation, running, and output of health checks
- ▶ The *health checks*

Health checks have been written by IBM, by vendors, and by clients. IBM ships approximately 140 health checks.

Enabling the IBM Health Checker for z/OS involves defining allocating, and formatting a data set and defining a started procedure. Starting with z/OS V2.1, the IBM Health Checker for z/OS starts automatically.

RACF ships with many health checks whose output are useful in understanding the overall level of security and adherence to generally accepted security practices. Here are those health checks:

▶ RACF_SENSITIVE_RESOURCES

This check examines selected key system data sets (the PARMLIB concatenation, the data sets that make up the RACF database, System REXX data sets, system link list data sets, and ICSF data sets), and raises an exception if the following situations occur:

- There is no RACF profile covering the data set and PROTECTALL(FAIL) is not in effect.
- The profile covering the data set has an overly permissive universal access (UACC).
- The profile covering the data set is in WARNING mode.

▶ RACF_<class-name>_ACTIVE

This set of checks examines the class that is named class-name and raises an exception if the class is INACTIVE. The classes that are checked are CSFSERV, CSFKEYS, FACILITY, OPERCMDS TAPEVOL, TEMPDSN, TSOAUTH, and UNIXPRIV.

One or more of these classes might be inactive depending on your system configuration. For example, if you are not using ICSF, then you do not need to have the CSFSERV or CSFKEYS class active and you can deactivate the health check.

▶ RACF_IBMUSER_REVOKED

This check ensures that the user ID IBMUSER has been revoked. IBMUSER is a well-known user ID and should be used only early in the configuration of RACF. One of the first steps that is done during the installation is to revoke IBMUSER.

▶ RACF_CERTIFICATE_EXPIRATION

This check locates the digital certificates that are contained within the RACF database and identifies which are expired or are about to expire. The *warning* interval can be specified by the installation, and defaults to 30. Certificates that are marked as *Trusted* are considered exceptions if they are expired or fall within the warning interval.

This check is useful in preventing an application outage because of a certificate expiration. Only certificates that are stored in the RACF database are examined. Certificates in other data stores, such as z/OS UNIX files, are not examined.

▶ RACF_GRS_RNL

This check validates that the serialization requests that are made by RACF have not been altered by the installation defining a resource name list (RNL), which affects the RACF serialization. Altering the RACF serialization can lead to severe problems, including the corruption of the RACF database.

▶ RACF_ICHAUTAB_NONLPA

This check flags an exception if it finds a RACF Authorized Caller Table (ICHAUTAB). The use of these tables is not recommended by IBM.

▶ RACF_UNIX_ID

This check examines your z/OS UNIX identity setup to see whether you are assigning unique z/OS UNIX identities, either by explicitly defining all of the contents of the user and group OMVS segments, or using the automated UID, GID, and OMVS segment generation facilities that are offered by BPX.UNIQUE.USER.

The check looks for the existence of BPX.DEFAULT.USER in the FACILITY class, SHARED.IDS, BPX.NEXT.USER, and BPX.UNIQUE.USER in the UNIXPRIV class, verifies that the UNIXPRIV class is active, the application identity mapping (AIM) stage of the RACF database is 3, and that BPX.NEXT.USER has APPLDATA.

9.5.1 Managing your health checks

You can manage health checks by using z/OS operator comments. You can examine the output of a health check by using the HZSPRNT utility. However, the easiest way to manage health checks and to look at the output of checks is by using the System Display and Search Facility (SDSF) or equivalent product. To use the SDSF, complete the following steps:

1. From the SDSF PRIMARY OPTION MENU, select CK (Health checker), as shown in Figure 9-7.

```
Display Filter View Print Options Search Help
-----
HQP7790 ----- SDSF PRIMARY OPTION MENU -----
DA   Active users          INIT  Initiators
I    Input queue          PR    Printers
O    Output queue         PUN   Punches
H    Held output queue    RDR   Readers
ST   Status of jobs      LINE  Lines
                                NODE  Nodes
LOG  System log          SO    Spool offload
SR   System requests     SP    Spool volumes
MAS  Members in the MAS  NS    Network servers
JC   Job classes         NC    Network connections
SE   Scheduling environments
RES  WLM resources      RM    Resource monitor
ENC  Enclaves           CK    Health checker
PS   Processes          ULOG  User session log

END   Exit SDSF
COMMAND INPUT ==>  ck                                SCROLL ==> HALF
```

Figure 9-7 Start the health checker in SDSF

2. SDSF shows a list of all of the health checks. You can use the UP (PF7) and DOWN (PF8) keys to scroll through this list. Scroll down until you see the RACF health checks. If you type S (select) next to any check (as shown in Figure 9-8), you can see the output of the check.

Display Filter View Print Options Search Help				

SDSF HEALTH CHECKER DISPLAY RACFR21				LINE 56-71 (141)
NP	NAME	CheckOwner	State	Status
	RACF_AIM_STAGE	IBMRACF	ACTIVE(ENABLED)	SUCCESS
	RACF_CERTIFICATE_EXPIRATION	IBMRACF	ACTIVE(ENABLED)	SUCCESS
	RACF_CSFKEYS_ACTIVE	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_CSFSEV_ACTIVE	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_FACILITY_ACTIVE	IBMRACF	ACTIVE(ENABLED)	SUCCESS
	RACF_GRS_RNL	IBMRACF	ACTIVE(DISABLED)	ENV N/
	RACF_IBMUSER_REVOKED	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_ICHAUTAB_NONLPA	IBMRACF	ACTIVE(ENABLED)	SUCCESS
	RACF_OPERCMD_ACTIVE	IBMRACF	ACTIVE(ENABLED)	SUCCESS
s	RACF_SENSITIVE_RESOURCES	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_TAPEVOL_ACTIVE	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_TEMPDSN_ACTIVE	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_TSOAUTH_ACTIVE	IBMRACF	ACTIVE(ENABLED)	SUCCESS
	RACF_UNIX_ID	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RACF_UNIXPRIV_ACTIVE	IBMRACF	ACTIVE(ENABLED)	EXCEPT
	RCF_PCCA_ABOVE_16M	IBMRACF	ACTIVE(ENABLED)	SUCCESS
COMMAND INPUT ==>			SCROLL ==>	HALF

Figure 9-8 Select your RACF health checks in SDSF

3. The check output is then displayed, as shown in Figure 9-9.

```

Display Filter View Print Options Search Help
-----
SDSF OUTPUT DISPLAY RACF_SENSITIVE_RESOURCES      LINE 0      COLUMNS 02- 81
COMMAND INPUT ==>                                SCROLL ==> HALF
*****
***** TOP OF DATA
*****
CHECK(IBMRA CF,RACF_SENSITIVE_RESOURCES)
SYSPLEX: LOCAL      SYSTEM: RACFR21
START TIME: 08/22/2014 15:56:50.685055
CHECK DATE: 20120106 CHECK SEVERITY: HIGH

                          APF Dataset Report

S Data Set Name                      Vol      UACC Warn ID*  User
-----
ASM.SASMMOD1                         ZDR21  Read No  ****
V ATC.V2R1M4.AUTHLIB                 DRVPSL
V CBC.SCBCCMP                         ZDR21
CBC.SCCNCMP                          ZDR21  Read No  ****
CBC.SCLBDLL                          ZDR21  Read No  ****
CBC.SCLBDLL2                         ZDR21  Read No  ****
E CEE.SCEERUN                        ZDR21
. . .
. . .
. . .
* High Severity Exception *

IRRH204E The RACF_SENSITIVE_RESOURCES check has found one or
more potential errors in the security controls on this system.

Explanation: The RACF security configuration check has found one or
more potential errors with the system protection mechanisms.

System Action: The check continues processing. There is no effect on
the system.

```

Figure 9-9 RACF_SENSITIVE_RESOURCES health check output

9.6 RACF settings

The configuration of RACF is something that many security standards for z/OS address, which is good. However, blind adherence to those standards does not ensure that your systems are secure.

This section describes some matters that have real security impact. This is *not* an exhaustive analysis of all the RACF settings that are possible, and it is not an attempt to explain what each setting does (that's the role of the RACF manuals). We add details here for clarification purposes only.

9.6.1 INITSTATS

This setting ensures that statistics about who has authenticated to your z/OS are recorded. This is a basic control, but we have found systems where this is not set on.

Preferred practice: Ensure that INITSTATS is on.

9.6.2 SAUDIT

The SAUDIT setting ensures that RACF commands that are run by users with the system-Special or group-Special attributes are logged.

Preferred practice: Ensure that SAUDIT is on.

If a command is authorized for use without checking the SPECIAL attribute, such as when the user with the SPECIAL attribute owns the resource, then logging is not performed because SAUDIT is specified. Instead, logging follows the normal logging rules for the resource.

Preferred practice: Ensure that users with the SPECIAL attribute do not own resources or profiles.

9.6.3 CMDVIOL

The CMDVIOL setting ensures that all command violations are logged. This means that any command that fails because of a security check failure is logged.

Preferred practice: Ensure that CMDVIOL is on.

9.6.4 OPERAUDIT

The OPERAUDIT setting ensures that all accesses that are granted because of the OPERATIONS attribute are logged.

Preferred practice: Ensure that OPERAUDIT is on.

9.6.5 AUDIT classes

The AUDIT options specify which classes or resource are to be audited. This means that any changes made to the resource profiles in the class that is specified are logged whether they are made through a RACF command or through lower-level system interfaces.

For the USER class, this includes password and password phrase changes.

Preferred practice: Ensure that all active classes are audited.

It is possible to specify SETROPTS AUDIT(*). This is equivalent to setting AUDIT on for all classes that are defined. This should be regarded as a shortcut command only. If a new class is defined, then the AUDIT(*) setting does not apply to that class. It is the security administrator's responsibility to ensure that the AUDIT setting is on for each class.

9.6.6 GLOBAL classes

Few things cause as much confusion in RACF as GLOBAL classes. The first thing to note is that this is not really a security setting. If GLOBAL is used, then it is used as a performance setting. In many cases, the use of GLOBAL provides far too much access to resources.

We, the authors, have stated above that we do not favor the use of UACC or the all-users setting of ID(*) in access lists. If you use neither of these settings in any profile, then GLOBAL is not applicable to your environment. Use of GLOBAL grants only access that is denied.

The use of role-based access controls (RBACs) should define which resources must be accessed by which roles and at which levels. If you have a mixed group of user types, such as CICS users and TSO users, then few resources ever must be accessed by all users.

However, it might be that you have a system that has only TSO users (or users of some others type). In this case, it might be reasonable that you want to grant all users (which is the same as all TSO users) READ access to ISPF libraries, SYS1.HELP, SYS1.BROADCAST, and so on. In addition, you want to take advantage of the performance benefits of the GLOBAL access table (GAT).

GLOBAL can be defined to grant access to specific resources at a specified access level without any further checking. This means that no other profile is used and no logging takes place.

GLOBAL profiles never *deny* access. So, it should be clear now that this is *not* a security feature.

Preferred practice: Examine your GLOBAL settings carefully to ensure that they do not compromise security requirements. Ensure that you have good grounds for the use of GLOBAL, and that changes to the GAT are monitored and justified. Review GLOBAL profiles at regular intervals for continued applicability.

Preferred practice: Ensure that the access granted by the GAT is reflected in the resource profiles protecting the resources. It is possible that the GAT can be deactivated. This may have a performance impact, but should not have a security impact.

9.6.7 Enhanced Generic Naming

Enhanced Generic Naming (EGN) often is a confusing area. This option is relevant for the DATASET class only. EGN effectively is set ON for all other classes (except USER, GROUP, and the digital certificate classes).

EGN allows the use of a wider range of generic structures in the generic profiles that protect data set names. In general, this is a good thing, and often helps with some separation of security for data sets (depending on your naming standards).

If you have never had EGN turned on, then if you set EGN on, there is no change of any access that is granted to any existing data sets. However, the existing profiles might be displayed differently by the commands. This may affect any routines that respond to the data that is displayed by the RACF commands for data sets.

Preferred practice: Set EGN on if possible. Beware any routines that examine the output of RACF commands.

9.6.8 BATCHALLRACF

The BATCHALLRACF setting ensures that if a job enters your system without a RACF user who is associated with it, it fails.

Preferred practice: Ensure that BATCHALLRACF is set on.

9.6.9 PROTECTALL

The PROTECTALL setting applies to data sets only.

PROTECTALL ensures that all data sets are protected by a profile. The profile may be generic or discrete,¹¹ but it will exist. With this setting on, you cannot create a data set that does not have a profile to protect it.

As a preferred practice, *all* your data sets should be protected.

Preferred practice: Ensure PROTECTALL (FAILURES) is set.

9.6.10 Tape data set protection

Tape data sets should be protected. In many installations, tape is used as a backup. It may be used as part of the storage hierarchy of DFSMSHsm.

In order for RACF to provide adequate tape protection, the TAPEDSN option must be set on, and the TAPEVOL class must be active.¹²

Tape data is sometimes seen as not as important as front-line DASD data. However, it must be secured in the same manner as other data. Consider the following matters regarding this topic:

- ▶ Allowing wide access to all backups, as many users might need to restore data.

This situation potentially allows one user access to a copy of another user's data. Is this what you want? Is this part of your security design?

- ▶ Allowing BLP to be secured by using JES only.

JES allows BLP to be used based on the job class. In z/OS V2.1, the job class can be secured. However, this action is not possible for previous releases of z/OS. BLP should be secured by using the FACILITY class profile called ICHBLP. (This check is effective only if the TAPEVOL class is active.)

¹¹ use of a discrete profile requires that the data set is security-indicated in the DASD VTOC for non-VSAM data sets, and indicated in the catalog for VSAM data sets.

¹² If you run a tape management product for tape data protection, then ensure that its protection is enabled all the time if TAPEDSN or TAPEVOL are not used.

Preferred practice: Protect tape data by using the same set of data classification as disk data. Ensure that BLP is secured. Use tape encryption if tapes are moved away from their working environment.

9.6.11 Erase on scratch

The impact that is associated with erase on scratch often is given as the reason why erase of scratch is not used for all data sets. However, it is reasonable to use this option with the smaller subset of known sensitive data sets. Here are examples of such data sets:

- ▶ RACF database
- ▶ System dump data sets
- ▶ SMF data sets
- ▶ SPOOL data sets
- ▶ ICSF keystores

Preferred practice: Make strategic use of the erase on scratch capability for operating system sensitive data sets, and other data sets that are known to contain sensitive, unencrypted information.

9.6.12 Inactive user ID revocation

If a user ID has not been used for a long time, then the user ID might be a target for someone who can use a social attack to gain access to the password.

Inactive user IDs should be made unusable by setting the value on the INACTIVE interval on SETROPTS. The value that is used is installation- or standards-specific.

Preferred practice: Ensure that a value is set on the INACTIVE interval.

9.6.13 Password management

The management of passwords is an important aspect of any security administrator and must be approached with some care if common weaknesses are to be avoided.

There have been numerous articles written about password management, and about mechanisms for breaking passwords. The encryption algorithm that is used in managing passwords in RACF is single DES, and some experts have maintained passwords would be stronger in RACF if triple DES or AES were used instead.

It should also be obvious to anyone that if a user ID and password are exposed, then an identity is effectively exposed. So, if someone is using the user ID to access the resources to which the user ID has normal access, *no alert is raised*.

Consider the following items:

- ▶ If passwords are easy to guess, then no amount of encryption in the password database will help.
- ▶ Dictionary attacks against password databases are the fastest way to expose passwords, and the RACF database is no exception.
- ▶ If your staff does not take steps to protect their passwords, then they are likely to be exposed.

- ▶ If your staff uses the same password for multiple applications, then the passwords are as strong as the weakest level of protection of those systems.
- ▶ People take easy ways out, and are not good at password management.
- ▶ A password that is more complex and chosen from a wider number of characters or is simply longer is more difficult it is to crack with a dictionary attack.

It should be clear here that using *dual-case passwords* and *password phrases* is a *good thing*. However, these capabilities rarely are used.

Preferred practice: Enable dual-case passwords and encourage their use by using the RACF command **SETROPTS PASSWORD(MIXEDCASE)**.

Preferred practice: Use password phrases wherever possible by using the **PHRASE** parameter with each **ADDUSER** command that is used to create a user ID.

In z/OS, it is always necessary to have a password that is defined for a user ID, even if a password phrase is also in use.¹³ It is possible to have a process for assigning random passwords to such user IDs.

Preferred practice: Have a system for randomizing passwords that are associated with user IDs for which password phrases are used. If a user always signs on to systems that have full phrase support, use the process to effectively eliminate the password.

Few standards for RACF and z/OS mandate these capabilities, possibly because these standards are “behind the curve” or they anticipate difficulties with older software products that do not support these newer features.

Here are the other settings that can be made with the **PASSWORD** parameter of the **SETROPTS** command:

- ▶ **HISTORY**
This setting records the number of previous passwords that are held to ensure that they are not reused.
- ▶ **INTERVAL**
This setting sets the maximum number of days between password or password phrase changes.
- ▶ **MINCHANGE**
This setting sets the minimum number of days between password or password phrase changes.
- ▶ **REVOKE**
This setting sets the number of unsuccessful attempts for passwords and password phrases before the user ID is revoked.
- ▶ **RULEn**
This setting can specify up to eight rules to be applied to define an acceptable password.
- ▶ **WARNING**
This setting specifies the number of days before a password or password phrase expires that a warning message is given.

¹³ This restriction does not apply to RACF on z/VM.

All of the above settings should be used. However, as the values for these settings usually are supplied by standards, use the settings that are supplied by your chosen standard.

It probably is beneficial to be educated about the management and protection of passwords. However, this is a subject that is best handled by the wider enterprise security in your organization. It pertains to much more than just z/OS and RACF.

9.6.14 GENERICOWNER

The SETROPTS GENERICOWNER setting ensures that a user with CLAUTH to a particular class cannot create a more specific profile to gain access to a resource to which he was denied control. Every installation should have this set.

Preferred practice: Set GENERICOWNER on.

9.6.15 Multi-level security options

Few installations use multi-level security (MLS). If there is a need to implement MLS, then you should study the subject carefully and ensure that you test your environment with care.

MLS and mandatory access controls can be used for installations that have well-defined data classifications, and can be configured to prevent the de-classification of that data.

The following SETROPTS settings are related to MLS: COMPATMODE, MLACTIVE, MLS, MLSTABLE, SECLABELAUDIT, SECLABELCONTROL, SECLEVELAUDIT, MLFSOBJ, MLIPCOBJ, MLNAMES, and SECLBYSYSTEM.

For more information about MLS, see *z/OS Planning for MultiLevel Security and the Common Criteria*, GA22-7509-10. The details for the implementation of MLS are beyond the scope of this book.

9.7 JCL procedures

JCL procedures, also known as cataloged procedures, do not have a separate class protecting them. However, started tasks use cataloged procedures, so the security of procedure libraries is important.

If procedure libraries are used privately by using the **JCLLIB** JCL statement, then READ access to the library that is specified is checked during the expansion of the JCL. If no access is granted, then the JCL fails. However, if the procedure is in a library that is associated with the JES, then no such access check is performed.

The organization of procedure libraries is important for security and needs careful attention, as illustrated by the following example.

Bill, who is the sysprog who works at Simple Banking Corporation, is asked to supply a new procedure library for his developers. He looks at the current set of libraries and decides to create a library that is called SYST.PROCLIB. This library will be used often, so he decides to place the new library at the top of the set of procedure libraries to make searching for the new procedures more efficient.

The set of procedure libraries for the installation now looks like the following lines:

```
//PROC00 DD DISP=SHR,DSN=SYST.PROCLIB
// DD DISP=SHR,DSN=STC.PROCLIB
// DD DISP=SHR,DSN=SYS1.PROCLIB
```

The procedure for ICSF is called CSF, and it is in STC.PROCLIB. The configuration of ICSF includes an exit that prevents most users from calling a User Defined Extension (UDX), which is enabled in the Crypto Express devices at Simple Banking Corporation.

Jeff works in the applications development department and has been trying to get access to the UDX function to try to crack some of the PIN values in a copy of a database that was copied from production to development for testing purposes. He needs access to the UDX. However, the ICSF exit prevents him getting that access.

Jeff makes a copy of the configuration parameters for ICSF and then builds a procedure for ICSF with the changed configuration parameters specified in it. He stores this procedure in SYST.PROCLIB under the name CSF.

At the next IPL, the changed CSF procedure in SYST.PROCLIB is used and ICSF starts without the exit active. This enables Jeff to use the ICSF services to crack the PINs in the copied customer database.

This example show that there must be separation of control so that overriding a procedure name is not possible.

Preferred practice: Keep started task procedures in a separate set from other cataloged procedures. Ensure that the order of libraries does not provide opportunities for Trojan horse JCL procedures.

9.8 UNIX System Services

In the experience of the authors, nothing in z/OS has caused as much confusion as the whole subject of UNIX System Services.

In multiple audits of z/OS systems carried out over recent times, exposures were found because of badly setup UNIX System Services environments. Some of these bad setups were because of misunderstanding the nature of the controls that are available or not understanding that the controls exist. However, in most cases, apparently there was little attention paid to designing the security for UNIX System Services. There has been a failure to understand the risks that are involved.

UNIX System Services have been part of z/OS since z/OS was first available. Previously, it was part of OS/390, and before OS/390 it was part of MVS. It was introduced to MVS in 1993, so at the time of writing, it has been around for 20 years.

UNIX System Services provide a full set of UNIX capabilities to z/OS. It can be configured to be fully POSIX-compliant if that is what is wanted. However, UNIX in its raw form does not adhere to the levels of security controls in the rest of z/OS.

z/OS UNIX therefore added a significant set of extra controls to restrict, limit, and divide authorities that are deemed dangerous or can be otherwise be used to endanger the security or integrity of z/OS. These controls are useless if they are not activated, or if the file systems are created with wide access.

UNIX System Services file systems can contain APF-authorized programs (although there are no *libraries* of APF programs). UNIX System Services file systems can be imported from other systems and then mounted into the file structure.

The subject of UNIX System Services security is wide-ranging, and is covered in *z/OS UNIX Security Fundamentals*, REDP-4193. To secure your UNIX System Services environment, create a design and standards using the contents of this paper.

The following sections describe some items that are recommended for high attention. Some of the following sections assume that you have read *z/OS UNIX Security Fundamentals*, REDP-4193.

9.8.1 z/OS UNIX security

There are two profiles in the RACF class FACILITY that alter the nature of z/OS UNIX security. These profiles are BPX.DAEMON and BPX.SERVER. Without these profiles, the system is said to be in UNIX level security. Without these profiles, the security revolves around which processes run with UID 0.

BPX.DAEMON controls who or what is allowed to take on the identity of other users. A daemon process in UNIX is a process that takes on serial identities of others.

BPX.SERVER controls which processes in UNIX are allowed to manage multiple identities within an address space. Each identity is associated with a process.

If systems run without these profiles, then the security of UNIX System Services is restricted based on who can work in UID 0.

Preferred practice: Always define the BPX.DAEMON and BPX.SERVER profiles in the FACILITY class. Restrict access to these profiles to those users with an absolute need. This is not expected to include human users, just user IDs for processes and daemons.

9.8.2 UID 0

In UNIX systems on many platforms, UID 0 is used to grant access to *administer* the server. However, this level of authority is entirely inappropriate for z/OS UNIX System Services. UID 0 grants far too much privilege.

Several possibilities exist to address this situation.

BPX.SUPERUSER

The first possibility is to allow users to switch to UID 0 when they must, but not set UID 0 with their user ID. This type of access uses the `su` command to switch the UID. To authorize this action, a RACF profile in the FACILITY class is used. The resource name is BPX.SUPERUSER. If a user ID has access to this resource, then it can run the `su` command to switch the effective UID to 0.

Some software packages that require UID 0 privileges are written to use BPX.SUPERUSER.

Although it is apparent that the user is not permanently running with superuser privileges, it does nevertheless mean that all UID 0 privileges are available if the user wants to use them.

UNIXPRIV

To ensure that users are given the least privilege that is needed to perform their job role, use the UNIXPRIV class.

The RACF class UNIXPRIV can be used to subdivide the authorities of UID 0. The set of authorities and the resources that are needed are shown in Table 9-2. This table is taken from *z/OS UNIX System Services Planning*, GA32-0884-00. Consult any later versions of that manual for updates.

Table 9-2 z/OS UNIX privileges for resources

Resource name	z/OS UNIX privilege	Minimum access required
CHOWN.UNRESTRICTED	Allows users to run the chown command to transfer ownership of their own files.	READ
FILE.GROUPOWNER.SETGID	Specifies that a directory's set-gid bit is used to determine the group owner of any new objects that are created within the directory. <i>Not recommended</i> except for privileged users who must administer and install the system.	None required
RESTRICTED.FILESYS.ACCESS	Specifies that RESTRICTED users cannot gain file access by virtue of the 'other' permission bits.	None required
	Can be overridden for a specific user/group.	READ
SHARED.IDS	Allows users to assign UID and GID values that are not unique.	READ
SUPERUSER.FILESYS.ACLOVERRIDE	Specifies that ACL contents override the access that was granted by SUPERUSER.FILESYS.	None required
	Can be overridden for specific users or groups. The user or group must have the same access that would be required for SUPERUSER.FILESYS while accessing the file.	
SUPERUSER.FILESYS (Authorization to the SUPERUSER.FILESYS resource provides privileges to access only local files. No authorization to access Network File System (NFS) files is provided by access to this resource.)	Allows user to read any local file, and to read or search any local directory.	READ
	Allows user to write to any local file, and includes privileges of READ access.	UPDATE
	Allows user to write to any local directory, and includes privileges of UPDATE access.	CONTROL (or higher)
SUPERUSER.FILESYS.CHANGEPERMS	Allows users to run the chmod command to change the permission bits of any file and to use the setfacl command to manage access control lists for any file.	READ

Resource name	z/OS UNIX privilege	Minimum access required
SUPERUSER.FILESYS.CHOWN	Allows user to run the chown command to change ownership of any file.	READ
SUPERUSER.FILESYS.MOUNT	Allows user to run the TSO/E MOUNT command or the mount shell command with the nosetuid option. Also allows users to unmount a file system with the TSO/E UNMOUNT command or the unmount shell command with the nosetuid option. Users who are permitted to this profile can run the chmount shell command to change the mount attributes of a specified file system.	READ
	Allows user to run the TSO/E MOUNT command or the mount shell command with the setuid option. Also allows user to run the TSO/E UNMOUNT command or the unmount shell command with the setuid option. Users who are permitted to this profile can run the chmount shell command on a file system that is mounted with the setuid option.	UPDATE
SUPERUSER.FILESYS.QUIESCE	Allows user to run quiesce and unquiesce commands for a file system that is mounted with the nosetuid option.	READ
	Allows user to run quiesce and unquiesce commands for a file system that is mounted with the setuid option.	UPDATE
SUPERUSER.FILESYS.PFSCTL	Allows user to use the pfscctl() callable service.	READ
SUPERUSER.FILESYS.USERMOUNT	Allows non-privileged users to mount and unmount file systems with the nosetuid option.	READ
SUPERUSER.FILESYS.VREGISTER (The SUPERUSER.FILESYS.VREGISTER resource only lets a server such as NFS initialize. Users who are connected as clients through facilities such as NFS do not get special privileges based on this resource or other resources in the UNIXPRIV class.)	Allows a server to use the vreg() callable service to register as a VFS file server.	READ
SUPERUSER.IPC.RMID	Allows user to run the ipcrm command to release IPC resources.	READ

Resource name	z/OS UNIX privilege	Minimum access required
SUPERUSER.PROCESS.GETPSENT	Allows user to use the <code>w_getpsent()</code> and <code>__getthent()</code> callable services to receive data for any process. Allows users of the <code>ps</code> command to output information about all processes. This is the default behavior of <code>ps</code> on most UNIX platforms.	READ
SUPERUSER.PROCESS.KILL	Allows user to use the <code>kill()</code> callable service to send signals to any process.	READ
SUPERUSER.PROCESS.PTRACE (Authorization to the BPX.DEBUG resource also is required to trace processes that run with APF authority or BPX.SERVER authority.)	Allows user to use the <code>ptrace()</code> callable service through the <code>dbx</code> debugger to trace any process.	READ
SUPERUSER.SETPRIORITY	Allows user to increase own priority.	READ
SUPERUSER.SHMMCV.LIMITS	Allows the user to create up to 4,194,304 mutexes or condition variables to be associated with a single shared memory segment. The overall system total of mutexes and condition variables for authorized users must be less than 134,217,729. When authorized applications create the maximum number of mutexes and condition variables, the system requires more auxiliary storage to be available. System dumps that include the OMVS address space also require larger system dump data sets to contain the increased size of that address space. It is unlikely that applications will create the maximum number of structures that are allowed. If the maximum number is created, the increase in auxiliary storage and system dump data set size is roughly 350 GB.	READ

It should be apparent from the levels of granularity of the privileges that are shown in Table 9-2 on page 241 that users can be restricted a great deal and still can fulfill their roles. Granting of access to UID 0 is rarely necessary.

However, these privileges should be carefully monitored and given only when needed. Consider the following example.

Jill Brown has been tasked to perform a software installation of an application package. The package has had several problems that the vendors are addressing by shipping new versions of the code as a complete file system. After the files system is copied to the z/OS system, it must be mounted.

James, the security administrator, has given Jill the authority to do the mounts herself. However, James failed to understand the significance of the access level to the UNIXPRIV resource SUPERUSER.FILESYS.MOUNT and gave Jill UPDATE access. Unknown to James, Jill is being blackmailed to get copies of customer accounts.

Jill has a System z Hercules emulator at home running an unauthorized copy of z/OS. She prepares a program that is marked as APF-authorized in her file system. The program uses APF authorization to elevate the access privileges of TSO users.

Jill takes a copy of this file system to work on a USB flash drive and mounts it with SETUID privileges. This also acknowledges the APF bit in the file security packet for her program, so she can run her program and get access to the private data.

So, because of a small misunderstanding by a security administrator, a data breach can easily occur.

Preferred practice: Activate the UNIXPRIV class and define discrete resources for all of the privileges. Set the audit settings to log all accesses, both successful and failures. Keep the access lists of these resources to an absolute minimum. Document all cases where these privileges have been granted and why they were granted.

Some software packages for z/OS UNIX System Services have been ported from other UNIX systems, where using UID 0 is a normal way of operating. If so, you have only a few options:

- ▶ Run the package with UID 0 and accept the risk.
- ▶ Put pressure on the vendor to change the way the operating system operates.
- ▶ Reject the package in favor of one that meets your security requirements.

The greatest danger here is not being aware of what authorities are granted, and how they can be used or misused.

9.8.3 BPX profiles

z/OS UNIX System Services uses many other SAF profiles that start with the characters BPX. Most of these profiles are defined in the FACILITY class, and one (BPX.SRV.userid) is in the SURROGAT class.

The list of these profiles varies from time to time as new controls are added. The full list can be found in *z/OS UNIX System Services Planning*, GA22-7800-19. Consult the latest edition of this book for the latest information about these security checks.

Some of these security checks are concerned with restricting access to facilities because they can be used to disrupt normal activity (rather than being associated with data access). Other profiles control access to performance-related options. You might want to restrict access to some resources.

The following sections describe the set of checks that exist at the time of writing. Much of the following text is taken from *z/OS UNIX System Services Planning*, GA22-7800-19.

BPX.CF

This profile controls access to the `_cpl` service. The `__cpl` callable service calculates coupling facility structure sizes that are required by the Coupling Facility Resource Manager (CFRM) policy through a web interface.

BPX.CONSOLE

This profile allows a authorized user the ability to use the `_console()` or `_console2()` services. This service allows programs to run selected commands and wait on the response from MVS.

Preferred practice: If you have UNIX System Services programs that must communicate with the systems operator, or modify their own status or that of other programs, then these programs might need access to this profile. Define and grant access only where it specifically is needed.

BPX.DAEMON

BPX.DAEMON serves two functions in the z/OS UNIX environment:

- ▶ Any superuser that is authorized to use this profile has the daemon authority to change MVS identities through z/OS UNIX System Services without knowing the target user ID's password or password phrase. This identity change can occur only if the target user ID has an OMVS segment defined. If BPX.DAEMON is not defined, then all superusers (UID=0) have daemon authority. If you want to limit which superusers have daemon authority, define this profile and permit only selected superusers to use it.
- ▶ Any program that is loaded into an address space that requires daemon level authority must be defined to program control. If the BPX.DAEMON FACILITY class profile is defined, then z/OS UNIX verifies that the address space has not loaded any executable files that are uncontrolled before it allows any of the following services that are controlled by z/OS UNIX to succeed:
 - `seteuid`
 - `setuid`
 - `setreuid`
 - `pthread_security_np()`
 - `auth_check_resource_np()`
 - `_login()`
 - `_spawn()` with user ID change
 - `_passwd()`

Daemon authority is required only when a program does a `setuid()`, `seteuid()`, `setreuid()`, or `spawn()` user ID to change the current UID without first having run a `_passwd()` call to the target user ID. To change the MVS identity without knowing the target user ID's password or password phrase, the caller of these services must be a superuser. Additionally, if a BPX.DAEMON FACILITY class profile is defined and the FACILITY class is active, the caller must be permitted to use this profile. If a program comes from a controlled library and knows the target UID's password or password phrase, it can change the UID without having daemon authority.

Preferred practice: Define a profile of name BPX.DAEMON and restrict its access list. Do not assign it to any task or user without a full explanation and reasoning.

BPX.DAEMON.HFCTL

This profile controls which users with daemon authority are allowed to load uncontrolled programs from MVS libraries into their address space. BPX.DAEMON.HFCTL does not allow generic profiles.

Preferred practice: Define a profile of name BPX.DAEMON.HFCTL and restrict its access list. Do not assign it to any task or user without a full explanation and reasoning.

BPX.DEBUG

Users with READ access to BPX.DEBUG can debug certain types of restricted processes. These do not include processes that have a PID of 1 or, for users using a default UID, processes that use a default UID. To debug programs that run with APF authority or with BPX.SERVER authority, use DBX to call the ptrace callable service.

Preferred practice: Define a profile of name BPX.DEBUG and restrict its access list. Access to this resource should normally be needed only for problem analysis.

BPX.EXECMVSAPF.program_name

This profile allows unauthorized callers of the execmvs callable service to pass an argument that is greater than 100 characters to an authorized program. If the FACILITY class resource exists, then unauthorized callers can pass arguments greater than 100 characters to the program name that is specified in the FACILITY class profile. Individual users do not need to be given access to the profile. If you do not want unauthorized callers to pass an argument greater than 100 characters to any authorized programs, do not define any BPX.EXECMVSAPF.program_name profiles. To allow certain authorized programs to be called with an argument greater than 100 characters, define a profile for each program:

```
BPX.EXECMVSAPF.YOURPGM
BPX.EXECMVSAPF.MYPGM
```

To allow a group of commonly named authorized programs to be called with an argument greater than 100 characters, define a profile that allows for pattern matching. For example, if you have a set of related programs that all begin with the same three characters, MYP, define the following:

```
BPX.EXECMVSAPF.MYP*
```

As a result, all unauthorized callers can pass an argument greater than 100 characters to any authorized program that begins with the characters MYP.

To allow all unauthorized users the ability to pass any argument up to 4096 characters long to any authorized program, then define one profile that is named BPX.EXECMVSAPF.* However, defining this type of profile is not a preferred practice.

BPX.FILEATTR.APF

This profile controls which users are allowed to set the APF-authorized attribute in a z/OS UNIX file. This authority allows the user to create a program that runs in an APF-authorized manner. This is similar to the authority of allowing a programmer to update SYS1.LINKLIB or SYS1.LPALIB.

Preferred practice: Define a profile of name BPX.FILEATTR.APF and restrict its access list. At the most, assign access to one or more systems programmers.

BPX.NEXT.USER

This profile enables automatic assignment of UIDs and GIDs. The APPLDATA field of this profile specifies a starting value, or range of values, from which RACF derives unused UID and GID values. For more information about BPX.NEXT.USER, see *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

BPX.FILEATTR.PROGCTL

This profile controls which users are allowed to set the program control attribute. Programs that are marked with this attribute can run in server address spaces that run with a high level of authority.

Preferred practice: Define a profile of with the name BPX.FILEATTR.PROGCTL and restrict its access list. At the most, assign access to one or more system programmers.

BPX.FILEATTR.SHARELIB

This profile indicates that extra privilege is required when setting the shared library extended attribute through the `chattr()` callable service. This prevents the shared library region from being misused.

BPX.JOBNAME

This profile controls which users are allowed to set their own job names by using the `_BPX_JOBNAME` environment variable or the inheritance structure on `spawn()`. Users with READ or higher permissions to this profile can define their own job names.

Preferred practice: Define a profile of with the name BPX.JOBNAME and restrict its access list.

BPX.MAINCHECK

This profile extends the enhanced program security protection to your UNIX daemons and servers that do not use RACF execute-controlled programs. BPX.MAINCHECK does not allow generic profiles.

BPX.MAP

This profile controls access to the `_map` and `_map_init` services.

BPX.POE

This profile controls access to the `_poe` service.

BPX.SAFFASTPATH

This profile enables faster security checks for file system and IPC constructs.

When the BPX.SAFFASTPATH FACILITY class profile is defined, the security product is not called if z/OS UNIX can quickly determine that file access will be successful. When the security product is bypassed, better performance is achieved, but the audit trail of successful accesses is eliminated.

BPX.SAFFASTPATH does not allow generic profiles.

Preferred practice: Define the BPX.SAFFASTPATH profile *only* if you are sure that you will *never* need to audit successful accesses to UNIX System Services files. Although the performance advantages of this option might be attractive, *careful attention* should be paid to this decision.

BPX.SERVER

This profile restricts the use of the `pthread_security_np()` service. A user with at least READ or WRITE access to the BPX.SERVER FACILITY class profile can use this service. It creates or deletes the security environment for the caller's thread. This profile also is used to restrict the use of the BPX1ACK service, which determines access authority to z/OS resources. Servers with authority to BPX.SERVER must run in a clean program-controlled environment. z/OS UNIX verifies that the address space has not loaded any executable files that are uncontrolled before it allows any of the following services that are controlled by z/OS UNIX to succeed:

- ▶ `seteuid`
- ▶ `setuid`
- ▶ `setreuid`
- ▶ `pthread_security_np()`
- ▶ `auth_check_resource_np()`
- ▶ `_login()`
- ▶ `_spawn()` with user ID change
- ▶ `_passwd()`

BPX.SMF

This profile checks whether the caller attempting to cut an SMF record is allowed to write an SMF record. It also tests whether an SMF type or subtype is being recorded.

BPX.SHUTDOWN

This profile controls access to the `oe_env_np` service to register and block for OMVS shutdown.

BPX.SRV.userid

This profile allows users to change their UID if they have access to `BPX.SRV.userid`, where *userid* is the MVS user ID that is associated with the target UID. `BPX.SRV.userid` is a RACF SURROGAT class profile.

Preferred practice: Define `BPX.SRV.userid` resources only when required to do so. Do not create a generic profile to cover all user IDs (such as `BPX.SRV.**`).

BPX.STOR.SWAP

This profile controls which users can make address spaces non-swappable. Users that are permitted to have at least READ access to `BPX.STOR.SWAP` can start the `__mlockall()` callable service to make their address space either non-swappable or swappable. When an application makes an address space non-swappable, it might cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, using this service can reduce the installation's ability to reconfigure storage in the future. Any application using this service should warn the customer about this side effect in their installation documentation.

BPX.SUPERUSER

This profile allows users to switch to superuser authority, which is described in 9.8.2, "UID 0" on page 240.

BPX.UNLIMITED.OUTPUT

This profile allows users to use the `_BPX_UNLIMITED_OUTPUT` environment variable to override the default spooled output limits for processes.

BPX.WLMSEVER

This profile controls access to the WLM server functions `_server_init()` and `_server_pwu()`. It also controls access to these C language WLM interfaces:

- ▶ `QuerySchEnv()`
- ▶ `CheckSchEnv()`
- ▶ `DisconnectServer()`
- ▶ `DeleteWorkUnit()`
- ▶ `JoinWorkUnit()`
- ▶ `LeaveWorkUnit()`
- ▶ `ConnectWorkMgr()`
- ▶ `CreateWorkUnit()`
- ▶ `ContinueWorkUnit()`

A server application with read permission to this FACILITY class profile can use both the server functions and the WLM C language functions to create and manage work requests.

Preferred practice: Do not define a generic profile of the form BPX.**.

9.8.4 Sharing IDs

The use of UID 0 and its ability to be shared among several RACF user IDs is a single instance of shared IDs.

Sharing IDs is not a good thing to do. If IDs are shared, then there can be confused access entries in file systems. Processes can be taken over by IDs in unexpected ways.

Preferred practice: Do not share UID values across multiple RACF user IDs.

The use of the default OMVS segment was removed from z/OS V2.1. Every site that previously used this facility must provide an alternative mechanism for providing access to UNIX System Services for those users who used the default OMVS segment.

9.8.5 UNIX file system security

The file system implementations on which this chapter focuses are the z/OS HFS and zFile System (zFS). Other file systems also are supported in z/OS, such as Network File System (NFS) and Temporary File System (TFS). Although they abide with the basic security model that we describe for HFS and zFS, they do not directly involve z/OS data sets, and require specific setups that are not addressed in this book. Further information about NFS can be found in *z/OS V1R8.0 Network File System Guide and Reference*, SC26-7417.

The z/OS UNIX file systems data is kept in data sets that are managed by DFSMS with a specific type of HFS or VSAM linear (the latter for the zFS file system). The z/OS UNIX logical file system gives the UNIX users the view of the data in the HFS or zFS data sets as though they were in files and directories.

In z/OS terminology, a *file system* is one data set that contains z/OS UNIX data. A z/OS system usually has many file systems that make up the full UNIX file tree. One data set hosts the *root* file system and contains the top of the files and directories hierarchy, and the lower levels of the hierarchy are stored in other file systems, that is, data sets, that provide the lower levels of the hierarchy by being logically *mounted* at a directory of the next higher level, as shown in Figure 9-10.

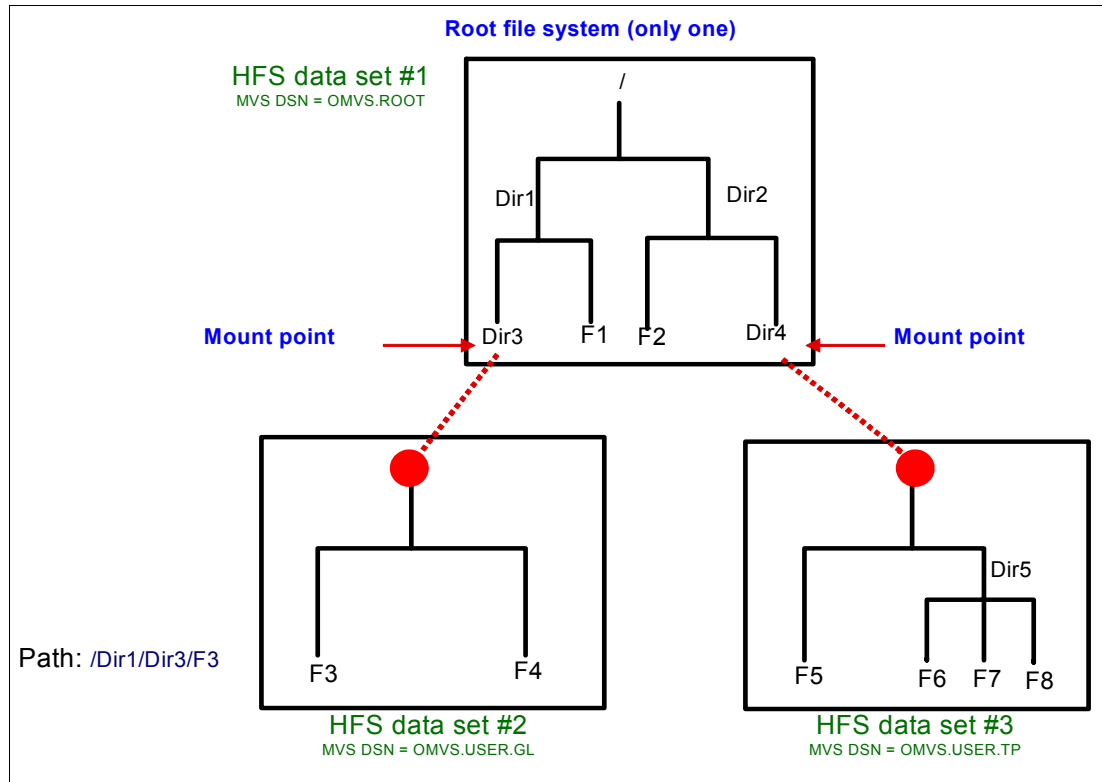


Figure 9-10 The z/OS UNIX Hierarchical File System

A file system is mounted by running the TSO **MOUNT** or UNIX **mount** command, which specifies the directory under which the file system must be mounted. SYS1.PARMLIB(BPXPRMxx) contains the **MOUNT** commands to be automatically run at z/OS UNIX initialization.

Protection of the file system data sets

These data sets must be defined in the RACF DATASET class with UACC(NONE). The z/OS UNIX kernel address space must have access to these data sets. The kernel initialization OMVS STC is defined with the TRUSTED attribute, or its user ID is given UPDATE permission to the Hierarchical File System data sets.

A common convention is to give these data sets a high-level qualifier of OMVS.

Preferred practice: Do not use the user's name for the high-level qualifier for the file system data set because doing so "opens" a door for the user to use his or her MVS identity to manipulate legitimately data in the data set. This data can be data, FSP, or programs that the user is not authorized to modify by using his or her z/OS UNIX identity.

In a sysplex environment, the HFS files may be shared between members of the sysplex. However, certain system directories, such as `/tmp` for temporary files, cannot be shared among sysplex members, so dedicated file systems are required in each member of the sysplex to contain such directories. For more information about HFS file sharing in a sysplex, see *z/OS V1R8.0 UNIX System Services Planning*, GA22-7800.

Mount security

To add file systems dynamically, run the **MOUNT** command and specify the mount point and the data set name to be mounted. The file system has a specific UNIX security environment that can be specified at mount time. There are three optional parameters for the **MOUNT** command (the default value is shown in italics):

MODE (READ/RDWR): **READ** limits access to read-only operations for the files and directories in the mounted file system.

SETUID/NOSETUID: **SETUID** specifies that the `setuid` and `setgid` attributes in the file security packets are to be accepted.

Preferred practice: User-mounted file systems and public file systems such as `/tmp` and `/var` should be mounted as **NOSETUID**.

SECURITY/NOSECURITY: **NOSECURITY** specifies that there is free access to the files and directories that are mounted in the file system, provided the user has search access to the mount point directory and the file system **MODE** is **RDWR**. The files extended attributes, such as “a” (APF) and “p” (program control), are not accepted. File access auditing is still operating. All files that are created in the file system are owned by `UID(0)`, regardless of the `UID` of the actual file creator.

MOUNT requires the requester to have superuser authority or access to the `SUPERUSER.FILESYS.MOUNT` profile in the `UNIXPRIV` class. `UPDATE` access to the profile is required to specify **SETUID**, while `READ` access is sufficient for **NOSETUID** (keep in mind that **SETUID** leads to running code under an identity different from the caller’s identity when the `setuid` or `setgid` file attributes are set).

Preferred practice: Do not use the user’s name for the high-level qualifier for the file system data set because doing so “opens” a door for the user to use his or her MVS identity to manipulate legitimately data in the data set. This data can be data, FSP, or programs that the user is not authorized to modify by using his or her z/OS UNIX identity.

There is no privilege check for the identity that runs the **MOUNT** command that is related to the data set resource. Having superuser privilege or being permitted to access `SUPERUSER.FILESYS.MOUNT` gives **MOUNT** authority for any data set to which the kernel address space is permitted access. It is important to ensure that **MOUNT** authority is given to duly trusted users, as these users can unmount an existing file system and mount it at a different mount point with **NOSECURITY**.

In addition, any public file system such as `/tmp` or `/var` should have the sticky bit attribute turned on to protect the use of these directories. By default, the IBM provided `/etc/rc` script turns on this flag for these directories, but each installation should verify that this is the case on their system. If this is not done, any user can rename and replace any other users files and subdirectories in these public directories.

File security

Many audits of z/OS system have found file systems with access rights that are poorly designed. Managing the permission bits in File Security Packets is not something at which many z/OS users are experienced. Indeed, the nature of the way that the security of new data sets is managed by DFSMS and RACF is such that most users have little or no involvement in the setting of security access rules.

This is not the case with UNIX System Services files. Each file has its own FSP (much like files did in the early days of RACF when we used discrete profiles for RACF-protected data sets.) So, as directories and files are created, each one has a set of permission bits that must be addressed.

Those system programmers who configure UNIX System Services files for packages must be aware of the rules for accessing files, and adjust the permission bits accordingly.

Preferred practice: Provide time and materials to train systems programmers and others performing software installation to know how to manipulate permission bits.

Preferred practice: Ensure that testing is performed on the security of software that is within UNIX System Services. Provide definitions of the security requirements for the files within the file system.

Preferred practice: Monitor update accesses to sensitive configuration files. For those files, make sure that the audit setting in the FSP contain entries for successful write accesses.

9.8.6 Access control lists

In addition to the use of the permission bits in the FSP for each UNIX System Services file, it is possible to use access control lists (ACLs) for each file. There can be up to 1024 ACL entries per file.

Although this mechanism might be the preferred way to restrict or control access if no other way can be designed, it is more complex and should be used sparingly.

Warning: Large file systems with large ACLs on thousands of files increase the size of the file system and reduce the processing speed.

9.8.7 APF and other privileged bits in the FSP

Programs that are marked with the APF bit may use the MODESET SVC to switch state and key. This provides access to system level controls and control blocks, which normally is unavailable to programs.

The program control bit can also be set. This effectively makes the program a trusted program. So, any programs that use these bits should be carefully set with permission bits that grant only the necessary access. The ability to set these bits should be carefully controlled.

Preferred practice: Restrict access to the profiles BPX.FILEATTR.APF and BPX.FILEATTR.PROGCTL.

9.8.8 Sanction lists

As a further level of control over APF-authorized programs and program-controlled programs, a *sanction list* can be constructed and used. The sanction list is specified in the **AUTHPGMLIST** parameter in the BPXPRMxx member of SYS1.PARMLIB. The parameter points to a file name in the */etc* directory.

Sanction lists contain three separate lists that are delineated by three keywords:

▶ **:authprogram_path**

This keyword is the start of a list of directories that is used only in the running of a hfsload (or C dload), exec, spawn, or attach_exec from an authorized program.

▶ **:programcontrol_path**

This keyword is the start of a list of directories that is used only in the running of a hfsload (or C dload), exec, spawn, or attach_exec from an executable file that is running program-controlled.

▶ **:apfprogram_name**

This keyword is the start of a list of program names that are allowed to get control of APF-authorized programs as a result of an exec or spawn. These names are MVS program names.

The sanction list provides an extra layer of control over which programs can be marked as trusted or APF-authorized. However, it has some downsides:

- ▶ The list must be managed. If new APF programs are required, then they must be marked as APF-authorized as normal. In addition, the path names must be entered into the sanction list.
- ▶ The list can be updated dynamically, but the in-storage list is refreshed from that list every 15 minutes.
- ▶ There is a performance impact with the sanction list each time a program is loaded with one of the bits set.

It is a local decision whether the sanction list is a useful line of defense. If a user has managed to get access to UID 0, then he can modify the sanction list and then mark his own programs APF-authorized.

However, if the attacker has gained access to a more granular authority, then the sanction list might prevent further privilege elevation.

9.9 DFSMS

DFSMS consists of several components, each of which manages data. So, the purpose of this section is not so much to describe the management of data security in terms of access lists to individual data sets. The issues that must be addressed are those about how data is managed from the perspective of the storage manager.

In many z/OS installations, there is a need for a person or group of persons to manage data at the macro level. These people are not concerned with the contents of each data set. They are instead concerned with the following aspects of data management:

- ▶ Placement of data sets
- ▶ Naming standards for data sets
- ▶ Management of storage volumes

- ▶ Backup and recovery processes for data sets
- ▶ Removal of unwanted data sets
- ▶ Migration of data sets on a use frequency basis
- ▶ Installation of new storage devices and the migration of data sets to those devices
- ▶ Disposal of old storage devices.

Let us examine each of these items in turn.

9.9.1 Placing data sets

DFSMS provides a mechanism to examine each request for a new instance of a data set and has capabilities to perform the following actions:

- ▶ Alter a data set's attributes (for example, DCB or SPACE parameters)
- ▶ Alter a data set's placement (for example, ensure that it is placed on a different volume)
- ▶ Prevent a data set's allocation (that is, fail the data set creation)

Much of placement is related to data set standards. Such standards are not really at the core of security, but you must remember that security rules frequently rely on naming standards for data sets, and the SMS rules also frequently rely on such standards.

So, the most important aspect of DFSMS rules for altering the position or shape of a data set is that in doing so should not make any change that alters the security characteristics of the data set. This alteration might occur if a data set was placed onto a volume where access might be achieved through a different path. So, if a data set allocation was redirected to a volume that was shared with another z/OS system that used a different security database, this might affect the security of the data. There also might be an effect on security if DASDVOL profiles are being used, and the access list for the requested DASDVOL resource is different than the access list for the volume used.

Preferred practice: Storage administration should work with security administration to ensure that data placement does not alter the intended security of a data set.

9.9.2 Naming standards for data sets

Storage administration and security administration must cooperate regarding the assignment of data set naming standards. DFSMS rules are geared to data set names in many cases. Security rules must take DFSMS rules into consideration.

9.9.3 Managing storage volumes

Storage administrators need the ability to manage data sets at the macro level, that is, they frequently must be able to copy, move, back up, and restore data sets. However, they should not be accessing the data within those data sets.

In many enterprises, this task is managed by giving storage administration users the OPERATIONS attribute. However, this attribute grants more access than is needed for their job role. OPERATIONS potentially allows full access to every data set on the z/OS system.¹⁴

¹⁴ OPERATIONS takes effect only after an access list is searched. So, if the user is denied access by a USER or by all GROUP entries in the access list, then access is denied despite the ownership of the OPERATIONS attribute.

Some z/OS installations use DASDVOL profiles. These profiles allow data sets to be manipulated by using the DFSMS component called DFSMSdss. This is a sophisticated memory dump, restore, copy, and move program that can use DASDVOL profiles for authorization.

DASDVOL profiles are widely misunderstood. The authors have seen many instances of generic DASDVOL profiles that grant far too much access to IT staff. For example, a data set may be deleted if ALTER access is granted to the DASDVOL resource matching the volume on which a data set exists, even if the access list of the data set denies that access.

A better way

IBM has designed controls for storage administrators so that the use of OPERATIONS and DASDVOL profiles can be phased out.

These controls use RACF resources in the FACILITY class. These resources start with the prefix STGADMIN, and have many forms. For example, storage administrators working with DFSMSdss can be granted access to the following resources,

- ▶ STGADMIN.ADR.STGADMIN.COMPRESS
- ▶ STGADMIN.ADR.STGADMIN.COPY
- ▶ STGADMIN.ADR.STGADMIN.COPY.DELETE
- ▶ STGADMIN.ADR.STGADMIN.COPY.RENAME
- ▶ STGADMIN.ADR.STGADMIN.DEFRAG
- ▶ STGADMIN.ADR.STGADMIN.DUMP
- ▶ STGADMIN.ADR.STGADMIN.DUMP.DELETE
- ▶ STGADMIN.ADR.STGADMIN.RELEASE
- ▶ STGADMIN.ADR.STGADMIN.RESTORE
- ▶ STGADMIN.ADR.STGADMIN.RESTORE.RENAME

However, *standard* users of DFSMSdss (that is, not storage administrators) might be granted access to the following resources,

- ▶ STGADMIN.ADR.DUMP.CNCURRNT
- ▶ STGADMIN.ADR.DUMP.INCAT
- ▶ STGADMIN.ADR.DUMP.PROCESS.SYS
- ▶ STGADMIN.ADR.DUMP.TOLERATE.ENQF
- ▶ STGADMIN.ADR.RESTORE.BYPASSACS
- ▶ STGADMIN.ADR.RESTORE.DELCATE
- ▶ STGADMIN.ADR.RESTORE.IMPORT
- ▶ STGADMIN.ADR.RESTORE.TOLERATE.ENQF
- ▶ STGADMIN.ADR.COPY.BYPASSACS
- ▶ STGADMIN.ADR.COPY.CNCURRNT
- ▶ STGADMIN.ADR.COPY.INCAT
- ▶ STGADMIN.ADR.COPY.PROCESS.SYS
- ▶ STGADMIN.ADR.COPY.TOLERATE.ENQF
- ▶ STGADMIN.ADR.CONVERTV
- ▶ STGADMIN.ADR.DEFRAG
- ▶ STGADMIN.ADR.RELEASE.PROCESS.SYS
- ▶ STGADMIN.ADR.RELEASE.INCAT

These lists of resource names might not be exhaustive. There are certainly STGADMIN resources in the FACILITY class that are checked by other components of DFSMS, such as catalog management, DFSMSshm, and DFSMSrmm.

The resource names that can be used and the capabilities that they control can be found in the administration manuals for the various DFSMS components:

- ▶ *z/OS DFSMSdftp Storage Administration*, SC26-7402-15
 - STGADMIN.IGG. profiles
 - STGADMIN.IDC. profiles
 - STGADMIN.IGD. profiles
 - STGADMIN.IGWSHCDS. profiles
- ▶ *z/OS DFSMSdss Storage Administration*, SC35-0423-15 for STGADMIN.ADR. profiles
- ▶ *z/OS DFSMSShsm Storage Administration*, SC35-0421-12 for STGADMIN.ARC. profiles
- ▶ *z/OS DFSMSrmm Implementation and Customization Guide*, SC26-7405-12 for STGADMIN.EDG. profiles

Preferred practice: Examine the STGADMIN profiles that are available at each new release of z/OS and carefully use them. Phase out all use of the OPERATIONS attribute and DASDVOL profiles.

9.9.4 Backup and recovery processes for data sets

The backup and recovery of data sets normally is an automated process that uses DFSMSShsm and DFSMSdss. As shown in “A better way” on page 255, the use of STGADMIN profiles in the FACILITY class is the preferred way to proceed.

9.9.5 Removing unwanted data sets

The removal of unwanted data sets might occur at various times:

- ▶ A user ID must be deleted because a user has left the organization.
- ▶ Data was moved from one volume to another, and the old volume must be decommissioned.
- ▶ Data must be removed because of compliance requirements.

In each case, a decision must be made whether a backup is taken before removal. In the case of compliance, it is likely that all copies should be removed.

Storage administrators usually know that when a data set is deleted that the data that is associated with the data set frequently is left in place on the disk. Data set deletion involves the removal of an entry in the VTOC of the storage volume, and the returning of the space that is occupied by the data set to the free space pool for the volume.

Thus, if a data set is created that uses the free space that was relinquished, then the data that is associated with the old data set can be accessed.¹⁵ Therefore, data sets that contain sensitive information must be given the ERASE attribute in the RACF profile that protects them.

9.9.6 Migrating data sets on a use frequency basis

The migration of data sets from one storage group or medium to another, based on use frequency, is managed by the DFSMS component called DFSMSShsm (frequently known as HSM).

¹⁵ In later levels of DFSMS, each new data set has an end-of-file (EOF) marker set at the start of the data set during data set creation. This means that an attempt to read data using QSAM or BSAM returns an EOF situation signaling end of file. However, more sophisticated access method capabilities can still access the data beyond the track containing the EOF marker.

Control of access to the capabilities of HSM can be achieved by using the STGADMIN.ADR. profiles in the FACILITY class (see “A better way” on page 255).

The migration and recall of data sets using HSM is equivalent to the moving of data from one volume to another.

9.9.7 Installing storage devices and migrating data sets to those devices

This type of activity is related to moving data sets from volume to volume, so the same issues apply as mentioned in 9.9.1, “Placing data sets” on page 254.

9.9.8 Disposing old storage devices

The disposal of storage devices normally requires the erasure of all data on the volume. However, if Full Disk Encryption (FDE) is used, then the data is cryptographically erased after the authentication credentials are removed from the Security Key Lifecycle Manager servers for the IBM System Storage DS8000 device.

9.10 JES and SDSF

JES is the primary subsystem that is a required component of any z/OS system. The JES that runs can be JES2, JES3, or theoretically a locally developed subsystem. The authors have never come across a locally developed primary subsystem.

SDSF is the Spool Search and Display Facility. It provides a set of panels and dialogs that provide a simple interface for interfacing with JES2 or JES3.

9.10.1 JES command security

Each JES has a set of commands that must be secured in much the same manner as MVS commands (see 9.2.3, “MVS commands” on page 204).

The resources that protect the JES commands can be found in the JES2 and JES3 manuals:

- ▶ “Providing security for JES2” in the *z/OS JES2 Initialization and Tuning Guide*, SA22-7532-11, found at:
http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.hasa300/secchap.htm%23secchap
- ▶ “Providing Security for JES3” in the *z/OS JES3 Initialization and Tuning Guide*, SA22-7549-10, found at:
http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.iata600/prov.htm%23prov

Access to each type of command must be examined carefully and access granted to the various role groups at your enterprise as needed.

Because JES2 and JES3 introduce new features and capabilities from time to time, new commands might be unprotected.

Preferred practice: Create a fallback generic profile for JES2 or JES3 commands names so that new commands are covered by that profile. In most z/OS installations, this means creating a profile of JES2.** or JES3.** in the OPERCMDS class. The profile has a null access list and a UACC of NONE.

Access to JES commands should be restricted to operations staff and systems programmers or administrators who need to perform job management. Other users of z/OS who normally use TSO should use SDSF to gain access to JES2 and JES3 facilities.

9.10.2 NODES profiles

Profiles may be defined in the NODES class to define the levels of trust that are bestowed on other z/OS systems¹⁶ that communicate by using Network Job Entry (NJE). NODES profiles are explained in detail in “Commands from NJE Nodes”, in *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683-15, found at:

http://www.ibm.com/support/knowledgecenter/#/SSLTBW_1.13.0/com.ibm.zos.r13.icha700/cmdnje.htm%23cmdnje

NODES profiles have a different means of operation from most profiles in RACF. For example, the access list on the profiles is not used at all. However, the UACC setting is important, as it defines the level of trust for the NJE node in question.

If you use NJE and you define profiles for those NJE nodes with a UACC value of UPDATE, then this means that you are trusting the remote node to perform user validation. This should be configured only when the remote node is managed such that either of the following is true:

- ▶ The RACF databases are effectively identical (by virtue of RRSF, or similar).
- ▶ There is a formal recognized trust arrangement between the management of the two nodes.

If you use a UACC value of READ, then you are still trusting the remote node to send properly constructed data conforming to NJE standards. It might be possible for mal-constructed NJE control blocks to alter the behavior of your system.¹⁷

&RACLNDE

The RACF profile &RACLNDE may be defined in the RACFVARS class and may be given member names that correspond to NJE nodes. If this is done (and the RACFVARS class is RACLSTed), then all the NJE nodes that are defined in &RACLNDE are treated as LOCAL nodes and are trusted to perform user ID validation.

Preferred practice: Use &RACLNDE only for those z/OS systems that share a RACF database or a copy of it that is kept in synchronization using RRSF. If you have a development system with different security management, then *do not* define that system in &RACLNDE; use NODES profiles with UACC=READ or UACC=NONE as appropriate.

9.10.3 JESINPUT profiles

JESINPUT profiles are important when you use JES2 RJE workstations or JES3 RJP workstations.¹⁸

¹⁶ Or possibly z/VM systems using RSCS.

¹⁷ To the authors' knowledge, no evidence for this items or the contrary has been tested.

¹⁸ In the authors' experience, these workstations rarely are used today.

If you are using these workstations, then define them to the JESINPUT class and allow appropriate access.

9.10.4 JESJOBS profiles

JESJOBS profiles can be used for several different functions. They can control submission of jobs, cancellation of jobs, and the use of JES job classes.

JOBNAME controls at job submission time

The prime use of JESJOBS profiles is to control the use of job names. Profiles with the following structure are used:

```
SUBMIT.nodename.jobname.userid
```

If a given user has READ access to a resource, then the user may submit a job for processing. Submit in this context means that the user ID can use the internal reader capability to pass JCL to the primary JES.

In the profile structure above, *nodename* is the NJE node name for the local node, *jobname* is taken from the jobcard for the job being submitted, and *userid* is the user ID that is associated with the job to be run. This user ID might be the submitting user ID.

JESJOBS profiles also may be used to prevent submission of jobs for a specific user ID. So, if a profile of the structure SUBMIT.*nodename.*.USER01* is defined, and USER02 has no access to this profile, then USER02 cannot submit jobs for USER01. This applies even if USER02 knows the password for USER01.

CANCEL controls

If a user attempts to cancel a job, then access to the following profile is checked:

```
CANCEL.nodename.userid.jobname
```

This profile can be used to control who can cancel jobs for a given user ID, or who can cancel jobs for a specific jobname, or many combinations in between.

JOBCLASS controls

Job class controls are new with z/OS V2R1.

The controls can be configured to check either that the submitting user has access to the job class, the execution user ID has access to the job class, or both. Control is provided by creating the following profiles in the FACILITY class:

- ▶ JES.JOBCLASS.OWNER
- ▶ JES.JOBCLASS.SUBMITTOR

Resource names with the following structure are used:

```
JOBCLASS.nodename.classname.jobname
```

The user ID (or user IDs) needs READ access to the above profile, where *nodename* is the NJE node name of the local node, *classname* is the name of the class for the job, and *jobname* is taken from the job card.

Preferred practice: If your JES definitions allow BLP to be used on a restricted list of classes, provide JOBCLASS profiles in the JESJOBS class to restrict who can use those classes.

9.10.5 SURROGAT profiles

SURROGATE profiles may be used to allow one user to submit jobs for another user without knowledge of the second user's password.

SURROGATE profiles are frequently used in job scheduling software to allow the job scheduler to submit multiple jobs to run under separate user IDs.

Profiles of the form `execution-userid.SUBMIT` can be defined and access then granted to some other user ID. This then allows the user IF to submit jobs for the `execution-userid`.

Preferred practice: Use surrogate job submission with great care. There is no control that is supplied over the nature of the JCL that is run under the `execution-userid`. So, anyone having that surrogate authority can do anything that the `execution-userid` can do. SURROGAT profiles are the source of many exposures in the authors' experience.

9.10.6 WRITER profiles

WRITER profiles can be used with NODES profiles to provide protection of NJE resources or other output devices.

If READ access is not allowed to a WRITER resource, then the output devices cannot be used. For NJE, the resource name is `jesname.NJE.nodename`.

So, if there is no access to the node name, the output cannot be routed to that node. This can be used as part of a data loss prevention (DLP) strategy.

9.10.7 SDSF protection

SDSF provides a window into JES2 and JES3 services from TSO. It is a useful way to provide users access to batch jobs on the input, execution, and output queues.

SDSF also provides a means to allow selected users to use MVS and JES commands. However, it frequently is an easy way for attackers to discover much about your systems.

SDSF can be controlled by using a set of SDSF parms. This is not the way that a secure environment should configure SDSF security.

The use of RACF requires that the SDSF class is active. There is a wealth of security profiles that can be defined and many ways of allowing users to perform actions.

The authors have often encountered situations where SDSF security was not well-designed. Too much access has been granted to far too many users.

Preferred practice: The design of the security controls for SDSF for each installation is different. However, each of these controls should be carefully designed with respect to the role of each group requiring access. The usual principle of minimum access should be carefully observed.

SDSF frequently is used to supply access to the SYSLOG. Often, this is granted to users who never use it, or is simply granted to all users.

Preferred practice: Restrict access to SYSLOG to those users who are performing tasks requiring it. Do not grant read access to SYSLOG to all users.

Preferred practice: Produce a design document for SDSF controls and ensure that it relates to the roles of SDSF users. Testing of the effectiveness of the controls is needed because of the complexity of the mechanisms that are used by SDSF.

9.11 TSO/E

Time Sharing Option Extended (TSO/E) is a base element of z/OS. It allows users to share interactively computer time and resources. In general, TSO/E makes it easier for people with all levels of experience to interact with the z/OS system. TSO/E users include system programmers, application programmers, information center administrators, information center users, TSO/E administrators, and others who access applications that run under TSO/E. It originally contained a set of commands and a set of system services that users and programmers could use when writing their own commands. During its lifecycle, it has been continually enhanced and, in combination with ISPF, which runs under TSO/E, it is often the primary user interface to the z/OS system.

9.11.1 Regulating access to the system

To regulate access to the system, you can use either the RACF data base or SYS1.UADS, which is the user attribute data set (UADS). However, the RACF data base is the preferred choice. Passwords in the UADS data set are stored in plain text and not encrypted, so it is important that the UADS data set has a universal access level or UACC of NONE. If RACF is installed, you should use the RACF data base to regulate access to the system and store information about each TSO/E user. The RACF data base contains profiles for every entity (users, data sets, or groups) that is defined to RACF. If you use the RACF data base to maintain information about TSO/E users, additional information about users is also stored in the RACF data base. For more information about the RACF data base, see the *z/OS Security Server RACF System Programmer's Guide*. You can find different versions of this guide at this website:

<http://publibz.boulder.ibm.com/cgi-bin/bookmgr/DOCNUM/SA22-7681/CCONTENTS?#HDRPART3>

To add, change, and delete user IDs, use RACF commands or panels. The *z/OS Security Server RACF Command Language Reference* provides an overview of the RACF commands that you can use to maintain user information. You can find different versions of this guide at this website:

<http://publibz.boulder.ibm.com/cgi-bin/bookmgr/DOCNUM/SA22-7687/CCONTENTS?#HDRPART3>

9.11.2 Limiting the use of TSO/E commands

Limit the use of certain TSO/E commands at your installation. For example, you might want to limit certain commands to conserve resources, such as spool space. You can restrict the TSO/E commands that users can use from the following interfaces:

- ▶ READY mode TSO/E
- ▶ The TSO/E Session Manager
- ▶ Background mode
- ▶ ISPF/PDF panels

Table 9-3 lists the commands that you can limit in each environment and the functions you can use to limit the commands.

Table 9-3 Commands that you can limit

Environment	Commands that you can limit	Method
READY mode TSO/E	ACCOUNT, OPERATOR, RACONVERT, and SYNC	ACCOUNT and RACF commands, and the logon exit IKJEFLD or IKJEFLD1
	SUBMIT, OUTPUT, STATUS, and CANCEL	ACCOUNT and RACF commands, and exit routines
	CONSOLE and CONSPROF	RACF commands, logon exit IKJEFLD or IKJEFLD1, and the CONSOLE and CONSPROF exit routines
	PARMLIB and TESTAUTH	RACF commands and exit routines
	TRANSMIT, RECEIVE, SEND, LISTBC, LOGON, LOGOFF, ALLOCATE, ALTLIB, TEST, FREE, OUTDES, PRINTDS, EXEC, and OPERATOR SEND	Exit routines
TSO/E Session Manager	Any TSO/E command	Session Manager exit routines
Background mode	Any TSO/E command	SYS1.PARMLIB member IKJTSoxx or CSECT IKJEFTNS
ISPF/PDF panels	Any TSO/E command	ISPMTCM macro statement

9.11.3 Limiting access to data sets

You also need to control the access TSO/E users have to data sets. For example, as a preferred practice, allow users to allocate only data sets having a high-level qualifier of the user's user ID.

You can use RACF to specify which users can access the following items:

- ▶ Non-VSAM and VSAM data sets
- ▶ Generation data groups

To RACF protect a data set, use either RACF commands or ISPF RACF panels to build a profile for the data set. The profile contains information about the users who can access the data set.

With RACF installed, security label checking can be activated. In this case, each data set and each user has security labels that are associated with them. The security label of the data set and the security label of the user's current session are checked. Access to the data set is determined by the result of that check.

For more information about using RACF to protect data sets, see *z/OS Security Server RACF Security Administrator's Guide*. You can find different versions of this guide at this website:

<http://publibz.boulder.ibm.com/cgi-bin/bookmgr/DOCNUM/SA22-7683/CCONTENTS?>

9.11.4 Summary of TSO/E resources that are protected by using RACF

Table 9-4 summarizes the RACF profiles that are needed to protect various TSO/E resources.

Table 9-4 Summary of resources that are protected by using RACF

RACF class name	Profile name	Resource protected
TSOPROC	Procedure name	Logon procedure
ACCTNUM	Account number	Account number for TSO/E session
PERFGRP	Performance group	Performance group for TSO/E session
TSOAUTH	ACCT	ACCOUNT , SYNC , and RACONVRT commands
	JCL	SUBMIT , CANCEL , OUTPUT , and STATUS commands
	MOUNT	Allows this user to run dynamic allocation requests that result in the need for volume mounting
	OPER	OPERATOR command
	RECOVER	EDIT command recovery facility
	PARMLIB	PARMLIB command (read access for LIST, UPDATE access for UPDATE)
	TESTAUTH	TESTAUTH command
	CONSOLE	CONSOLE and CONSPROF commands

9.11.5 IKJTSO00

This member of SYS1.PARMLIB can contain definitions of programs and commands that can be run in an APF-authorized state during TSO sessions.

There are three lists:

▶ AUTHPGM

This is the list of batch programs that can be started by using a **CALL** statement in TSO. When the programs are started, they can be passed a 100 byte parameter in a similar manner to what is used by the **PARM** statement in JCL.

The program runs and completes its role and then passes control back to the caller. The caller can be one of the following items:

- TSO in READY mode
- A REXX exec
- A TSO CLIST
- A program that has started the designated program through the IKJEFTSR service.

Typically, the list includes programs such as IEBCOPY¹⁹ and IDCAMS.

▶ AUTHCMD

This is a list of TSO commands that can run APF-authorized. These commands include the RACF commands, such as **ADDUSER**, **PERMIT**, and **CONNECT**.

▶ AUTHTSF

This is a list of programs that may be started APF-authorized when started by using the IKJEFTSR service. Some programs should not be allowed in this list. *IDCAMS should not be placed in this list* according to *z/OS MVS Initialization and Tuning Reference*, SA22-7592. Placing IDCAMS in AUTHTSF is an integrity exposure.

Preferred practice: The authors frequently have found IKJTSOxx specifications with IDCAMS placed in AUTHTSF. It appears there are third-party software products that have advised that IDCAMS be placed here. If you have IDCAMS placed here, then it should be removed. If this causes problems with any software product, then this should be taken up with the vendor.

When authorized code runs under TSO, the code runs on a parallel task control block (TCB). The normal TCB is placed in a wait state while the parallel TCB runs the authorized code. The normal TCB runs in problem program state, and the parallel TCB runs in supervisor state.

Thus, the management of the running of authorized code under TSO is rather more complex than when using JCL to switch from one program (that might not require APF authorization) to another that does require APF authorization. In JCL, this involves a job step change.

It is important that APF-authorized code that runs under TSO is coded carefully so as not to introduce any integrity exposures.

9.12 Integrated Cryptographic Service Facility

The Integrated Cryptographic Service Facility (ICSF) is a software component of z/OS that is available with z/OS or as a downloaded SMP/E installable web deliverable that provides secure, high-speed cryptographic services in the z/OS environment. ICSF provides the following components:

- ▶ An API for cryptographic operations
- ▶ Key management and keystores (CKDS, PKDS, and TKDS) for cryptographic key material

¹⁹ IEBCOPY does not need to be authorized for z/OSV 1.13 and later.

- ▶ Access to high-speed hardware Cryptographic Coprocessors and Accelerators, and CP Assist for Cryptographic Function (CPACF)

9.12.1 Protecting ICSF keys and APIs

ICSF defines several SAF resource classes to protect its keys and APIs:

- ▶ CSFSERV protects the APIs and panels.
- ▶ CSFKEYS protects cryptographic keys in the CKDS and PKDS.
- ▶ CRYPTOZ protects cryptographic tokens in the TKDS.
- ▶ XCSFKEY protects symmetric key export of keys in the CKDS.

Health checks can examine the status of general resource classes and also the security characteristics of system-critical data sets. Here are the ICSF-related RACF Health Checks:

- ▶ RACF_SENSITIVE_RESOURCES, for the ICSF key data sets
- ▶ RACF_CSFSERV_ACTIVE, for the CSFSERV resource class
- ▶ RACF_CSFKEYS_ACTIVE, for the CSFKEYS resource class

For more information, see *IBM Health Checker for z/OS User's Guide*.

CSFSERV

The CSFSERV class controls access to ICSF callable services and ICSF TSO panel utilities. A UACC of NONE is preferred for customer installations. Access to cryptographic services and panels should be granted only to those processes and people who need access.

CSFKEYS

The CSFKEYS class controls access to the cryptographic keys in the CKDS and PKDS. A UACC of NONE is preferred for customer installations. Access to specific keys or tokens should be granted only to those processes and people who need access.

CRYPTOZ

The CRYPTOZ class controls access to and defines a policy for PKCS#11 tokens in the TKDS. PKCS#11 token access control is unique in that the tokens have different access levels and a differentiation between standard users and security officers. For each token, there are two resources in the CRYPTOZ class for controlling access to tokens:

- ▶ The resource USER.token-name controls the access of the User role to the token.
- ▶ The resource SO.token-name controls the access of the Security Officer (SO) role to the token.

Table 9-5 shows the SAF access levels for CRYPTOZ resources.

Table 9-5 SAF access levels for CRYPTOZ resources

CRYPTOZ resource	SAF access level		
	Read	Update	Control
SO.token-label	Weak SO Can read, create, delete, modify, and use public objects.	SO R/W Has the same capabilities as Weak SO, and can create and delete tokens.	Strong SO Has the same capabilities as SO R/W, and can read but not use private objects, that is, create, delete, and modify private objects.
USER.token-label	User R/O Can read and use public and private objects.	Weak User Has the same capabilities as User R/O, and can create, delete, and modify private and public objects. Cannot add, delete, or modify certificate authority objects.	User R/W Has the same capabilities as Weak User, and can add, delete, and modify certificate authority objects.

For more information, see *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521-15, found at:

http://www.ibm.com/support/knowledgecenter/#/SSLTBW_1.13.0/com.ibm.zos.r13.csfb300/toc.htm

XCSFKEY

The XCSFKEY class controls the ability to export a symmetric key with the Symmetric Key Export (CSNDSYX, CSNFSYX, or CSNDSXD) callable service.

By enabling the Symmetric Key Label Export control for AES or DES keys, and creating profiles in the XCSFKEY resource class, you can increase the level of access authority that is needed to export AES or DES keys without increasing the level of authority that is needed to access the keys for other operations.

9.12.2 Protecting key data sets

To ensure that no one has direct access to your ICSF key data set (KDS), it is a preferred practice to protect your KDS data set name resource in the DATASET class. If a data set profile is used, as opposed to using the **PROTECTALL(FAIL)** option, for example, the profile should have a UACC of NONE.

Access to the keys within the KDS data sets can be obtained through the ICSF API calls or the ICSF ISPF interface. As an additional layer of security, the CSFKEYS general resource class can be set up to protect individual keys, and the CSFSERV general resource class can be set up to protect the ICSF API calls and the ICSF ISPF interface.

9.12.3 Keystore Policy

A Keystore Policy can be established to define rules for the use of encrypted key tokens that are stored in the CKDS and PKDS.

Each Keystore Policy control is a resource in the XFACILIT class. The controls are enabled by creating a profile for the resource by using the **RDEFINE** command. The controls are disabled by deleting the profile by using the **RDELETE** command.

Here are the Keystore Policy controls:

- ▶ Key Token Authorization controls verify that the user has authority to the key tokens in the CKDS or PKDS. Here are the controls in this class:
 - CSF.CKDS.TOKEN.CHECK.LABEL.WARN
 - CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
 - CSF.PKDS.TOKEN.CHECK.LABEL.WARN
 - CSF.PKDS.TOKEN.CHECK.LABEL.FAIL
- ▶ Default Key Label Checking controls specify that ICSF should use a default profile (CSF-CKDS-DEFAULT or CSF-PKDS-DEFAULT in the CSFKEYS class) to determine application access to tokens that are not stored in the CKDS or PKDS. These controls can be enabled only if key token authorization controls are enabled. Here are the controls in this class:
 - CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL
 - CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL
- ▶ Duplicate Key Token Checking controls prevent applications from storing duplicate tokens in the CKDS or PKDS. Here are the controls in this class:
 - CSF.CKDS.TOKEN.NODUPLICATES
 - CSF.PKDS.TOKEN.NODUPLICATES
- ▶ Granular Key Label Access controls increase the level of access authority that is required to create, write to, or delete a key label. Here are the controls in this class:
 - CSF.CSFKEYS.AUTHORITY.LEVELS.WARN
 - CSF.CSFKEYS.AUTHORITY.LEVELS.FAIL
- ▶ Symmetric Key Label Export controls specify that the profiles in the XCSFKEY class should be used to determine access to the AES or DES keys that an application is attempting to export by using the Symmetric Key Export (CSNDSYX, CSNFSYX, or CSNDSXD) callable service. Here are the controls in this class:
 - CSF.XCSFKEY.ENABLE.AES
 - CSF.XCSFKEY.ENABLE.DES
- ▶ PKA Key Management Extensions controls specify that the ICSF segment of profiles in the CSFKEYS class (and the XCSFKEY class when a Symmetric Key Label Export control is enabled) are checked to determine additional restrictions about how keys that are covered by the profile can be used. Here are the controls in this class:
 - CSF.PKAEXTNS.ENABLE.WARNONLY
 - CSF.PKAEXTNS.ENABLE

For more information, see *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521-15.

9.12.4 Trusted Key Entry Workstation

IBM offers the Trusted Key Entry Workstation (TKE) as a means for ensuring secure creation and management of key material and for managing the crypto adapters on the host. For more information, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*, SA23-2211-06, found at:

http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.csfb600/toc.htm

9.12.5 Access Control Points

Access Control Points (ACPs) must be enabled in the ICSF Role for each service or subservice to be run on a cryptographic coprocessor. It is preferable to enable ACPs only for callable services that you use. If an access control point is disabled, the corresponding ICSF callable service fails during execution with an access denied error. TKE is required to modify ACP settings.

A new or a zeroized coprocessor (or domain) comes with an initial set of ACPs that are enabled by default. All other ACPs, representing potential future support, are left disabled.

When a firmware upgrade is applied to an existing cryptographic coprocessor, the upgrade might introduce new ACPs:

- ▶ If TKE is enabled or an EP11 card is configured, the firmware upgrade does not retroactively enable the new ACPs. These ACPs must be enabled through the TKE (or subsequent zeroize) to use the new support.
- ▶ If TKE is not enabled, the firmware upgrade retroactively updates the new ACPs that would be enabled by default.

For a list of ACPs, see the following publications:

- ▶ For CCA callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SA22-7522-14, found at:
http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.csfb400/toc.htm
- ▶ For CCA utilities, see *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521-15, found at:
http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.csfb300/toc.htm
- ▶ For PKCS#11 mechanisms and functions, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*, SA23-2231-03, found at:
http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.csfb00/toc.htm

9.12.6 Disabling SAF checks for improved performance

To improve your system performance, disable the SAF checks that are described in the following sections:

- ▶ “CSFIQF and CSFIQF2”
- ▶ “CSNBOWH and CSNBRNG” on page 269

CSFIQF and CSFIQF2

Services that make calls to the cryptographic coprocessors are protected by the CSFSERV class. For example, the CSFIQF service, which makes calls to the cryptographic coprocessors, is protected by the CSFSERV class. However, CSFIQF2, which returns information from internal ICSF control blocks rather than calling the cryptographic coprocessors, is not SAF protected. Using CSFIQF2 instead of CSFIQF can improve performance.

CSNBOWH and CSNBRNG

CSFSERV authorization checks for the One Way Hash (CSNBOWH) and Random Number Generate (CSNBRNG) callable services contribute to significant CPU consumption. Disabling the SAF check, if it makes sense for your installation, might improve the performance of applications that call these services.

Two resources in the XFACILIT class can be defined to disable SAF checking:

- ▶ CSF.CSFSERV.AUTH.CSFOWH.DISABLE
- ▶ CSF.CSFSERV.AUTH.CSFRNG.DISABLE

9.12.7 Special Secure Mode

Special Secure Mode (SSM) can be enabled to allow clear keys to be imported into the CKDS. Typically, Special Secure Mode is disabled. For those instances where clear key import is necessary, SSM can be enabled using the SSM setting in the ICSF options data set (requiring an ICSF restart) or by defining the CSF.SSM.ENABLE resource in the XFACILIT class (dynamically).

9.13 Started procedures

A procedure consists of a set of JCL statements that are frequently used together to achieve a certain result. Procedures usually are in the system procedure library, SYS1.PROCLIB, which is a partitioned data set. A started procedure usually is started by an operator, but can be associated with a functional subsystem. For example, DFSMS is treated as a started procedure even though it does not need to be started with a **START** command.

Only RACF defined users and groups can be specifically authorized to access RACF-protected resources. However, started procedures have system-generated **JOB** statements that do not contain the **USER**, **GROUP**, or **PASSWORD** parameters.

To enable started procedures to access the same RACF-protected resources that users and groups access, started procedures must have RACF user IDs and group names. By assigning them RACF identities, your installation can give started procedures specific authorization to access RACF-protected resources. For example, you can allow JES to access spool data sets. To associate the names of started procedures with specific RACF group names and user IDs, you and your RACF system programmer can do one of the following actions:

- ▶ Set up the STARTED class (the preferred method).
- ▶ Create a started procedures table (ICHRIN03).

Let us take a close look at the following details:

- ▶ Assigning RACF user IDs to started procedures
- ▶ Authorizing access to resources
- ▶ Setting up the STARTED class
- ▶ Using the started procedures table (ICHRIN03)
- ▶ Started procedure considerations

9.13.1 Assigning RACF user IDs to started procedures

As with any other user ID and group name, the user ID and group name that you assign to a started procedure must be defined to RACF by using the **ADDUSER** and **ADDGROUP** commands, and the user must be connected to the group. You also might need to use the **PERMIT** command to authorize the users or groups to get access to the required resources.

Protected user IDs

The user IDs that you assign to started procedures should have the **PROTECTED** attribute. Protected user IDs are user IDs that have the **NOPASSWORD**, **NOPHRASE**, and **NOOIDCARD** attributes. They are defined or modified by using the **ADDUSER** and **ALTUSER** commands.

Protected user IDs cannot be used to log on to the system, and are protected from being revoked through incorrect system access attempts. The following example shows a protected user ID being defined for a CICS region, and an existing user ID that is used by JES being given the **PROTECTED** attribute:

```
ADDUSER CICS03 DFLTGRP(STCGROUP) OWNER(STCADMIN) NOPASSWORD
ALTUSER JES DFLTGRP(STCGROUP) OWNER(STCADMIN) NOPASSWORD NOPHRASE
```

If you do not specify **NOPASSWORD** for a user ID that is assigned to a started procedure, you should specify a password and change the password periodically. If you do not specify a password and do not specify **NOPASSWORD**, RACF uses the default group name as the password. Anyone who knows this user ID and password combination can gain access to any resource that the started procedure can access.

Revoked user ID: If the associated user ID is revoked for any reason, the started procedure might have problems allocating new SMS-managed data sets, submitting batch jobs, and obtaining printed output.

Undefined User IDs

A started procedure runs as an undefined user if the following items are true:

1. It is run without associating its name with a RACF defined user ID and group name.
2. The user or group is not defined.
3. The user is not connected to the group.

A started procedure running as an undefined user can access RACF-protected resources if the universal access authority for the resource is sufficient to allow the requested operation. However, if a started procedure requires access at a higher level than universal access, you must associate the started procedure with a RACF defined user ID and group name.

9.13.2 Authorizing access to resources

A started procedure can gain access to RACF-protected resources in the following ways:

- ▶ By the user ID or group name that is assigned for any other user of the system (for example, universal access, entry and access list, and OPERATIONS).
- ▶ By having the privileged attribute, which allows the started procedure to pass all authorization checking (unless the CSA or PRIVATE operand is specified on the RACROUTE request). No installation exits are called, no SMF records are generated, and no statistics are updated. (Bypassing authorization checking includes bypassing the checks for security classification of users and data.)
- ▶ By having the trusted attribute, which means the same as privileged, except that you can request an audit by using the **SETROPTS LOGOPTIONS** command.

9.13.3 Setting up the STARTED class

With the STARTED class, you do not need to change code or initiate an IPL of the system to add or modify RACF identities for started procedures. You can modify the security definitions for started procedures dynamically by using the **RDEFINE**, **RALTER**, and **RLIST** commands. For more information about these commands, see z/OS Security Server RACF Command Language Reference, SA22-7687-16, found at:

http://www.ibm.com/support/knowledgecenter/#/SSLTBW_1.13.0/com.ibm.zos.r13.icha400/toc.htm

In effect, the STARTED class provides a dynamic started procedures table.

To set up the STARTED class, enter the commands that are shown in Example 9-1.

Example 9-1 Set up the STARTED class

```
SETROPTS GENERIC(STARTED)

RDEFINE STARTED JES2.* UACC(NONE)
  STDATA(USER(JES2) GROUP(STCGROUP) TRUSTED(YES))
RDEFINE STARTED ** UACC(NONE)
  STDATA(USER(=MEMBER) GROUP(STCGROUP) TRACE(YES))

SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
  or, if the STARTED class is already in use:
SETROPTS RACLIST(STARTED) REFRESH
```

Defining profile data

Profiles in the STARTED class include the STDATA segment, which contains fields for user ID, group name, trusted flag, privileged flag, and trace flag:

- ▶ The user ID can be a RACF user ID or the character string =MEMBER, which indicates that the member name is used as the user ID.
- ▶ The group name can be a RACF group name or the character string =MEMBER, which indicates that the member name is used as the group name.
- ▶ If tracing is specified, RACF issues operator message IRR812I during RACROUTE REQUEST=VERIFY or VERIFYX to indicate which profile is used. This message can be used during diagnosis of security problems with started procedures to determine which profile was used for a particular started procedure.

RACF performs partial diagnosis when creating the STDATA segment to help you define profiles that work correctly. For example, RACF verifies that a specified user ID is connected to the group name, if specified.

Attention: Be sure to specify a group name (not =MEMBER) as the GROUP value of the STDATA segment, if both of the following items are true:

- ▶ The profile name contains generic characters (*, %, or &).
- ▶ The USER value of the STDATA segment is the character string =MEMBER.

If you do not specify a group name, a new started procedure or job can be assigned on execution to a user ID that matches an existing user ID on your system. Consider defining a special group (for example, STCGROUP) for started procedures and job user IDs, and using this group name as the GROUP value of the STDATA segment.

In addition, be careful which libraries your started procedures come from and do not let your users update them. For information about specifying procedure libraries, see *z/OS JES3 Customization*, SA22-7542-10, found at:

http://www.ibm.com/support/knowledgecenter/#!/SSLTBW_1.13.0/com.ibm.zos.r13.iatb100/toc.htm

Specifying STARTED class profile names

You can start jobs and procedures by using the **START** command. The **START** command specifies the member name to start and, optionally, the job name to use.

For started procedure (job) address spaces, resource names in the STARTED class are of the form member.job, where:

member	The 1 - 8 character name of a member of a partitioned data set that contains the source JCL for the task to be started. The member can be a job or a started procedure.
job	The name identifying the procedure to be started. If the START command does not specify a job name, and the member does not contain a JOB statement to supply a job name, the system uses the member name as the job name when constructing the resource name for STARTED class processing.

For system address spaces, resource names are of the form job.job, or (if your system programmer used the JOBSpace option of the ATTR parameter of the ASCRE macro to create the address space), member.job. To determine which form to use for a procedure that is associated with a system address space, see the documentation for the application or consult the programmer. For programming details about creating address spaces, see "Creating address spaces" in the *z/OS MVS Programming: Extended Addressability Guide*, SA23-1394.

For each sample **START** command that is run for a started procedure (job) address space, Table 9-6 on page 273 shows the resource name that is used for STARTED class processing, and a list of names for STARTED class profiles that can be defined for each STARTED class resource.

Table 9-6 Sample profile names for STARTED class resources

START command	STARTED class resource name	STARTED class profile name
S CICS	CICS.CICS	CICS.CICS
		CICS.*
		CICS.**
S CICSP	CICSP.CICSP	CICSP.CICSP
		CICSP.*
		CICSP.**
		CICS*.**
		CICS*
S CICST	CICST.CICST	CICST.CICST
		CICST.*
		CICST.**
		CICS*.**
		CICS*
S IMS, JOBNAME=IMSPROD	IMS.IMSPROD	IMS.IMSPROD
		IMS.IMSP*
		IMS.*
		IMS.**
S IMS, JOBNAME=IMSTEST	IMS.IMSTEST	IMS.IMSTEST
		IMS.IMST*
		IMS.*
		IMS.**

9.13.4 Using the started procedures table (ICHRIN03)

Your RACF system programmer can use the started procedures table (ICHRIN03) to associate the names of started procedures with specific RACF user IDs and group names.

The started procedures table also can contain a generic entry that assigns a user ID or group name to any started task that does not have a matching entry in the table. In this case, the table can specify that the started task name should be used as the user ID or group name, or it can assign a specific user ID or group name to the started task. You should ensure that the generic entry, if used, assigns a generic user ID or group name (for example, STCGROUP).

To modify the security definitions for started procedures by using the started procedures table, complete the following steps:

1. Edit the started procedures table.
2. Assemble and link edit the updated table.
3. Initiate an IPL of the system.

For information about how to code the started procedures replaceable module, and for a complete description of the started procedures table (ICHRIN03), see the *z/OS Security Server RACF System Programmer's Guide*, SA23-2287.

9.13.5 Started procedure considerations

Here are some things to consider when you use started procedures:

- ▶ Even if your installation uses the STARTED class, you must have a started procedures table (ICHRIN03). RACF cannot be initialized if ICHRIN03 is not present. A dummy ICHRIN03 is shipped with and installed by RACF. If you use the STARTED class, you should leave your existing ICHRIN03 in place, in case, for example, someone unintentionally deactivates the STARTED class.
- ▶ For installations that have an existing started procedures table (ICHRIN03) and you want to use the STARTED class, a sample REXX exec is provided in member ICHSPTCV in SYS1.SAMPLIB to process the output of ICHDSM00 and build RDEFINE commands to duplicate an existing started procedures table.
- ▶ To make sure that critical system tasks (those marked TRUSTED or PRIVILEGED in ICHRIN03) start successfully, define specific STARTED profiles for them in the STARTED class.
- ▶ (Guideline) Assign the TRUSTED attribute when one of the following conditions applies:
 - The started procedure or address space creates or accesses a wide variety of unpredictably named data sets within your installation.
 - Insufficient authority to an accessed resource might risk an unsuccessful IPL or other system problem.

For a list of required and optional candidates for the TRUSTED attribute, see “Assigning the RACF TRUSTED attribute” in the *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

- ▶ When the STARTED class is active, RACF uses it before using the started procedures table (ICHRIN03). A generic profile such as ** or *.* with a valid STDATA segment overrides all the entries in ICHRIN03.
- ▶ To make sure that RACF uses the STARTED class, verify that all **START** commands have a matching profile with an STDATA segment that assigns a user ID. To do this task, complete the following steps:
 - a. Define an appropriate generic profile that matches all possible **START** commands (for example, ** or *.*).
 - b. Specify =MEMBER or a user ID of limited privileges.
 - c. Specify a group name, if you have specified =MEMBER as the USER value.

This approach ensures that, for any **START** command, there is always a matching profile with an STDATA segment that assigns a user ID. In addition, using this approach avoids the following situations, which cause RACF to use ICHRIN03 to process the **START** command:

- There is no matching profile.
- There is a matching profile, but it does not have an STDATA segment.
- There is a matching profile with an STDATA segment, but no user ID is specified.
- There is a matching profile with an STDATA segment and no user ID is specified, but the assigned user ID matches an existing user ID on your system.

- ▶ When RACROUTE REQUEST=VERIFY or VERIFYX is issued with a started procedure name, RACF checks to see whether the STARTED class is active. If it is active, RACF uses the STARTED class to determine the user ID, group name, trusted flag, and privileged flag to use. If the STARTED class is not active, RACF uses the started procedures table (ICHRIN03). RACF also uses the started procedures table, and issues message IRR813I or IRR814I if the STARTED class is active but one of the following situations occurs:
 - RACF cannot find a matching profile in the STARTED class.
 - RACF finds a matching profile but the profile does not assign a user ID.

9.14 Special procedures for privileged users

Privileged user is a wide term. We use this term here to mean anyone in the organization who has some privilege that is bestowed on the user ID that they use. Here are some examples:

- ▶ The RACF administrator, who has the system SPECIAL attribute
- ▶ The storage administrator, who has the system OPERATIONS attribute, or has access to various STGADMIN resources in the FACILITY class
- ▶ The DB2 administrator, who has access to system functions in the DB2 subsystem
- ▶ The devolved RACF administrator, who has the group-SPECIAL attribute for a division of users
- ▶ A devolved RACF administrator, who has the CLAUTH attribute for a particular class

There will be other users. The systems programmers also might be included in this list if they have update access to system configuration libraries and settings. Such users must use greater care than other users when working with the configurations of their RACF accounts. They are trusted to look after some aspect of the security of the enterprise and they must ensure that the users, resources, and facilities that they use are worthy of the same trust.

Particular care must be taken when running any process that is supplied by a potentially untrusted source.

Consider the following example:

Freda is a RACF administrator who has the SPECIAL attribute. She receives requests and acts on them if the authorization requirements are met.

Freda uses some REXX routines that were developed for her to simplify tasks, such as adding new TSO users. For each new TSO user, several actions must take place, and the REXX routine that is called NEWUSER works with some ISPF panels to make the task faster and less error-prone. These routines are stored in SECGROUP.REXXLIB and the library is carefully controlled so that only the security administrators can update it.

Freda uses the standard TSO logon procedure for her company. This procedure includes using a library of REXX routines that the systems programmers, and some senior developers, are authorized to update. However, the library GROUP.REXXLIB is at the top of the list.

This is a dangerous situation for Freda. Any user who has UPDATE authority to the library containing those REXX programs, *or any library above it in the concatenation*, can cause her to run a different REXX program without Freda being aware of it.

The library list is as follows.

```
//SYSPROC DD DISP=SHR,DSN=GROUP.REXXLIB  
// DD DISP=SHR,DSN=ISP.SISPCLIB  
// DD DISP=SHR,DSN=SECGROUP.REXXLIB
```

Anyone who has update access to GROUP.REXXLIB can copy the REXX program NEWUSER from SECGROUP.REXXLIB and then update it to perform a command and store it in GROUP.REXXLIB. The command might be as simple as to grant the perpetrator the SPECIAL attribute. The command is run without notifying Freda.

Freda is running a program but is not in control of her environment. The replacement REXX routine is effectively a Trojan horse.

The solution here is to ensure that the SYSPROC concatenation of libraries is revised. The security administrator group should not trust a library that is available to too many people who can perform updates.

Similarly, Freda should not run any REXX program of which she has insufficient knowledge. She should run only processes that are trusted.

In this case, it is preferred practice to have a separate LOGON procedure for the security administration personnel.

Preferred practice: Ensure that privileged users have the appropriate LOGON procedures for their role. These procedures should be designed so that it is not possible for Trojan horses to be introduced.

Preferred practice: Keep the logon procedures for TSO users in a library separate from other procedures. This library should be carefully controlled and monitored for changes.

Everyone who has a special privilege has a responsibility to ensure that they are not running code that is untrusted. This takes much careful design.

9.15 Multiple LPARs and shared DASD

Many installations run multiple LPARs on System z servers. This is a normal capability and is one of the strengths of System z. The design and implementation of the LPAR microcode is accredited as having the capability to achieve system separation at the EAL5+ level of the Common Criteria.

Many organizations have a setup similar to what is shown in Figure 9-11 on page 277.

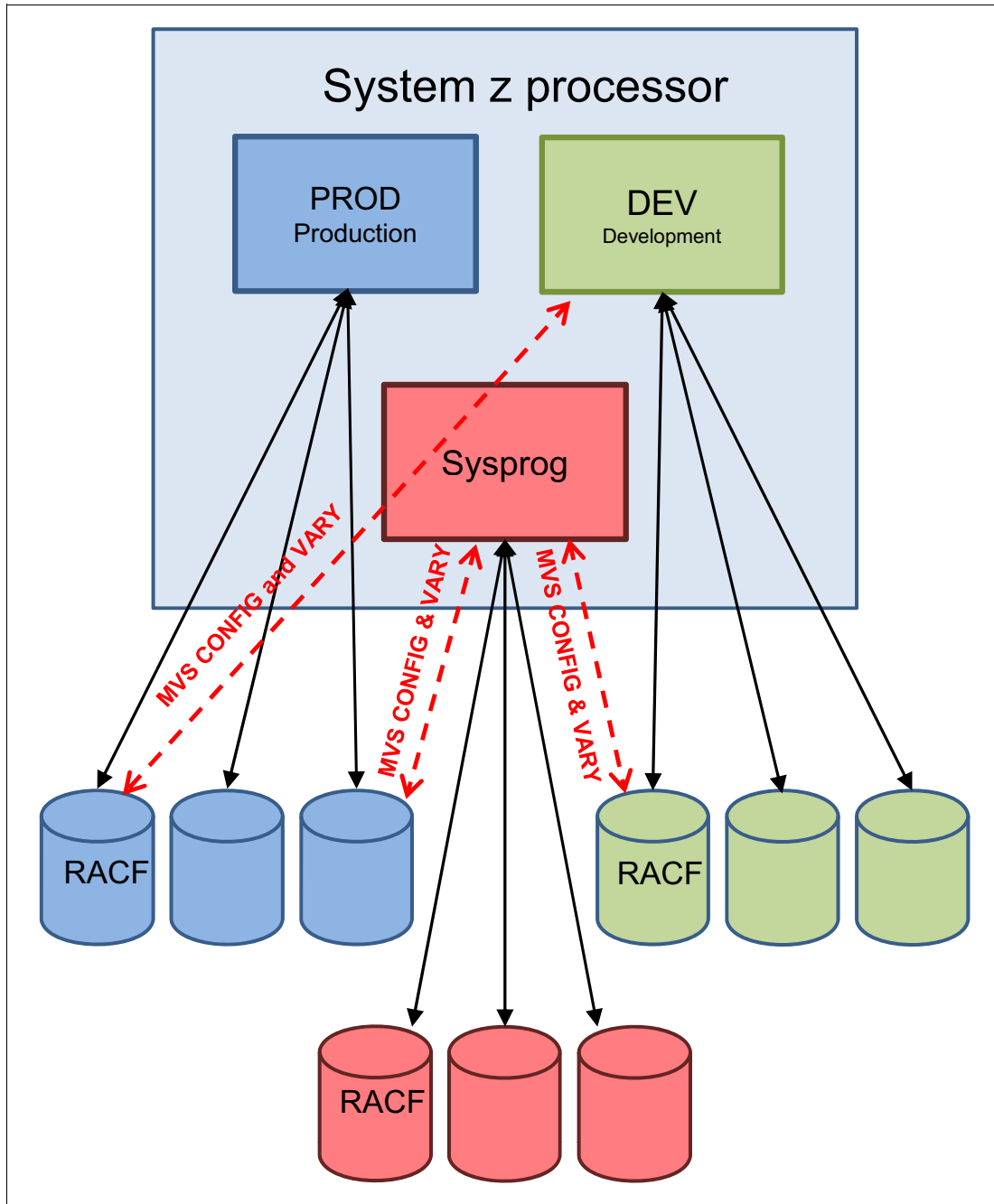


Figure 9-11 Multiple LPARs and shared DASD

A single DS8000 type device is used to host all the DASDs in use by all three LPARs. However, the three LPARs have distinct RACF databases, and distinct security policies that are administered by different groups. In particular, the system programming group runs a relatively open system and grants access to many groups to perform testing. They use this system for building new z/OS releases for production. The SMP/E libraries are held on the systems programming LPAR.

Channel paths are defined in the IOCP from each system to all of the DASDs. This is done for recovery reasons. If the production system is down and unable to restart, then the volumes for that system can be varied online to one of the other systems and repaired from there.

The number of exposures that are associated with such a system configuration is large:

- ▶ If production volumes can be varied online to the SYSP system, then anyone on that *relaxed security* system can gain access to the data.
- ▶ If those production volumes can be varied online to the SYSP system, they can be updated, including APF-authorized libraries.
- ▶ Programs exist to manipulate RACF databases that are offline. Such a program might be used to manipulate the production RACF databases.
- ▶ The new operating system is not protected in the SYSP environment, and it is relatively simple to slip in an undocumented authorized program into a system library.

The management of a multiple-LPAR installation in this manner might have been acceptable years ago, but this is no longer the case.

Preferred practice: Do not allow systems with separate RACF databases to share a DASD. Do not define paths that allow a DASD to be varied online from a system that has a different security database.

The confusion here is between recoverability and security. Other methods can be used to ensure that these LPARs are recoverable. Those options should be chosen.

9.16 Conclusion

z/OS security design and management is a complex matter. Implementing a security design to meet the appropriate risk profile for your organization requires careful planning.

Undertaking such a task can be simplified by using the appropriate tools to understand your existing security profile, design changes to that security profile, and implement those changes in a risk-free manner.

One such set of tools is the IBM Security zSecure products.

Whether such tools are used or if the work is undertaken manually, the effort can be considerable. However, doing so reduces risks greatly.



IBM z/VM security

This chapter describes z/VM base protection and some software tools and hardware features that complement this environment. This chapter examines the security controls and features that are available to the z/VM operating system. It offers suggestions about what settings are most related to security and integrity.

Because z/VM is simultaneously a hypervisor and an operating system, its security considerations are different from a traditional operating system such as z/OS.

Protection against attempts to breach the security of your system and against inadvertently compromising the integrity of your system and data should be part of the planning and administration for your z/VM system. z/VM has built-in facilities and support for products to aid you in this task.

This chapter covers the following topics:

- ▶ Overview of z/VM security
- ▶ Securing the virtual machine
- ▶ Hypervisor security
- ▶ Secure connectivity
- ▶ External Security Managers
- ▶ Features and extensions to z/VM
- ▶ Preferred practices in z/VM security

10.1 Overview of z/VM security

The core of z/VM is the Control Program (CP), which creates and maintains virtual environments for virtual machines (VMs) (guests). You can use virtualization of the machine to accomplish the following goals:

- ▶ Manage many servers by using universal management tools
- ▶ Reduce management costs
- ▶ Maximize the usage of existing hardware

Like all System z Operating Systems, z/VM can run alone or share the mainframe with other logical partitions (LPARs), which may host another z/VM, Linux for System z, or z/OS instance. Within a single z/VM instance, though, you can virtualize multiple Linux for System z guests, z/OS guests, z/VSE guests, or combinations of these guests.

z/VM can either virtualize the same System z server on which it is running or can emulate hardware features that are not necessarily installed on that particular model of server. It can also provide a customized environment for each guest. z/VM provides System z features to the guest operating systems transparently and simultaneously, without having a physical server per guest. At the same time, z/VM can isolate each guest operating system and handle access to real devices as needed.

The number of guests that z/VM can operate concurrently is limited only by the amount of resources that are available to the System z. This is in contrast to LPARs, which have a fixed limit to the number of LPARs that depend on the System z that is used. Guests of a z/VM host run within user IDs, or just ID for short. Typically, people logging on to z/VM are called users, and automated user IDs are known as service machines and run in a disconnected state (which means without a console or display terminal attached).

The RACF Security Server feature for z/VM (RACF/VM) complements and extends basic z/VM security features while providing more granularity and accountability of all accesses and operation events. Its access control includes user verification, resource authorization, and logging capabilities. RACF is flexible and its impact on z/VM operation is barely noticeable.

The z/VM Lightweight Directory Access Protocol (LDAP) server is based on a client/server model that provides client access to an LDAP server. An LDAP directory provides an easy way to maintain directory information in a central location for storage, update, retrieval, and exchange.

The Conversational Monitor System (CMS) is a single-user operating environment that runs only as a z/VM guest. It has only the security features that are needed for a single-user system, and for resource access control relies on the control of the z/VM.

IBM z/VM Directory Maintenance Facility (DirMaint™) uses standard z/VM facilities to isolate users from each other and from the system and protect files from outside interference or contamination.

The z/VM provides a way to share the hardware cryptographic resources among the guest operating systems. All z/VM guests can have unlimited access to all available cryptographic accelerators, including CP Assist Cryptographic Function (CPACF) and the CEX2C when configured as a pure accelerator (CEX2A).

zSeries networking facilities offer a unique security advantage. Queued direct input/output (QDIO) architecture uses direct memory access (DMA) to transfer network data from the address space of the sender to the address space of the receiver.

The z/VM Virtual Switch (VSWITCH) offers the ability to connect a virtual network directly to an external physical LAN, without requiring a z/VM guest to act as a router.

When considering the security and integrity of z/VM, there are two primary factors to consider: guest isolation and hypervisor security.

10.1.1 The principles of guest isolation

z/VM is designed to host concurrent workloads in a virtualized environment. These workloads are competing for real resources (CPU, memory, disk access, and others), so the hypervisor must keep guest workloads separate from one another. This principle is called *guest isolation*. Guest isolation prevents guests from threatening the integrity of either another guest or the hypervisor itself.

Guest isolation in z/VM begins at the instruction level. The Start Interpretive Execution Instruction (SIE) on System z creates the environment for VMs. Most of the instructions are run by the hardware, with little z/VM intervention. The SIE is implemented on the System z by the Interpretive Execution Facility. The z/VM CP receives an interruption when the hardware detects conditions such as timer expiration, an unassisted input/output operation (I/O), instructions that require some privileges, and program interrupts. SIE also handles the use of region, segment, and page tables that were set up previously by the z/VM CP for the user ID. SIE instruction is available only on System z, and it is used by z/VM to decrease CP impact.

10.1.2 The principles of hypervisor security

Resources that can be shared between guests are CPU, real memory, disk volumes, network adapters, printers, and devices that are specific to System z, such as cryptographic cards. It also supports the latest storage area network (SAN) and virtual tape library systems. For example, in the case of disk storage, z/VM can partition a disk volume and assign portions to each guest. Control of read-only and read/write access or none at all is at the discretion of the z/VM administrator.

Hypervisor security is the extension of guest isolation in that it is the set of policies and procedures that is implemented within the hypervisor layer itself to control and audit access to these resources.

10.2 Securing the virtual machine

The principle of guest isolation only separates one VM from another. To make a VM more secure, its scope of responsibility must be defined. A VM should have only the authority to perform the functions that are required by hosted workload. This section describes the following mechanisms through which a VM's role is defined:

- ▶ Privilege classes
- ▶ Assigning a new privilege class to a VM
- ▶ LOGONBY
- ▶ The COMMAND directory statement

10.2.1 Privilege classes

Every VM that is defined in the z/VM User Directory has one or more privilege classes that are defined. A privilege class represents a job or role that is associated with a VM workload. They are used in z/VM to implement Role-Based Access Control (RBAC).

There are seven privilege classes that are defined by default in z/VM, which are represented by the letters A - G. These letters represent specific roles in the z/VM operating environment, ranging from System Operator to General User. A *privileged user* is any VM that has greater than a Class G authority on the system.

The system administrator can also create new privilege classes that meet local security policy and the roles that are defined for guest workload. These classes can be represented by letters I - Z, or the numbers 1 - 6.

To determine what CP commands and diagnostic instructions to which a VM has access, run **QUERY COMMANDS**.

10.2.2 Assigning a new privilege class to a VM

The **SET PRIVCLASS** command can, if enabled for the hypervisor, be used to modify a VM's operating context. For most VMs, this removes only a privilege class that has been granted to a VM in the User Directory. However, a version of this command exists for Class C users, which can give that VM any privilege class. This version of the command is "escalation of privilege" writ large and should be controlled by RACF/VM. Alternatively, it can be removed from Class C or the feature can be disabled in z/VM.

10.2.3 LOGONBY

Some VMs, such as **OPERATOR**, are highly privileged by nature. Yet it may be the case that your shop has multiple administrators who required access to a privileged VM. This introduces a problem: If multiple individuals share the logon credentials for a privileged guest, who is held accountable if something is taken offline, damaged, or stolen? Sharing credentials introduces a problem. It is solvable by **LOGONBY**.

The **LOGONBY** statement in z/VM is part of a VM definition in the z/VM User Directory. It lists up to eight other VMs (commonly associated with human users, rather than another service VM). When logging on to the source VM, the credentials of another VM is used instead. For example, if you specify on the OPERATOR VM the statement **LOGONBY BWHUGEN LENDB AXELB**, and then the following statement was entered on the z/VM LOGON screen (**LOGON OPERATOR BY BWHUGEN**), then the password or password phrase for VM BWHUGEN is required for access. So, auditing records note that BWHUGEN logged on to the OPERATOR VM, providing accountability for access to privileged operations.

A VM with **LOGONBY** should have its password set to 'LBYONLY'. If OPERATOR's password remains intact, then it might still be accessed without providing alternative credentials.

This function is extended in RACF/VM through the SURROGAT class of operations.

10.2.4 The **COMMAND** directory statement

In some instances, there might be configuration requirements for a VM that require the running of privileged commands at the start of a VM. Although there might be a temptation to adjust privilege classes and add these commands to a PROFILE EXEC, the better solution is to use the **COMMAND** statement.

The **COMMAND** statement is part of a VM's User Directory entry. This statement, which supports up to 255 characters, can run a privileged command after the instantiation of a VM but before the guest has formally undergone an IPL. This bypasses the need to give a VM a specific clearance level while allowing flexibility in configuration.

For example, the guest LINUX01 might need to **ATTACH** a specific device to run on its workload. The ability to **ATTACH** a real device is a Privilege Class B command, and it is highly restricted and controlled. Rather than grant the VM access to Privilege Class B, or even move **ATTACH** to a second user-defined privilege class, the User Directory entry can be added as follows:

```
COMMAND VARY ON 1234  
COMMAND ATTACH 1234 TO &USERID AS 4567
```

These commands allow the hypervisor itself to configure the guest without granting excess privilege.

Although the **COMMAND** statement is limited to 255 characters, multiple statements can exist for a single VM definition.

10.3 Hypervisor security

The z/VM System Configuration file is to the hypervisor what the User Directory is to a VM: a flat file that defines operational and security context.

10.3.1 Default settings

The z/VM System Configuration file provides statements that allow for the basic defaults of the hypervisor to be set to values that match local security policy. This includes a privilege class for VMs (G by default) and a privilege class for System Operators (A by default). If the System Operator function (as represented by the OPERATOR VM) must be controlled more thoroughly, this latter setting can be changed to a user-defined privilege class.

Other pertinent security settings include TDISK_CLEAR (which purges temporary disk space when the space is released back to the hypervisor), SET_PRIVCLASS (which enables or disables the command that is described in 10.2.2, “Assigning a new privilege class to a VM” on page 282) and the sharing of real devices (which is pertinent any time the RACF Database must be shared by more than one hypervisor, such as in a Single System Image (SSI) cluster).

10.3.2 Command restructuring

The **MODIFY COMMAND** and **MODIFY DIAGNOSE** commands can be used to remap CP commands and diagnose instructions from their default privilege classes into one or more alternatives. This allows the security administrator to remove potentially dangerous commands from use before a workload is deployed.

Here are few examples:

► **MODIFY COMMAND SHUTDOWN PRIVCLASS S**

In this example, the **SHUTDOWN** command is removed from Privilege Class A and placed into Privilege Class S. Because Class A is an IBM default, there are IBM-defined service VMs that have access to this command (which, as the name implies, shuts down the entire z/VM LPAR). Because this command is so disruptive, your security policy might recommend that it be restricted to particular VMs with the Class S role.

► **MODIFY COM XAUTOLOG IBMCLASS A PRIVCLASS X**

Some commands, such as **XAUTOLOG**, come in multiple variations; the operations and implications change with the privilege class. **XAUTOLOG** Class A allows one VM to start another. If your security policy demands that this be removed from Class A and only Class A, the **IBMCLASS** subcommand provides this option without disrupting other versions of **XAUTOLOG** (Classes B and G).

► **MODIFY CMD QUERY SUBCMD NAMES IBMCLASS G PRIVCLASS Z**

The **CP QUERY** and **CP SET** commands have many subcommands; they can be modified by a subcommand, rather than at the **QUERY** or **SET** level. In this case, **QUERY NAMES** displays the VM name of every running machine on the system. This might be highly privileged information, and certainly is not necessary for a Class G user, so the example moves that command from Class G (and Class G only) into Class Z instead.

MODIFY can be used to build new privilege classes that represent roles that are pertinent to your installation. This is useful from a security perspective: As new versions of z/VM are released, the default IBM privilege classes are updated with commands that might be in accordance with the scope of responsibility of your workloads. User-defined privilege classes are not updated in this fashion, so a new function can be analyzed before VMs are upgraded with these newer capacities.

10.3.3 Observers and secondary users

You occasionally might need an administrator or service VM to monitor operations inside another VM (such as OPERATOR). Rather than conferring logon access to the target machine, privileged status can be conveyed onto a “viewer” VM.

The **OBSERVER** authority sends the events on a VM's console to the VM with that authority. So, if you run **SET OBSERVER** for BWHUGEN to view OPERATOR, then all of OPERATOR's console output appears within the BWHUGEN VM. That said, the **OBSERVER** status does not allow BWHUGEN to send commands to the OPERATOR console. This is useful for monitoring and automating responses to system events, or to provide a record of operations on a highly privileged VM.

The more powerful format, **SECUSER**, allows one VM to see the console output of another and grants the authority to send commands to that console. This might represent a security issue if used incorrectly because a source VM's privilege class might differ (even far exceed) that of the Secondary User. It should be used with a great degree of caution.

10.3.4 Virtualized hardware cryptographic features

z/VM provides data encryption services for guest workload rather than for the hypervisor layer itself. It does this through two particular mechanisms: support for hardware cryptographic features and for IBM tape storage encryption.

IBM System z hardware cryptography on z/VM

The z/VM provides a way to share the hardware cryptographic resources among the guest operating systems. Any z/VM guest can use the CPACF if this no-cost feature is enabled on the System z Support Element (SE). This on-chip cryptographic offload allows guests to drive symmetric key operations (DES, Triple-DES, AES, and SHA hashing functions) without increasing CPU usage. The z/VM SSL-TLS Server uses CPACF automatically if it is available.

Additionally, z/VM supports guest access to the CryptoExpress features. These hardware features, orderable at a cost with the System z hardware, allow for asymmetric key operation acceleration and dedicated coprocessing for special guest cryptographic operations. At the time of writing, the most recent version of CryptoExpress is CryptoExpress 4S.

z/VM guest access to CryptoExpress domains is managed in the z/VM user directory through the **CRYPTO** statement. A guest may either have dedicated access to one or more domains on an Adjunct Processor (AP) queue, or it may have shared access to hypervisor-owned domains. If a guest requires its own Master Key access, or requires Secure Key operations, then dedicated access is pertinent from a security perspective. If a guest must offload only basic accelerator operations, then shared access is sufficient.

IBM tape storage encryption

z/VM provides support for the encryption of data (hardware encryption) that is written to auxiliary storage media by IBM tape drives (for example, IBM Virtualization Engine TS77nn Series). z/VM CMS support cannot enable hardware encryption. Therefore, CMS applications or licensed programs (for example, IBM Backup and Restore, or Computer Associates VM:Backup) that are used to write data to tape drives cannot directly enable encryption.

However, z/VM includes the **ATTACH** and **SET RDEVICE** commands, which provide for the enabling of the hardware encryption on the tape drive before the tape drive is dedicated to the server that is running the CMS application or licensed program.

To use this capability, the tape drive control unit must be able to communicate with an encryption key manager (EKM) and an associated keystore. z/VM expects the EKM communication to take place through TCP/IP as an *out-of-band* connection. The z/VM DASD Dump Restore (DDR) utility provides the **KEY** option for the output statement to enable directly the output tape drive for hardware encryption.

10.4 Secure connectivity

System z networking facilities are an intrinsic part of the platform and offer a unique security advantage. QDIO architecture uses DMA to transfer network data from the address space of the sender to the address space of the receiver. For HiperSockets networks, this means that network traffic cannot be “sniffed” by third parties (other z/VM guests or other LPARs). For OSA-Express, network traffic is moved directly to and from the adapter to the address space of the send or receiver.

z/VM uses several different communication technologies.

10.4.1 TCP/IP

The z/VM TCP/IP stack offers modern point-to-point communication over standardized transmission protocols. In z/VM, the TCP/IP feature is part of the base product and is composed of a suite of Service Virtual Machines (SVMs). Each one offers different functions that are related to its specific protocol of support. Here are some examples:

- ▶ TCPIP, for basic connectivity to the hypervisor through the Transmission Control Protocol / Internet Protocol (TCP/IP)
- ▶ FTPSERVE, for File Transmission Protocol (FTP) support
- ▶ LDAPSRV, for the LDAP
- ▶ MPROUTE, for communications routing

Although each part of the TCP/IP suite has its own set of security controls, most offer exits that connect them to RACF/VM. This allows (for example) user ID processing for FTP connections to be verified by an External Security Manager (ESM) (and the appropriate auditing records to be cut, as a result). Additionally, the LDAP server can be connected to RACF/VM directly, and uses RACF definitions for its security policy database. This allows for a broader integration of security controls and Identity Management within the hypervisor.

The most security-relevant part of the TCP/IP suite is the SSL-TLS Server. This pool of VMs is associated with one TCP/IP stack. It encrypts and decrypts traffic being transmitted to and from the hypervisor layer. This is important: Any end-to-end security policy includes hypervisor connectivity as a point of concern. After all, if someone can control your hypervisor, the chance exists that they can control the guests within.

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols are managed through the TCPMAINT VM. (TCPMAINT may be configured for **LOGONBY** access). Specific encryption algorithms and levels of these protocols can be allowed or disallowed, in accordance with your security policy. These settings cannot be changed dynamically, and any incoming connection that does not meet your security policy is rejected long before a LOGON screen can be presented.

The other side of SSL and TLS is digital certificate management. Any connection using SSL or TLS requires a digital certificate to connect. The z/VM system presents a server certificate to confirm its own identity; for dynamic TN3270 connections, a client certificate may be requested (or required) as well. At its most stringent, mutual validation of these certificates allows both the server and the client to come to a mutual understanding of trust before the possibility of LOGON enters the picture.

Certificate management in z/VM is handled through the GKSADMIN VM.

10.4.2 Guest LANs and Virtual Switches

In addition to TCP/IP, a local area network (LAN) may be defined within the hypervisor itself. This allows for guest communication to pass back and forth without ever leaving the hypervisor layer. The initial support in z/VM was called the Guest LAN feature. A global LAN segment is defined. Each VM in the Guest LAN needs to define only a virtual network interface card (vNIC) and connect to the Guest LAN segment. A Guest LAN can connect to a physical LAN by routing through an OSA-Express adapter and then to a LAN switch; however, physical connectivity is not required for operation.

As a preferred practice, disable TRANSIENT Guest LAN creation. This function allows General User VMs (Class G) to create their own dynamic Guest LANs; this might represent a breach of information flow control.

The z/VM VSWITCH offers a more securable alternative to Guest LAN. VSWITCH grants the ability to connect a virtual network directly to an external physical LAN without requiring a z/VM guest to act as a router. Instead, routing falls to the hypervisor layer itself. This provides a performance benefit: Virtual guests acting as routers consume processor cycles to move packets between the virtual and physical networks, and routing adds latency to network traffic. Guests that are connected to a VSWITCH can be defined in the subnet of the external network to which they are connected.

VSWITCH support can be in Layer 3 (TCP/IP) format, but it is more often used in Layer 2 (Ethernet) mode instead.

In addition, a VSWITCH can participate in a IEEE 802.1Q VLAN. VLANs offer the following management and security improvements over physical network topologies:

- ▶ VLAN subnets (and the hosts on those VLANs) can be defined according to the logical organization of network traffic. Broadcast domains are defined by the VLAN and not limited to the physical LAN.
- ▶ VLAN administration can simplify network management. For example, when moving hosts on a VLAN, the network configuration can be changed at the switch (instead of each individual host).
- ▶ Virtual switches also can be configured to support *uplink ports* (an isolated tunnel from the physical LAN to the guest, which can allow for the funneling of traffic for inspection) and *link aggregation* (which can combine multiple OSA ports into a logical group, for greater throughput). These commands are highly relevant to security, as they pertain directly to network access for VMs.
- ▶ VLANs can further be controlled and audited by RACF/VM.

For more information about Guest LANs and VSWITCH technology, see the *z/VM Connectivity Guide*, SC24-6174.

10.4.3 HiperSockets

z/VM supports the use of HiperSockets, which is defined as part of the System z environment. The important point to consider, when using HiperSockets is that it should be treated the same as any other LAN segment, which means access to it might need to be controlled. Not every guest on a certain z/VM LPAR requires access to information on another LPAR on that central processor complex (CPC).

z/VM supports a HiperSockets-VSWITCH bridge. This allows two important functions:

- ▶ Transmission of HiperSockets traffic from one point of origin to cross over customer-controlled networks (for example, IEDN or QDIO CHPIDs) into physical networks or even into another System z machine
- ▶ Control of HiperSockets communications through the VSWITCH security controls

10.5 External Security Managers

ESMs provide a centralized and effective security mechanism for authentication and authorization of users to access system resources.

RACF/VM is the IBM supplied ESM for the z/VM hypervisor layer. An ESM is an integral component of virtualization security in modern environments.

10.5.1 Reasons to use an ESM for z/VM

It is critical to deploy an ESM such as RACF/VM for following reasons:

- ▶ It allows greater control of CP commands.
- ▶ It provides much more comprehensive auditing than the base hypervisor.
- ▶ It can override base CP security functions.
- ▶ It encrypts VM passwords.
- ▶ It supports password phrases, which are mixed-case passwords 14 - 100 characters in length.
- ▶ It allows for discretionary access controls (DACs), which enable the creation of access control lists (ACLs) to determine who can use which resources.
- ▶ It allows for mandatory access controls (MACS), which enable system-level labeled security that builds impassable walls around system resources, and allows for multi-tenancy in the environment

All of these features and functions are available within a single security interface for the system.

Preferred practices that are related to RACF/VM are described in 10.7.5, “Installing and deploying an ESM” on page 294.

10.5.2 Control of privileged VM commands

Certain z/VM commands are considered particularly security relevant; RACF/VM provides the option of enabling specific ACLs to determine what VMs can run that command successfully.

For example, the **SHUTDOWN** command, which is used by security auditors of z/VM, is a Class A command. Class A is not common in a z/VM system, and a guest workload does not often contain it; however, several IBM supplied service VMs have access to it, and some users might have Secondary User access to one of those. Are you certain you that have closed all the loopholes?

RACF/VM mitigates this risk by enforcing a policy everywhere within the hypervisor. Enabling the VMCMD resource class and creating a VMCMD profile for the **SHUTDOWN** command with a Universal Access (UACC) of NONE means that the command is disabled by default. VMs can be added to the access list through PERMIT. Suddenly, **SHUTDOWN** is constrained to a small subset of possible users, and it can be properly audited as well.

10.5.3 Auditing of security operations

RACF/VM allows for the auditing of any security-relevant event on the system. This might refer to CP commands or diagnostic instructions; it also can refer to the **COMMAND** statement in the z/VM user directory, IUCV-related traffic within the hypervisor, or even SSI-specific events such as the VMRELOCATE of a VM from one member to another.

Although auditing all of them is possible, many of them are disabled by default. Running **RAC SETEVENT LIST** displays the current control and auditing settings for your RACFVM deployment. Auditing is enabled through VMXEVENT profiles, and can be configured to cut auditing records in the case of successes, or failures, or all possible events.

Auditing is managed in RACF/VM by the RACFVM VM. Periodically, the VM cuts a set of records to a secondary VM called RACFSMF, which maintains records locally until the command is run to print the SMF records to disk. Depending upon how aggressive your auditing settings happen to be, SMF records might overwhelm the RACFSMF storage space that is used to hold SMF records in advance of minidisk storage.

In the SMF CONTROL file, a setting that is called SEVER determines what RACFVM does if this “flooding” occurs. If SEVER=NO, then RACF/VM overwrites the “extra” SMF records sitting in memory. This constitutes a loss of security-relevant data. SEVER=YES means that if SMF records are not being “printed” to minidisk in a regular fashion, and the memory fills up, then RACF/VM halts all processing on the hypervisor until auditing can be restored. This is highly disruptive to system operations, but it ensures that no security-relevant data is lost.

10.5.4 Security zones and multi-tenancy

In addition to discretionary access controls for commands and resources on the system, system-wide labeling can be deployed in RACF/VM. At its most basic level, this allows RACF/VM to *color-code* all resources on the system: *blue* VMs can connect only to *blue* minidisks and *blue* networks, or run commands against other *blue* VMs. An attempt to connect a *blue* machine to a *red* minidisk fails, even if that machine were on the minidisk's ACL.

These MACs are system-wide and cannot be overridden. They change a possible yes into a no; they never turn a no security decision into a yes. Most importantly, they strengthen hypervisor security to prevent the transmission of data between two categories or projects being hosted within a single hypervisor. This concept, referred to as *multi-tenancy*, allows two distinct zones to be created within the hypervisor without the possibility of contamination or scope creep, even in cases where users in each project are colluding to transmit data inappropriately.

10.6 Features and extensions to z/VM

The core function of z/VM has been enforcing system integrity and guest isolation for many years. Extra features of z/VM enhance the functions and capabilities of the hypervisor layer, and each of these items has its own security considerations and concerns. This section describes some of the following considerations in more detail:

- ▶ DirMaint
- ▶ Single System Image clustering
- ▶ Systems Management APIs

10.6.1 DirMaint

DirMaint is a collection of service VMs that control and manage changes to VM and hypervisor definitions. These VMs serve as a focal point for directory-related operations; because they assume control of the directory, the configuration of DirMaint allows for an additional mechanism to protect VM definitions.

A user of DirMaint has its own privilege classes, intrinsic to DirMaint and separate from CP processing. A privileged VM by CP standards may not necessarily have the capability to run DirMaint commands, and vice versa. Care should be taken, however, not to grant excess privilege through the use of DirMaint.

Directory statements can be added, deleted, or altered by using the DirMaint directory statement-like commands. DirMaint provides automated validation and extent allocation routines to reduce the chance of operator error. The DirMaint uses standard z/VM facilities to isolate users from each other and from the system and protect files from outside interference or contamination.

DirMaint works with other products, each of which might have certain requirements that impact how you configure and install DirMaint. Planning for those requirements from other products is essential to installing DirMaint. Notably, DirMaint supports a DIRMAINT-to-RACF Connector Exit, which allows changes in DirMaint to be automatically reflected in the security policy that is maintained in the RACF/VM database.

The DirMaint functions are performed by two permanently disconnected VMs that are equipped with an automatic restart facility. The DIRMAINT VM owns and manages the directory; the DATAMOVE VM copies and formats CMS minidisks. Users start DATAMOVE functions by submitting commands to the DIRMAINT VM. DirMaint supports load balancing among multiple DATAMOVE machines, up to a maximum limit of 9999 machines. A DATAMOVE machine must be defined for each member of a z/VM SSI cluster.

Each DATAMOVE service machine must be defined to CP, to an ESM if one is installed, and to DIRMAINT. Although it is possible to have the same user ID for service machines on different systems within a cluster, this imposes restrictions on how spool files and SMSGs can be sent. Even with the same user ID, each DATAMOVE service machine requires its own read/write disk space. Therefore, as a preferred practice, each DATAMOVE service machine should have a unique user ID within the cluster.

When DirMaint is used for the deallocation of VM minidisks, this is accomplished by the DirMaint DATAMOVE Server. IBM internal tools, outside vendor packages, or manual procedures that bypass DIRMAINT minidisk routines for the de-allocation of VM minidisks must remove data from the minidisk at the time the minidisk is returned to the DASD pool for subsequent reallocation. This can be accomplished by a single pass overwrite of the entire minidisk.

Make sure that the DATAMOVE VM is logged on whenever z/VM is initialized. Place the CP **XAUTOLOG** command for DATAMOVE in the PROFILE EXEC of the AUTOLOG2 VM. The z/VM **CMS FORMAT** command and the IBM z/VM ICKDSF Facility **CPVOLUME FORMAT** and **TRKFMT** commands can be used to meet this requirement.

Before returning CKD and FBA DASD volumes to the DASD pool for subsequent reallocation, the contents must be made unreadable. This can be accomplished by a single pass overwrite of the entire DASD volume. For CKD DASD Volumes, the IBM z/VM ICKDSF Facility **CPVOLUME FORMAT** and **TRKFMT** commands can be used to meet this requirement. For FBA DASD Volumes, the IBM z/VM ICKDSF Facility **CPVOLUME FORMAT** command can be used to meet this requirement.

For more information about DirMaint, see the *Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6190.

10.6.2 Single System Image clustering

As of z/VM Version 6 Release 2.0, up to four z/VM LPARs can be interconnected in a z/VM SSI cluster. This operating environment allows for greater flexibility and ease of management of z/VM LPARs by consolidating hypervisor and VM definitions in to a single set of source files for member systems. It also enables the hypervisor to manage cross-system resource management (minidisks and networks), and allows a workload to adapt to planned outages by enabling Live Guest Relocation, which is the mobility of Linux guests from one z/VM LPAR to another without terminating guest operations.

Logon to an SSI is controlled throughout the cluster. A VM has the same password (or passphrase) on each system. A VM has the same privilege classes on each system. If a VM already is running on System A, it cannot log on to System B; error messages result, similar to a second attempt to log on to System A.

Connectivity inside an SSI cluster is handled through physical channel to channel adapters. Because communications between systems never leave the formal hardware, there is no need to encrypt traffic between systems.

Each member system has its own TCP/IP stack, including SSL-TLS servers. They can either share a digital certificate database or each maintains their own, depending upon your security policy requirements.

One RACF/VM server exists for each member of the cluster, but the RACF/VM database is shared between all members. This means RACF/VM becomes a focal point for security throughout the cluster, and that it enforces a single security context for the entire SSI.

For more information about SSI, see *CP Planning and Administration*, SC24-6178.

10.6.3 Systems Management APIs

In addition to all the base functions and cost features that are available, z/VM also provides a set of Systems Management APIs (SMAPIs). These APIs are controlled by another set of Service VMs, and a user of the API is enrolled into the service by a privileged administrator on a per-API per-target basis. Not every user of one API can act upon every possible target, and access to any certain API does not grant access to another.

The SMAPI infrastructure interacts with the z/VM CP and a directory maintenance product, such as DirMaint, to facilitate a “whole-hypervisor managed” environment. When using SMAPI and its extensions, the intent is to isolate the system administrator from the z/VM command-line interface. Although guest isolation and hypervisor security remain as strong as ever, especially when RACF/VM is there to enforce the boundaries and audit events, this process happens in the background and is not apparent.

For more information about the SMAPIs, see *z/VM: Systems Management Application Programming*, SC24-6324.

10.7 Preferred practices in z/VM security

The basics of z/VM security practices fall under the following categories:

- ▶ Define and deploy a security policy.
- ▶ Examine audit trails periodically.
- ▶ Apply recommended services.
- ▶ Data integrity
- ▶ Install and deploy an ESM, such as RACF/VM.
- ▶ User ID management.
- ▶ Passwords management.

10.7.1 Defining and deploying a security policy

Your z/VM system must be covered by a comprehensive security policy that includes the following items:

- ▶ Roles

Every VM has a role to play on the system. Defining these roles is integral to determining the scope of responsibility for your guest workload, or for the privileged users maintaining the hypervisor environment.

- ▶ Authentication

Access to system requires user ID and password or password phrase. There should be rules regarding the syntax of the password or the password phrase to avoid weak and easy to compromise choices. There also should be an expiration period to avoid the use of the same password indefinitely and a versioning criteria to avoid frequently repeated definitions.

- ▶ Authorization

Which CP commands or functions can a VM use? With which VM resources can a VM share and with whom?

- ▶ Accountability

Define the events that indicate that *something happened*.

- ▶ Audit

The audit trail is how the installation demonstrates its conformance to the corporate security policy. There must be defined the data access restrictions, the periodicity of data collection and data review and finally, where and for how long the audit data is archived.

10.7.2 Examining audit trails periodically

In z/VM, comprehensive auditing is available through RACF/VM. RACF/VM can audit every command and security-relevant event happening within the hypervisor, in accordance with a predefined security policy.

In the absence of RACF/VM, audit trails are generated by several CP command journaling options. LOGON, AUTOLOG, XAUTOLOG, and LINK journaling detects and records certain occurrences of the **LOGON**, **AUTOLOG**, **XAUTOLOG**, and **LINK** commands. Using the recorded information, you can identify attempts to log on to CP by users who enter invalid passwords. Also, you can identify any user who successfully enters the **LINK** command to a protected minidisk that user does not own.

Briefly, LOGON, AUTOLOG, XAUTOLOG, and LINK journaling works this way. Although journaling is turned on, CP monitors all occurrences of the **LOGON**, **AUTOLOG**, **XAUTOLOG**, and **LINK** commands. CP counts how many times a user enters one of these commands with an invalid password. CP can be set to take one or more of these actions when the count reaches a threshold value:

- ▶ Write a record to the accounting data set to record the incident.
- ▶ Reject subsequent **LOGON**, **AUTOLOG**, **XAUTOLOG**, and **LINK** commands that are entered by the user.
- ▶ Lock that terminal for a designated period.
- ▶ Send a message to a designated user ID to alert the installation about the incident.

While journaling is turned on, CP creates an accounting record each time it detects that a user successfully enters a **LINK** command to a protected minidisk that is not owned by that user. A protected minidisk is a minidisk whose password is anything but ALL for the type of LINK attempted, or a minidisk that is protected by an ESM.

To make LOGON, AUTOLOG, XAUTOLOG, and LINK journaling available and to specify options, you may use the **JOURNALING** statement in the system configuration file. To turn journaling on or off, use the class A **SET** command. To determine whether journaling is on or off, use the class A or C **QUERY** command. For more information about these commands, see the *z/VM: CP Commands and Utilities Reference*, found at:

http://www.ibm.com/support/knowledgecenter/#/SSB27U_6.3.0/com.ibm.zvm.v630.hcpb7/toc.htm

10.7.3 Applying recommended services

Periodically contact IBM Support Center and obtain a PSP for your z/VM. Carefully review the documentation and apply all the fixes that are provided.

Some services for z/VM might be particularly relevant to system security and integrity. These APARs are labeled SEC/INT and do not contain any additional information with regards to the problem included therein. Because service cycles vary from data center to data center, System z does not release data that is related to these vulnerabilities.

A subscription service is available for System z customers through the *System z Security Portal*.¹ When you are subscribed to this service, additional information that is related to the vulnerability is included in the form of a CVSS score and vector. This standardized form of annotating vulnerabilities allows you to determine whether a particular SEC/INT APAR might be relevant to your system.

10.7.4 Data integrity

Integrity statements that are made by IBM that are relevant to z/VM are associated with system integrity; they do not make statements about data integrity. Data integrity is related specifically to information on minidisks or administrator-controlled storage devices. Data integrity must be managed by the z/VM system administrator.

¹ You can register and find more information about the System z Security Portal at this website: http://www.ibm.com/systems/z/advantages/security/integrity_sub.html

An example of a data integrity concern is a multi-write (MW) link. If two VMs are granted simultaneous write access to the same minidisk, then changes made by one VM certainly will be overwritten by the other. This can lead to data problems. As a result, it should either be avoided (by not granting this access) or by mitigating it through the use of configuring reserve/release on the link access. This latter mechanism allows for locking of the minidisk against write access even when MW links are created.

10.7.5 Installing and deploying an ESM

ESMs such as the RACF/VM complements extend basic z/VM security features while providing more granularity and accountability of all accesses and operation events. An ESM's access control includes user verification, resource authorization, and logging capabilities.

User ID management

Here are some preferred practices for user ID management:

- ▶ Revoke IBMUSER and remove all its privileges.
- ▶ Convert any service VM not in use by your installation to NOLOG.
- ▶ Do not use the characters @, #, ¢, and “in a new user ID because the system, by default, assigns these characters as logical line editing symbols.
- ▶ Avoid using the characters colons (:), periods (.), semicolons (;), or slashes (/) in a new user ID. These characters are used, or can be used in the future, as delimiters in several CP commands that also take a user ID as a parameter. Results are unpredictable when these commands are entered with a user ID containing any of these delimiter characters.
- ▶ Do not create user IDs that are system keywords (such as command names, command operands, or 1- 4-digit user IDs that might be spool IDs). Assigning system keywords as user IDs can cause unpredictable results.
- ▶ Do not grant authorities or privileges beyond the scope of responsibility for any VM. This includes access to controlled CP commands, and advanced RACF privileges such as OPERATIONS, SPECIAL, or AUDITOR authority.

Password management

Here some preferred practices for password management:

- ▶ Employ passphrases. z/VM 5.3 with RACF 5.3 and higher allows for the use of a user ID and passphrase for user authentication in place of a user ID and password. You can select the method of user authentication to use (that is, a password, a passphrase, or a combination of both).
- ▶ Explore the use of mixed case passwords. z/VM 5.3 with RACF 5.3 and higher allows for the use of mixed case passwords. There is no requirement that mixed case passwords be used. A provider of service may use either uppercase passwords or mixed case passwords.
- ▶ Change the operator password for **RVARY** Commands. This must be changed from the default of YES. A guideline is to change it every 180 days.
- ▶ Expired new and reset passwords must be expired so whenever this user ID is logged on, its owner must provide a new password.
- ▶ Place special system DASD areas (allocation, warm-start, and checkpoint areas) on minidisks and then assign the minidisks to VMs with NOLOG passwords. This helps you identify areas on a DASD.

- ▶ Create minidisks for common user data and then assign the minidisks to VMs with NOLOG passwords. Users can then enter a CP **LINK** command to access those minidisks when they need the data.
- ▶ VM operator ID, logical operator IDs, and service machines passwords should be either logged by specifying the following items:
 - **AUTOONLY**: To avoid disclosing their passwords to other users and allowing these IDs to be activated only by **XAUTOLOG**.
 - **LOGONBY**: To provide individual accountability for shared use, as when each authorized user must provide its own user ID and password information for surrogate access to these machines.

10.8 Conclusion

Management of the hypervisor layer is a part of an end-to-end security solution. z/VM has been providing secure virtualization for over 40 years, and has been certified according to the Common Criteria regarding its capacity to provide both strong isolation of guests and multi-tenant environments. Knowing the features that are available, and the common practices that are associated with them, allows you to configure a secure environment in which to run guest workload, whether they are Linux, z/OS, VSE, or even CMS-based.



Linux on System z security

This chapter provides a collection of preferred practices for securing System z when you are running a Linux environment. This chapter highlights its unique features and also illustrates how taking advantage of such features can simplify system management.

The preferred practices to manage IT security are documented by several sources. These practices can be used to secure Linux on System z. However, several unique technologies that this platform uses should be considered when you define the security policies. These technologies can harden the overall security by providing centralized management capabilities and reduce the number of control checks compared to a server farm that is based on physically distributed servers.

This chapter describes some of the preferred practices that you can use to secure your data and your network, and also describes how to reinforce access control and authentication. This chapter can serve as a quick implementation guide that you can use when assessing the security requirements for your environment.

To acquire a complete and detailed look into security for Linux on System z, see *Security for Linux on System z*, SG24-7728.

This chapter covers the following topics:

- ▶ Security checklist
- ▶ Securing the logical access to z/VM
- ▶ Securing the data
- ▶ Securing the network
- ▶ Access control
- ▶ Authentication
- ▶ User management
- ▶ Audit
- ▶ Separation of duties

11.1 Security checklist

It is necessary to introduce security starting from the initial design of the environment. Every organization is likely to have specific requirements for information security that should be considered when building a checklist for security. Here is a high-level list that can be easily adapted and used on most environments:

- ▶ Protect your physical IT infrastructure.
- ▶ Secure the logical access to z/VM.
- ▶ Protect your data.
- ▶ Protect your virtual network.
- ▶ Secure the logical access to the Linux servers.
- ▶ Ensure continuous monitoring and auditing.

Physical security is common to any System z environment and is covered in 8.1, “Physical access controls” on page 188. All security principles and practices regarding physical security are applicable to all operating platforms on System z, including Linux.

The remainder of this chapter focuses on other practices to secure your Linux on System z infrastructure.

11.2 Securing the logical access to z/VM

When running Linux virtual guests with z/VM as the hypervisor, z/VM security is as important as what is implemented at the Linux level. When the hypervisor security is compromised, all the virtual components of your virtual IT infrastructure are potentially being compromised as well.

This section highlights how to secure several key z/VM functions and features that are important when supporting Linux servers as guests on this operating system.

11.2.1 z/VM user passwords

On a z/VM system, the passwords for users are not world-readable, and only z/VM administrators with specific privileges can access the user directory. The administrators can access to all users' passwords. For the sake of manageability and individual accountability, have an External Security Management (ESM), such as RACF, as the back-end database to hold the system users' passwords. An ESM can provide tools that make the privileges that are acquired through the authentication process expirable and subject to periodic evaluation and validation.

Password policies should be followed strictly, and using default or simple passwords is not recommended. It is always a preferred practice to control all accesses to users through the ESM.

11.2.2 Choosing the z/VM privilege class

The most basic level of security that z/VM provides is a user's privilege classes, which is a concept that sometimes is forgotten by Linux administrators. The CP commands that users can run on a z/VM system depends on the privilege class of which they are a part.

As a starting point, have your Linux virtual guests configured to have only the general user privilege class, which means they have access only to CP control functions over their own virtual machines and resources. A virtual Linux guest should not have additional privileges to define system-wide parameters of the z/VM system or other virtual guest settings, no matter how convenient it might be to perform such tasks from one of the Linux guests. The use of RACF can help reduce even more privileges that a user with privilege class 'G' can run.

11.2.3 z/VM network connection

There are various methods of connecting to a z/VM system for different purposes. This connection starts from the physical consoles and controllers that are attached to the System z and z/VM, and goes to the remote terminals that access z/VM through 3270 emulation.

Review network security at all levels to ensure that the data-in-transit is secured from threats. Use of encrypted communication is the key to ensure safe transfer of data bytes between hosts and clients.

11.3 Securing the data

This section describes the preferred practices for securing your information, and determining who can and who cannot access data from inside Linux and from z/VM. This section also describes how you can reduce vulnerability to harmful code and binary files by having fewer intrusion points, and how to encrypt your data to prevent information theft.

11.3.1 Securing your minidisks

The z/VM system is designed so that access to minidisks is denied for users who do not have the correct privileges, unless those privileges are otherwise granted. When an ESM is not in use, you must define passwords on the system's directory for the disks that you want to share. Although this sounds like a fairly secure approach, a preferred practice is to use an ESM to make your configuration more resilient and less error-prone. This section assumes that RACF is in use to secure your minidisks. However, this section provides examples about how to implement security when that is not the case.

You can use RACF to control who can link to z/VM minidisks by using profiles in the VMMDISK resource class. z/VM calls RACF for an authorization check in the VMMDISK class for two events:

- ▶ **LINK** command: One user's attempt to link to another user's minidisk
- ▶ **MDisk** event: A user linking to his or her own minidisk

MDisk events occur at logon time when a user's MDisk directory statements are being processed, or when a user runs a **LINK** command for a self-owned minidisk.

If the VMMDISK class was not previously enabled on your system, doing so is mandatory for the sake of security and integrity of your minidisks.

For performance or management reasons, certain situations might exist where having shared disks is preferable, but for the preferred practices that are covered in this chapter, assume that all the disks hold sensitive information worth protecting against unauthorized access. For each of those disks, a RACF profile should exist that determines who can access it and what type of privileges a user or group should have.

Anyone (even those with the most restrictive access possible, that is, READ) can copy the data files to another minidisk where they have control of the security characteristics, and then downgrade the security restrictions on those files. So, a preferred practice is to assign initially a universal access authority (UACC) of NONE, and then selectively, as required, grant access to the smallest number of users and groups possible.

With only a few precautions, you can have your minidisks secured against unauthorized access at the VM level. Using RACF, you can control who can or cannot access your disks, and you can also determine several of their access levels. In addition, RACF can also capture logs for audit purposes.

11.3.2 Reducing the intrusion points with shared disks

One golden rule of Information Management is that information should exist only in one location. This is easy to accomplish when you deal with applications that retrieve data from a database. But how can one handle a requirement to have files available across several servers and still maintain data integrity and keep data safe from unauthorized access?

On a distributed server farm, this task is accomplished by using a network service such as Network File System (NFS). However, because the data, confidential or not, flows through the network, such a solution requires additional security tools to eliminate or mitigate risks. Linux on System z can take advantage of z/VM features for this purpose. One such feature is the ability to connect virtually devices among guests within the same system. A possibility is to have a disk or a set of disks (an LVM group, for example) shared among multiple servers without the need for network services of any kind. Your data does not flow across the network and cannot be sniffed, reinforcing the overall security of your environment and reducing the number of intrusion points.

The ability to share disks among multiple servers is one of the values that is added by the platform, but you still must determine which file system goes on the top of those disks. z/VM can have the disks shared in read/write mode to all guests within the system, but you are not responsible for determining how the Linux systems control the I/O lock to avoid corruption of the file system, eventually leading to loss of data integrity.

If the ability to have multiple nodes accessing the same disks in read/write mode is a requirement, you might want to evaluate the use of a clustered file system. If you want to provide access to static content, such as the installation binary files for a specific software product, a static file system that is using z/VM minidisks that are linked in read-only mode is definitely worth implementing. It reduces the overall complexity and risk exposure of serving multiple copies of the same files across your servers.

If you do not have RACF enabled on your system, and you must share disks between virtual servers, you must determine which disk contains the data and then connect it to the target nodes.

z/VM can grant or deny access to the minidisks, depending on the password that is defined for that specific resource in the user directory. If you try to link a minidisk that is protected by a password without the appropriate password, an error message is issued. When ESM is not used, manageability is seriously compromised. Although the access to the user directory is protected and restricted to users with privileged access, it is still a clear text file, so the use of an ESM is preferred. The efforts to micro-manage many devices makes the whole process prone to human errors and more subject to security exposures.

The adoption of an ESM such as RACF, which was selected for this book, is considered a preferred practice for securing Linux servers on the System z platform.

11.3.3 Protecting the data with encrypted file systems

The dm-crypt subsystem in the Linux kernel is implemented as a device mapper that can be stacked on top of other devices that are managed through the device mapper framework. This means that it is possible to encrypt everything from entire disks to software RAID volumes and LVM logical volumes, adding a high level of flexibility to your encryption strategy, compared to cryptoloop, the predecessor subsystem of dm-crypt.

An important highlight is that the System z platform uses advantages through the use of cryptographic hardware and cryptographic functions that are built in to the central processor (CP). Through the use of Central Processor Assist for Cryptographic Function (CPACF), it is possible to offload cryptographic cipher calculations from the central processor, thus considerably reducing the processor cycles of such operations compared to the cost of having them done through software emulation. This must be considered when you size the hardware requirements for data encryption. The possibility to have this function enabled in all the servers on your virtual farm is a unique feature to help in evaluating which data to encrypt and how much of the resources are needed to do so.

For specific information about how to encrypt data on disks by using the Linux functions that are available on all the major distributions, see section 5.2, “File system encryption”, in *Security for Linux on System z*, SG24-7728.

Use the dm-crypt disk encrypt subsystem for flexibility and resilience. The dm-crypt subsystem is broadly supported on most Linux distributions, which makes it a reliable solution when the continuous development of the technology is considered. You also might consider having all your disks encrypted at a lower level in your storage array subsystem.

11.4 Securing the network

The System z platform provides a great security advantage when it comes to virtual networking capabilities. Using industry standards and providing various tools to control who can couple to the virtual devices, System z also provides auditing and troubleshooting functions. Using Linux on System z guests with correct connectivity can dramatically reduce physical intrusion points, making the physical infrastructure easier to maintain.

z/VM offers several possible network configurations, such as HiperSockets adapters, guest LANs, and virtual switches. Although guest LANs might be considered a viable alternative to provide connectivity among virtual servers, virtual switches are preferable compared with any other z/VM networking options.

11.5 Access control

In computer security, the ability of a certain subject or initiator to access a specific object or system resource is controlled by some sort of mechanism. Such mechanisms can either grant or deny access of a subject to an object, and also determine what privilege should be granted.

Based on that, two methods to control access are possible:

- ▶ The mandatory access control (MAC)
- ▶ The discretionary access control (DAC)

Although DAC enables owners of objects to grant access to other users, MAC has a *policy* as the center of all decisions. The security policies must be available and respected system-wide. The compliance of your organization's security policies should not depend on the discretion of each individual user. Because DAC cannot override what was determined by the MAC, only the function that is responsible for the overall security policy can change what is mandatory in your security policy.

DAC has been the usual method to control access to resources on most operating systems. It is built-in, and system administrators feel comfortable using it. An example of a DAC is the well-known file system object permission on UNIX like operating systems. Users can manipulate the bits that define what privilege the owner, the group, and others have on the system resources they own or have authority to control.

MAC, however, has been available for several years and for several platforms. Although various approaches can be used to implement this type of access control, it is closely related to multilevel security, which describes the process to grant access to an object, based on its classification, that is, based on its sensitivity and the security clearance that the initiator or the subject of the request has. That is exactly what we did with the implementation of the security labels while securing our network devices for the examples in this book.

With the advent of SELinux and AppArmor, MAC for Linux has become more mainstream. The increasing number of security threats that organizations face today has pushed more IT organizations into the implementation of system-wide security policies as an effective way to protect their systems. A well-implemented MAC can protect the resources from internal and from unknown external exposures.

Recommendation: To strengthen an organization's security policies, implement the MAC method.

11.6 Authentication

Although access control and authentication are closely connected, their preferred practices should be addressed separately. Access control defines who can access what and how this access can be made, but authentication involves determining whether someone really is who he or she claims to be. The users attempting to access a system or a resource must first give sufficient proof of their identity.

Linux offers a flexible interface to plug and unplug authentication mechanisms to meet the various security requirements that your organization might have. The Pluggable Authentication Modules (PAM) can be used as a powerful instrument to reinforce compliance with your security policies by only authenticating users who meet specific characteristics.

PAM separates the task of authentication into four independent management groups: account management, authentication management, password management, and session management. The groups can be stacked. Several PAM modules are available to meet a wide variety of security requirements.

Password challenge has been, for a long time, the most common method to verify somebody's identity during the authentication process. We recognize its use and understand that the authentication requirements vary from one installation to another. To avoid having passwords traveling over communication lines, the use of either RSA or DSA key pairs might be considered as an alternative. Although SSH provides an encrypted communication channel between the user and the server, the fact that the password does not go through the network is an additional factor that must be considered when strengthening overall security.

All the alternative methods that are listed here have advantages and disadvantages for the various scenarios and security requirements that the organizations have. We did not try to set a guideline or establish any type of comparison between the alternatives. Regarding authentication, there is no difference between a Linux server running on a System z server or another platform. The preferred practice is to take advantage of the PAMs and build the rules that suit your needs.

11.6.1 Improving security for SSH key pair

SSH is used as a secure alternative for Telnet and FTP services. Access to Linux guests can be facilitated with the use of SSH and the public key infrastructure (PKI). Each user has a public and private key pair that ensures the user ID.

Many users like to use this combination of SSH key pairs to access Linux guests. However, in complex environments where you must handle many servers, the management of SSH keys is a challenge for security administrators.

The process starts when an employee joins the organization. An SSH key pair is generated and the public key is added to the `authorized_keys` file in a user's home directory. Additionally, this file is synchronized to all Linux guests where the new employee requires access. Then, when an employee leaves the organization, you must remember to remove the public key from all `authorized_keys` files on all hosts. If you forget to perform this important task, you are putting your environment at risk for malicious access. You must take security seriously and accomplish it either by individually editing the `authorized_key` file to remove any no longer used keys at least once a year, or by creating an automatic process using shell scripts and so on to force key rollover.

11.7 User management

Depending on the number of servers that your organization must maintain, supporting distributed user IDs can be an effort. Even more difficult is providing security to all of them.

11.7.1 Centralized user repository

The adoption of a centralized repository to handle and maintain user information for multiple Linux servers is considered a preferred practice for the maintenance and consistency of the security policies that are applied to user management.

Being able to perform user administration tasks (such as adding, deleting, and changing account information), or resetting passwords from a single and centralized point, such as an LDAP server, can help keep security requirements and policies consistent throughout the infrastructure, and avoid having users' sensitive information, such as passwords, spread among hundreds of servers.

The traditional and most common method to authenticate users on Linux systems is through a password challenge. Unless a centralized repository is in use, the passwords must be maintained in the `/etc/shadow` file on each of the servers, multiplying the number of intrusion points to your systems. A single vulnerability on a single server can be exploited, and when the intruder is in, your entire infrastructure is at potential risk.

By using Linux native tools and functions such as PAM and the Name Service Switch (NSS), you can get the flexibility that you need to make easily your distributed Linux servers authenticate on a single, central LDAP server.

You can extend your LDAP repository with the scalability and availability you have for your Linux servers on the System z platform. If you have z/OS in your environment, another option is to use the z/OS LDAP server and RACF on z/OS.

Note: Although we suggest that your Linux user IDs be on a centralized repository, keeping the superuser (root) available locally is also wise. You can use this approach to log on to the servers during a planned or unplanned outage of the LDAP server, and to have each root user assigned a different password.

Implementing security is a matter of trade-offs. We explained how to turn on the virtual switch isolation, and make your virtual Linux servers fully compliant with your security policies by directing their traffic to the firewalls and routers. However, if your security policies allow you to establish a backchannel communication, such as guest LANs and HiperSockets, having a central user repository can help you reduce the intensive network traffic that using an LDAP server can cause on your external (outside the System z server) network segment.

11.7.2 Securing connections to the user information repository

As a default security preferred practice, all the connections to the LDAP server should be through an encrypted channel, either by using LDAPS on port 636, or TLS over port 389. Not all information about your central user repository is sensitive, and your security policies probably do not necessarily require this information to flow through an encrypted channel. However, as a preferred practice, use a secure, encrypted channel, which is much easier from an operational perspective than trying to classify all the information that is exchanged between your Linux clients and the LDAP server.

Some applications might not offer support to connections through SSL or StartTLS commands. However, from a security preferred-practice perspective, handle those applications as exceptions, assess their use risks as the business requires, and take the appropriate actions to restrict the scope of information they can get (as determined by your organization's security policies).

11.8 Audit

The first steps to make your entire organization less exposed to the threats and vulnerabilities of a business environment are to identify the most adequate security requirements for your business and, based on that discovery, create the policies that guide the overall configuration of your IT infrastructure. Information is considered one of the main assets of any company. A constant challenge is to make it always available and at the same time always secure.

The availability of strategic information can make the difference between a successful and a failed business. Today, security must deal with the information that the organization classifies as important, and also with the legal requirements within several lines of businesses around the world.

A well-defined security policy is worthless if you do not have a way to assess whether the policies are effective, that is, whether all the parties in the organization adhere to the policy and are playing the roles that they are expected to.

The ability to track changes in the profiles of resources that are handled by RACF extends the same ability to the virtual environment. In a virtual environment, you still must control the security of your physical resources, with the advantage of being able to control everything from a central location.

The resources and systems that are subject to auditing can vary from one organization to another, and the number of audit records that you must retain also depends on your security policies and legal obligations. Besides tracking all the changes that can potentially compromise your security, you must establish the appropriate actions to address them in time to avoid a major exposure.

11.9 Separation of duties

Separation of duties, as far as IT security is concerned, although laborious and in some cases expensive, is definitely a preferred practice. It avoids conflicts of interest and can better detect control failures that might lead to security breaches, information theft, and violations of the corporate security controls.

On z/VM, with RACF enabled, the security administrator, the system administrator, and the auditor are roles that can be easily deployed. Besides the auditor, which requires the user to be a member of a specific group, the MAINT and the SYSADMIN are standard users that are created to perform the system administrator and the security administrator roles, respectively.

In Linux, defining each job role scope is more complicated because everything converges on the root user ID. However, doing so is preferred by defining the policies, and having strong control of who in the organization has access to the superuser account. All other administrators, although able to run a privileged set of commands and with a high security clearance, should not be able to override the MACs and should not be able to manipulate the controls that are under the jurisdiction of another job role. As an example, the auditors should not be able to change the way that the resources are configured, but should be able to set up the audit logging requirements; such configurations should not be overrated by the owner of the resources or the system administrator. With the separation of duties, the functions of your systems and the integrity of your audit logs will not be compromised.

In numerous organizations, the same person might have various roles to play. Each role should have its own profile and user ID. With this approach, the individual with several roles is forced to change to the environment in which each role is performed. Such implementation can help protect the system against non-intentional changes to the security environment.

11.10 Conclusion

Linux on System z provides all the security controls, features, and applications that are available to Linux on other platform. In addition, IBM has enhanced Linux on System z to meet the needs of the most robust enterprise. Centralized authentication and audit features simplify management across the enterprise. Access to cryptographic functions can accelerate access for mobile applications because they work hand in hand with the application and data layer on System z. Hardware cryptography also provides the secure key and banking / financial functions that are necessary to secure the most critical applications and data. Cryptographic functions range from basic SSL acceleration to the latest Elliptic curve cryptography.

When deploying Linux on System z, organizations can take advantage of the Common Criteria certifications in hardware, virtualization, and Linux on System z. Hosting Linux on System z on z/VM as the virtualization platform can bring additional security through isolation and security controls for resources, such as the network and data.

Related publications

The publications that are listed in this section are considered suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks publications

For information about ordering these publications see “How to get Redbooks publications” on page 308. Some of the documents that are referenced here might be available in softcopy only.

- ▶ *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-8096
- ▶ *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-8097
- ▶ *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-8098
- ▶ *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-8099
- ▶ *IBM zEnterprise EC12 Technical Guide*, SG24-8049
- ▶ *IBM zEnterprise System Technical Introduction*, SG24-8050
- ▶ *Introducing the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security*, REDP-4528
- ▶ *Introduction to the New Mainframe: Security*, SG24-6776
- ▶ *Introduction to the System z Hardware Management Console*, SG24-7748
- ▶ *Security for Linux on System z*, SG24-7728
- ▶ *Security on the IBM Mainframe*, SG24-7803
- ▶ *Security on IBM z/VSE*, SG24-7691
- ▶ *System z Crypto and TKE Update*, SG24-7848
- ▶ *z/OS UNIX Security Fundamentals*, REDP-4193
- ▶ *z/OS V1R8.0 Network File System Guide and Reference*, SC26-7417
- ▶ *z/OS V1R8.0 UNIX System Services Planning*, GA22-7800

Other publications

These publications are also relevant as further information sources:

- ▶ “ESA/390 interpretive-execution architecture, foundation for VM/ESA” by D. L. Osisek, K. M. Jackson, and P. H. Gum in *IBM Systems Journal* Volume 30, Number 1, 1991
- ▶ “Securing a Virtual World” by Alan Altmark in *IBM Systems Magazine*, May 2009
- ▶ SHARE Session 12807 - IBM System z Hardware Management Console (HMC) Security, Brian Valentine, Kurt Schroeder, and Patrick Callaghan

- ▶ “Understanding z/VM Integrity and Security” by Alan Altmark in IBM Systems Magazine, November 2002
- ▶ *z/TPF - The evolution of transaction processing to open standards*, G299-0768
- ▶ *z/TPF Security*, GTPS-7MS5

How to get Redbooks publications

You can search for, view, or download Redbooks publications, Redpapers publications, Technotes, draft publications and additional materials, and order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Reduce Risk and Improve Security on IBM Mainframes: Volume 1 Architecture and Platform Security

SG24-7803-01

ISBN 0738440108



(0.5" spine)

0.475" x 0.873"

250 <-> 459 pages



SG24-7803-01

ISBN 0738440108

Printed in U.S.A.

Get connected

