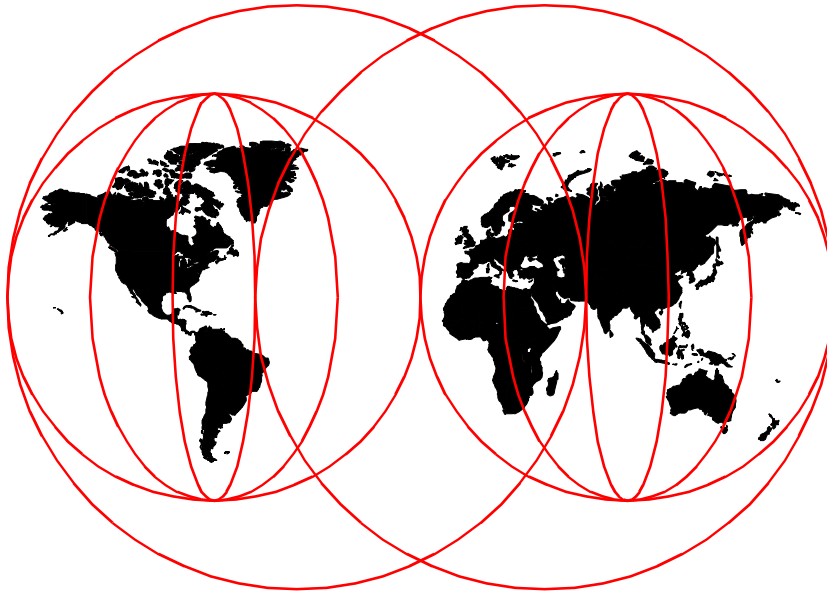


Understanding IBM SecureWay FirstSecure

*Heinz Johner, Valory Batchellor, Tiziana Dedato, Alvaro Franco, Alex Little,
Gerhard Rieger, Michael Roy*



International Technical Support Organization

www.redbooks.ibm.com

SG24-5498-00



International Technical Support Organization

Understanding IBM SecureWay FirstSecure

November 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, "Special notices" on page 349.

First Edition (November 1999)

This edition applies to Release 2 of the IBM SecureWay FirstSecure Program Offering.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B, Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1999. All rights reserved.
Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figuresxi
Tables	xv
Preface	xvii
The team that wrote this redbook	xvii
Comments welcome	xix
<hr/>	
Part 1. Introduction	1
Chapter 1. Security as a must for e-business	3
1.1 Meeting the demands of a changing world	3
1.2 Security policies	4
1.3 Preparing for threats	4
1.4 Risk assessment	7
1.5 The importance of standards	7
1.6 Related information	8
1.6.1 Web links	8
1.6.2 Discussion groups	9
1.7 Summary	9
Chapter 2. The systematic approach to managing security	11
2.1 Security with security policies	13
2.1.1 Assessing and defining an enterprise security policy	14
2.1.2 Centralized policy implementation	18
2.1.3 Policy-based authentication and authorization services	20
2.2 Secure boundary services	20
2.3 Immunity against intrusion	23
2.3.1 Traditional security and intrusion immunity	23
2.3.2 Protection against computer viruses	24
2.4 An infrastructure for public key technology	25
2.4.1 PKI description and capabilities	26
2.4.2 PKI components and functions	27
2.5 Security toolkit for custom-built applications	35
2.6 Conclusion	36
<hr/>	
Part 2. IBM SecureWay FirstSecure - Concepts and building blocks	39
Chapter 3. The IBM SecureWay FirstSecure building blocks	41
3.1 The Policy Director	41
3.1.1 Functions	42

3.1.2	Points of integration with other FirstSecure elements	43
3.2	The SecureWay Boundary Server	43
3.2.1	Points of integration with other FirstSecure elements	44
3.3	Intrusion Immunity	44
3.3.1	Tivoli Cross-Site for Security	45
3.3.2	Norton AntiVirus Suite	45
3.3.3	Points of integration with other FirstSecure elements	46
3.4	Public Key Infrastructure	46
3.4.1	Points of integration with other FirstSecure elements	47
3.5	The SecureWay Toolbox	48
3.5.1	Points of integration with other FirstSecure elements	48
3.6	The system environment	49
3.6.1	Hardening the system platform	49
3.6.2	Network name resolution	49
3.6.3	Physical design	50
3.7	Bringing it all together	50
Chapter 4.	The Policy Director (PD)	53
4.1	Policy Director architecture	55
4.1.1	User Registry	56
4.1.2	Management Server	56
4.1.3	Security Manager	56
4.1.4	Credentials Acquisition Server	57
4.1.5	Management Console	57
4.1.6	DCE Security Server	58
4.1.7	Distributed Computing Environment (DCE)	58
4.1.8	Bringing it all together	60
4.2	Authentication and authorization	61
4.2.1	Authentication	61
4.2.2	Authorization	66
4.3	Authorization: Objects, ACLs, and enforcers	67
4.3.1	WebSEAL	67
4.3.2	NetSEAL	71
4.3.3	NetSEAT	71
4.3.4	NetSEAL and NetSEAT working together	72
4.4	Users, groups, and security policies	74
4.5	Defining users and groups	79
4.6	Defining ACLs and attributes	82
4.6.1	Attributes	83
4.6.2	Permission attributes	86
4.6.3	Standard permissions summary	87
4.6.4	Utilizing ACLs as templates	88
4.6.5	Sparse ACL model: ACL inheritance	94

4.6.6	Evaluating an ACL	95
4.7	Policy Director namespaces	96
4.7.1	Standard namespace regions	97
4.7.2	Securing regions of the Policy Director namespace	100
4.7.3	Extending the Web space - Smart junctions	101
4.8	Managing administrators within the namespace	110
4.9	Accountability and monitoring	113
4.10	Scalability and availability	115
4.10.1	Replicated Policy Director	115
4.10.2	Utilizing the IBM SecureWay Network Dispatcher	117
4.11	Command line interface	118
4.12	Interoperability with other FirstSecure elements	118
Chapter 5. The SecureWay Boundary Server (SBS)		119
5.1	Introduction	119
5.2	Defining network boundaries	122
5.3	Secure boundary services	124
5.3.1	Traditional firewalls	125
5.3.2	Protocol content filters	129
5.4	Components of the IBM SecureWay Boundary Server	129
5.4.1	The IBM Firewall	130
5.4.2	MAILsweeper	135
5.4.3	WEBsweeper	137
5.4.4	SurfinGate	139
5.4.5	Security Dynamics SecurID authentication system	140
5.5	Building secure boundaries	141
5.5.1	Secured SMTP gateway	141
5.5.2	HTTP traffic	144
5.5.3	FTP traffic	147
5.5.4	HTTPS traffic	148
5.5.5	Domain Name Service	148
5.6	Scalability	149
5.7	Interoperability with other FirstSecure components	149
Chapter 6. Intrusion Immunity (II)		151
6.1	More about intrusion	151
6.2	Tivoli Cross-Site for Security	153
6.2.1	Security agent	154
6.2.2	Cross-Site server (Management Server)	155
6.2.3	Management console	156
6.2.4	Configuration considerations	156
6.2.5	Deploying Cross-Site for Security - Scenario 1	157
6.2.6	Deploying Cross-Site for Security - Scenario 2	158

6.2.7	Configuring Tivoli Cross-Site for Security	160
6.3	Norton AntiVirus Suite	162
6.3.1	Norton AntiVirus for Windows 95/98 and Windows NT	163
6.3.2	Norton AntiVirus for Windows NT Server	164
6.4	Interoperability with other FirstSecure components	164
Chapter 7. Public Key Infrastructure (PKI)		167
7.1	PKI for X.509 V3 architecture	169
7.2	Trust Authority 3.1	172
7.2.1	General capabilities	173
7.2.2	System configurations	177
7.2.3	Principles of operations	177
7.2.4	Default certificate types	179
7.2.5	Smart Card support	179
7.2.6	Directory support	180
7.2.7	Crypto hardware support	180
7.3	Supported standards	182
7.3.1	PKI basic standards	182
7.3.2	Standards that rely on a PKI	186
7.4	Scalability	187
7.4.1	Infrastructure	188
7.4.2	Systems	188
7.5	Interoperability with other FirstSecure components	188
Chapter 8. The SecureWay Toolbox (TB)		191
8.1	Toolbox APIs	192
8.1.1	LDAP	192
8.1.2	PKIX	192
8.1.3	Authorization API	193
8.2	Toolbox security services	193
8.2.1	IBM KeyWorks Toolkit	194
8.2.2	SSL APIs	197
8.2.3	LDAP API	198
8.3	Toolbox component APIs	203
8.3.1	PKIX API	203
8.3.2	Policy Director Authorization API	210
8.4	Documentation	214
<hr/>		
Part 3.	Deploying IBM SecureWay FirstSecure	215
Chapter 9. Installation planning and considerations		217
9.1	Pre-installation considerations	217
9.1.1	Hardening the operating systems	218

9.1.2	Security basics	219
9.1.3	Time synchronization	220
9.1.4	Low-level logging	221
9.2	Policy Director	221
9.2.1	First steps	222
9.2.2	Supporting software: DCE and LDAP	222
9.2.3	Policy Director prerequisites	225
9.2.4	Installing Policy Director	228
9.2.5	Compatibility, portability, and interoperability	236
9.2.6	Managing server configuration files	236
9.2.7	Testing the installation	237
9.3	SecureWay Boundary Server	238
9.3.1	Introduction	238
9.3.2	Basic firewall design	239
9.3.3	IBM Firewall	240
9.3.4	Additional filesets	242
9.3.5	Basic configuration of the IBM Firewall	243
9.3.6	Proxy user administration through Policy Director	244
9.3.7	SurfinGate	246
9.3.8	Security Dynamics SecurID authentication	249
9.3.9	MIMESweeper	249
9.3.10	Dependencies	251
9.3.11	Testing the installation	251
9.4	Intrusion Immunity	252
9.4.1	Prerequisites	253
9.4.2	Installation	254
9.4.3	Testing the installation	258
9.4.4	Operation	259
9.5	Trust Authority	259
9.5.1	System environment	260
9.5.2	Software requirements	261
9.5.3	Installation and configuration	262
9.5.4	Testing the basic installation	265
9.5.5	Running the RA desktop	267
9.5.6	Additional customization	269
9.5.7	The Trust Authority client	269
9.6	Toolbox	270
	Chapter 10. Deploying FirstSecure in practical scenarios	271
10.1	Policy Director	272
10.1.1	Locating Policy Director components	272
10.1.2	Technical implications	277
10.1.3	Scalability, availability, and load balancing	282

10.1.4	Managing IBM Firewall proxy users	283
10.2	SecureWay Boundary Server	287
10.2.1	A simple scenario	289
10.2.2	A more sophisticated scenario	290
10.2.3	Systems administration	291
10.2.4	Boundary services on the internal network	292
10.3	Intrusion Immunity	294
10.3.1	Locating intrusion detection components	295
10.3.2	Technical implications	296
10.4	Trust Authority	298
10.4.1	Locating Trust Authority components	299
10.4.2	Connectivity implications	300
10.4.3	More integration	302
<hr/>		
Part 4.	Appendices	305
	Appendix A. Introduction to cryptosystem	307
A.1	Basic terminology	307
A.2	The goal of a cryptosystem	308
A.3	Basic cryptographic algorithms	309
A.3.1	Secret-key encryption algorithm	310
A.3.2	Public-key encryption algorithm	311
A.3.3	Strength of cryptographic algorithms	312
A.4	The hashing function	313
A.5	Merging confidentiality and integrity functions	314
A.6	Digital signatures	316
A.7	Digital certificates	318
A.8	Cryptosystem as a base for a PKI	320
	Appendix B. Introduction to LDAP	321
B.1	The history	321
B.2	Protocol or directory?	322
B.3	What is a directory?	323
B.4	The LDAP information model	324
B.5	More LDAP features	325
B.6	Where to get more information	326
	Appendix C. Basic firewall and name service design	327
C.1	Introduction	327
C.2	Compartmentalized firewall environment design	330
C.2.1	Mail servers	332
C.2.2	Web proxies and servers	332
C.2.3	Mail/Web/FTP content filtering and anti-virus proxies	333

C.2.4 Encryption devices	333
C.2.5 Remote access servers	334
C.3 The need for highly available firewalls	334
C.4 Network address management	334
C.4.1 The Domain Name Service	335
C.4.2 DNS name structure	335
C.4.3 Static files	336
C.4.4 The Domain Name System (DNS)	336
C.4.5 The domain name space	337
C.4.6 DNS Security	345
C.4.7 Name server structures	345
C.4.8 Final word on DNS	347
C.5 Where to get more information	348
Appendix D. Special notices	349
Appendix E. Related publications	353
E.1 International Technical Support Organization publications	353
E.2 Other publications	353
E.3 Redbooks on CD-ROMs	354
E.4 Information on the Web	354
How to get ITSO Redbooks	357
IBM Redbook fax order form	358
List of abbreviations	359
Index	361
ITSO redbook evaluation	371

x Understanding IBM SecureWay FirstSecure

Figures

1. Corporate network layers	5
2. Functional elements of a security framework	12
3. Layers within a security architecture	13
4. Aggregation of security policies	19
5. PKI components	28
6. CA basic procedure	29
7. Example of a hierarchy of certificate authorities	31
8. Example of a certificate chain	32
9. FirstSecure overview architecture	37
10. Network environment with IBM SecureWay FirstSecure	51
11. Policy Director in the FirstSecure framework	53
12. Relationships between Policy Director elements and network services	55
13. Policy Director software services layers	59
14. DCE cell supporting Policy Director	60
15. Policy Director architecture	61
16. Logical smart junctions	69
17. Smart junctions as viewed through the Management Console	70
18. The combined NetSEAT and NetSEAL security model	72
19. NetSEAL server protecting back-end application servers	73
20. Using NetSEAL servers for secure communication across a WAN	74
21. Aggregation of security policies - Focusing on authorization	75
22. Relating users and resources via ACLs	78
23. User management interface	79
24. Defining a user	80
25. Group management	81
26. A group definition	82
27. Detail of ACLs view in the Management Console	83
28. ACLs mapping user/group permissions to a resource	84
29. Details of permission settings	87
30. The Object Space view on the Management Console	89
31. Detail of the objects displayed in the Object Space view	90
32. Detail of the ACLs displayed in the Object Space view	91
33. Push-down icon on Management Console	92
34. Result of pushing down the ACLs view	93
35. ACL inheritance	94
36. Policy Director namespace hierarchy	97
37. NetSEAL permissions	99
38. The Object Space view in the Management Console	103
39. The GSO Resource name view on the Management Console	105
40. Detail of GSO resource definition	105

41. Starting the junctioncp utility	106
42. The junction creation command	107
43. Detail of refreshed Object Space window	108
44. Assigning a resource credential	109
45. Separating Policy Director services across machines	116
46. Load balancing Policy Director using SecureWay Network Dispatcher	117
47. SecureWay Boundary Server	119
48. Differently rated networks	123
49. Required interconnections	124
50. Network boundaries	125
51. Logical mail flow	136
52. MAILsweeper with anti-virus (inbound mail)	143
53. Mail outbound chain	144
54. WEBSweeper with low-volume HTTP traffic	145
55. High-volume HTTP traffic configuration	146
56. Intrusion Immunity	151
57. Cross-Site for Security - Scenario 1	158
58. Cross-Site for Security - Scenario 2	159
59. Public Key Infrastructure	167
60. PKI policies	168
61. The overall Jonah architecture	170
62. The Trust Authority components	174
63. Sample RA Desktop view	176
64. The CDSA architecture	185
65. Toolbox	191
66. KeyWorks architecture	195
67. Authorization API layers	211
68. A secure firewall/DMZ solution	233
69. Sample firewall design	239
70. Chained proxy setup	247
71. Plugin setup	248
72. Management Console with new agent	258
73. Certificate enrollment dialog (partial page)	265
74. Check enrollment status form (partial page)	266
75. RA desktop (initial screen)	268
76. IBM SecureWay FirstSecure - The overall picture	271
77. Policy Director Components	273
78. Management Console after editing properties file	284
79. Detail of Directory Management Tool view	285
80. Initial user setup detail	286
81. Secureway Boundary Server components	287
82. Modern firewall environment	288
83. A simple firewall scenario	289

84. Sophisticated firewall environment	291
85. Required interconnections	292
86. Intranet with an internal firewall complex	293
87. Intrusion Immunity within FirstSecure	295
88. Cross-Site Agents and firewalls	297
89. Trust Authority within FirstSecure	299
90. Trust Authority for external users	302
91. Symmetric-key encryption	310
92. Asymmetric-key encryption	311
93. Hashing function - Integrity check	314
94. Combining confidentiality with integrity services	315
95. Digital signature mechanism	317
96. X.509v3 certificate format	318
97. X.509v3 certificate definition in ASN.1 notation	319
98. LDAP access to X.500	322
99. Stand-alone LDAP server	322
100. Simplest classic firewall	327
101. Classic DMZ and firewall design	328
102. Modern firewall environment	330
103. Example DNS domain	337
104. Example DNS subdomain	338
105. Domain names	339
106. Domains and Zones	341
107. Recursive mode example	343
108. Non-Recursive mode example	344
109. Internal Domain Server allocation	346
110. Implementing internal and external domain name spaces	347

Tables

1. Security requirements for business application life cycle	36
2. User definition worksheet	76
3. Group definition worksheet	76
4. Resource definition worksheet	76
5. ACL definition worksheet	76
6. Mapping ACLs to resources	76
7. User, ACL, and remote resource definitions	77
8. GSO user definitions	77
9. Some ACL examples	85
10. ACL types	85
11. NetSEAL ACL permissions	99
12. The traverse ACL permission	101
13. Access condition ACL permissions	101
14. The control ACL permission	101
15. Server management permissions	111
16. ACL management permissions	111
17. Action management permissions	112
18. Replica management permissions	113
19. Comparison of firewall mechanisms	127
20. Functions that initialize and terminate a connection	198
21. Session-handling functions	199
22. Functions that send or receive data	199
23. Functions for error handling	200
24. Parsing the results	200
25. Memory-freeing functions	201
26. LDAP URL functions	202
27. Character encoding functions	202
28. Other functions	202
29. Users registration and certificates creation	205
30. Revoking certificates and creating CRLs	205
31. Information about certificate requests	206
32. Certificate requests modification	206
33. Certificate requests modification	207
34. Retrieval of information about revocation requests	207
35. Revocation requests modification	208
36. Creation and management of browser-based certificate requests	208
37. Performance of various other tasks related to objects and certificates	209
38. Attribute list functions	213
39. Credentials functions	213
40. Functions for authorization decisions	213

41. Functions for initialization, shutdown, and error handling	214
42. API extensions	214
43. Additional AIX filesets	242
44. Cryptosystem matrix	308

Preface

The IBM SecureWay FirstSecure is the most comprehensive security solution framework in the IT market. It provides a security architecture and includes solutions for all disciplines of modern IT security that range from overall policy-based security management to desktop virus scanning. The IBM SecureWay FirstSecure offering includes five components: A Policy Director for policy-based authorization and authentication services, a boundary server that constitutes a firewall and content screening services for Web traffic (HTTP), mail (SMTP), and file transfer (FTP), an intrusion immunity component that scans network traffic for potential attacks and checks computers for viruses, a Public Key Infrastructure (PKI) that includes the IBM Trust Authority software for the management of digital certificates, and a security toolbox for application developers who need to integrate their applications in the security framework.

This redbook introduces and explains the architecture, the concepts, the building blocks, and the deployment of the IBM SecureWay FirstSecure framework. As a supplement to the rich set of documentation that comes with the product, this redbook will help you understand, plan, install, configure, and customize the new Release 2 of the IBM SecureWay FirstSecure. This redbook also discusses deployment scenarios, and appropriate solutions are developed and explained for typical environments.

The team that wrote this redbook

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

Heinz Johner is a professional technical author at the International Technical Support Organization, Austin Center. He writes extensively on the Distributed Computing Environment (DCE), security-related areas, and eNetwork deployment. Before joining the ITSO, he worked in the Professional Services organization of IBM Switzerland and was responsible for DCE and Systems Management in medium and large customer projects.

Valory Batchellor is a Senior I/T Specialist at the IBM Product Introduction Solutions Consultancy in Hursley, UK. For the last seven years, she has run Early Support Programs throughout EMEA for Tivoli and networking products. Prior to this, she worked in several areas including IBM Global Network, Systems Engineering, Product Management, and mainframe programming. She specializes in assisting people explain complex concepts to her in the belief that if she can understand them, so can others.

Tiziana Dedato is an Advisory I/T Specialist at the IBM Italy Global Services, Banking, Finance, and Securities Solutions Unit in Rome. She has experience in networking, software development, and system programming. She holds an Economics degree from Rome University in Italy. She has worked at IBM for nine years. Her areas of expertise include cryptography, and she has designed and developed a cryptographic system on an OS/390 platform for a customer in Italy.

Alvaro Franco is an RS/6000 SP specialist at IBM Uruguay. He has ten years of experience in the UNIX field. His areas of expertise include Open Systems, Networking, C programming, Internet issues, and AIX performance and administration. He has taught courses in these fields and has written about the RS/6000 SP. He worked for ORT University before joining IBM, where he assisted with Open Systems and Networking projects.

Alex Little is an Advisory Human Factors Engineer in the IBM Network Computing Software Division located in Research Triangle Park, NC and currently works in the RTP Human Interface Group. Alex has been with IBM for thirteen years.

Gerhard Rieger is a Software Service Engineer at IBM Austria in Vienna. He has three years of experience in the AIX field. He holds a doctoral degree in Electrical Engineering from Technical University of Vienna. He has worked at IBM for three years. His areas of expertise include software development, UNIX, and Internet security.

Michael Roy is the primary technical architect and IT management consultant for Blue World Information Technology Inc., an IBM Premiere Business Partner. He has 13 years of experience in distributed systems in IP network environments. He holds a degree in Computer Science and a variety of business management and technical certifications. His areas of expertise include Internet security and e-business solutions. He has designed and deployed wide-ranging, cross-industry customer solutions combining primarily utilizing IBM hardware and middleware.

Thanks to the following people for their invaluable contributions to this project or for their help in reviewing the contents of this redbook:

Nils Brubaker	IBM Raleigh
Rich Caponigro	IBM Austin
Frances Dodson	IBM Austin
Jeremy Finney	IBM Raleigh
Steve Fontes	IBM Raleigh
Shirley Fox	IBM Austin
Kevin O'Leary	IBM Austin

Karthik Ramamoorthy IBM Austin
Becky McKane IBM Raleigh
Mark Molnar IBM Austin
Ernst Plassman IBM Austin
Laura Rademacher IBM Raleigh
Ted Ralston IBM Austin
Dave Schneider IBM Austin
Roy Smith IBM Austin
Amy Wang IBM Gaithersburg
Gerald Weaver IBM Austin
Gerald Young IBM Raleigh

David Chung ITSO Austin
Martin Murhammer ITSO Raleigh
Jorge Ferrari ITSO Raleigh

Special thanks go to the editors for their help in finalizing the text and publishing the book:

Tara Campbell
John Owczarzak
International Technical Support Organization, Austin Center

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “ITSO redbook evaluation” on page 371 to the fax number shown on the form.
- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Send your comments in an Internet note to redbook@us.ibm.com

Part 1. Introduction

2 Understanding IBM SecureWay FirstSecure

Chapter 1. Security as a must for e-business

One of the fundamental components of e-business is the capability for employees, customers, and even competitors to work with a company's resources over the Internet. As these new business practices emerge, most enterprises are finding that their existing security infrastructure is not capable of meeting the rapidly changing and more rigorous demands of doing business over the Internet. The demands of network security have now gone far beyond simply managing user accounts and restricting access between internal and external networks to electronic mail and web traffic — these demands now require a sophisticated system that allows fine-grained access control to resources, yet is manageable enough to be tailored to protect systems from many types of security threats.

This chapter outlines some of the demands on network security systems and how a security solution can protect a business. Selected links to other security resources will be presented for readers who want to learn more about security concepts and products.

1.1 Meeting the demands of a changing world

The demands on Information Systems professionals are rapidly changing and expanding as the scope of business processing changes. Employees of an organization now need access to key business applications and data from virtually any location. Business data must now be accessed by trading partners at any time, while, at the same time, protecting access to sensitive resources. Customers want to be able to do business with a company via the Internet at any time and from any geographical location.

As these business demands evolve, so do the demands on security. While increasing numbers of people gain access to systems that can contain sensitive data, the potential for exposure also increases. It is no longer sufficient to just protect data from the "edge" of the Internet. Businesses must now make their public resources available to prospective customers while aggressively protecting their private resources with clear and reliable lines of demarcation.

It is also essential that system administrators be able to manage all these resources easily and without error. And, in case of any breach, the response must be swift and sure without disrupting other users.

Creating robust security systems is not the main business for most companies. Since it adds to the total cost of doing business, organizations

want it to be as simple and inexpensive as possible to implement and maintain.

1.2 Security policies

Policies are the means by which business practices are implemented within an organization's computer system. They are generally defined by a specific environment and reflect the specific needs of the organization. For example, an organization may have policies that determine who has access to particular assets or applications, what roles and responsibilities particular persons may have, and established procedures that dictate how some tasks are done. Or, in some organizations, confidentiality of resources may be required by government law, and legal recourse may even be possible against those who do not ensure that the policies are properly enforced.

Each of these, and more, may be reflected in an organization's *security policies*. Individual job needs may require access to information generally not available to other employees. A personnel manager may be able to access any employee's private information, while an accounting specialist may only have access to that person's financial data. And, an individual employee may only be able to access his or her own personal information.

One can easily see that in a large enterprise network, there may be a wide range of policies in effect, ranging from broad business policies to very specific rules addressing the access to particular fields within a data set. In general, these policies will be defined as specific business needs dictate.

Attempts to violate policies are commonly called *threats*. The following section will briefly discuss threats and how it is important that a carefully planned set of policies are defined to address them before the information is compromised.

1.3 Preparing for threats

Security may be defined as the freedom from worry about threats to safety. Within the context of an organization, this freedom of worry about threats may be achieved through a comprehensive set of security policies and tools to reduce the likelihood of specific types of threats from occurring.

Unfortunately, it is not usually practical to think in terms of totally eliminating threats to information — it is more appropriate to consider how disruptive or critical the loss of that information would be and how much security is commensurate with that resource. This is the *reality check* of any security system; it is unwise to spend great amounts of money establishing a system

to protect data that are relatively unimportant, nor is it wise to assume that critical data can never be compromised.

In order to do this, an organization must determine not only what resources need to be protected, but their value as well. Only then can a reasonable system be created, where the levels of security are commensurate with the value of the resources being protected. *Risk assessment* is an important topic, which will be discussed in greater detail throughout Chapter 2, “The systematic approach to managing security” on page 11.

The probability of a threat can be difficult to estimate since threats may come from a variety of sources with many different approaches and techniques as well (see Figure 1). Security threats may range from the innocent access of private information allowed by a lenient policy to a malicious attack by hackers with the intent to disrupt business or destroy critical data. Threats may also originate from within an organization as well — some estimates of annual damage from threats place the damage from internal sources at greater levels than from external sources. Therefore, it is critical that an organization be prepared to adequately address threats arising from both internal and external sources, with a level of protection that can resist attacks from even highly-skilled users with access to sophisticated tools. Even accidental damage could corrupt important data or disrupt operations at a critical time, both of which could be extremely costly to an e-business.

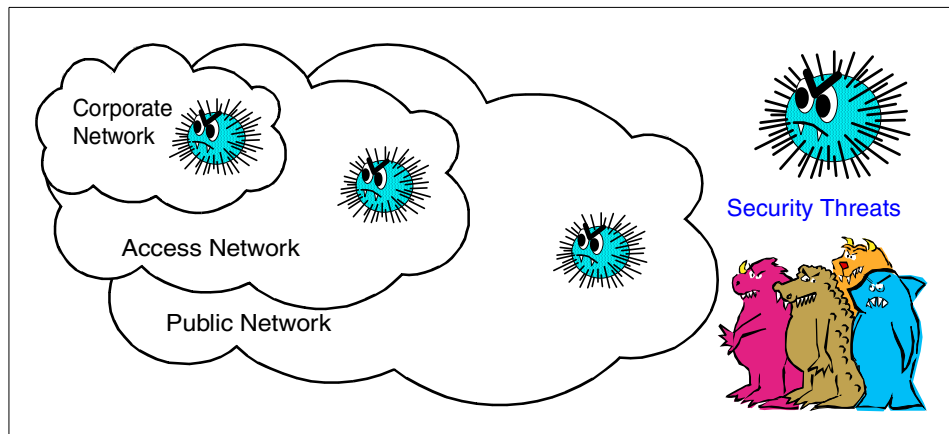


Figure 1. Corporate network layers

An organization may attempt to identify the resources that need to be protected and the policies that need to be put in effect to ensure their safety. Other policies may be needed to address possible threats. Some examples of policies created to thwart potential threats are: An asset classification policy,

a privilege policy, regulatory or compliance policy, an availability policy, and an access policy. And, there may be many more types. However, it is important to note that policy enforcement may require either technical or non-technical measures to best enforce it.

Some examples of non-technical measures are:

- Site security, where access is physically controlled.
- Physical security, locks required on containers or laptop computers.
- Personnel security, ensured via background checks and ID tags.
- Operational practices, dictating the control of keys, information control (phone books, organization charts, code names), or security awareness and training.
- Legal practices, where liability is limited via disclaimers, and violators are prosecuted.

Technical countermeasures are enforced using policy based security products shown in the categories below. They are frequently listed as the **5-A's**: **A**uthorization, **A**ccountability, **A**vailability, **A**dministration, and **A**ssurance.

Authorization	Authorization rules govern confidentiality, access control, and data integrity.
Accountability	Ensures that specific actions can be attributed to a certain individual.
Availability	The system is always available when promised.
Assurance	The user can be confident that the system security is operating as intended.
Administration	The system is manageable, yet can be changed as needs dictate.

The specific countermeasure taken depends on the source and type of threat likely for this component. These categories will be discussed in subsequent chapters.

Security threats can also originate from different physical locations, as depicted in Figure 1. Attacks may be detected or remain hidden. Successful attacks, whether detected or non-detected, can cause many problems for an organization, such as:

- Financial loss (theft, software licensing violations, and so on)
- Loss of public trust

- Corporate image (for instance, Web page alterations, major loss of capital)
- Loss of intellectual capital
- Human resource and/or customer privacy
- Litigation

Any of these problems can be very damaging to a business.

1.4 Risk assessment

Most threats can be prevented via appropriate security policies and their enforcement. However, some threats are naturally more difficult to prevent, which usually means that they are much more costly to guard against. Since most organizations are not in the business of implementing security — rather, it adds to the total cost of doing business — it is important to analyze the value of the resources to be protected against the amount and cost of security appropriate to attain that level. This information is needed to establish a good balance between the two.

The cost of security is a significant consideration; given unlimited financial commitment, nearly perfect policy enforcement can be attained. However, all resources really do not require a totally impenetrable system — in some cases, the total cost of replacing lost or damaged information may be significant, but substantially less than the cost of nearly perfect security. In this case, it is quite reasonable for an organization to decide that the most appropriate level of security be a prevention against the most likely threats, knowing that it is not cost-effective to attain more protection.

Sometimes, external requirements may dictate a prescribed set of policies for the protection of certain types of information. For example, the United States government frequently prescribes minimum levels of security for some types of information and requires any organization with access to that information to have specific policies in place. This generally requires the use of some set of standard enforcement methods to adequately protect these resources.

1.5 The importance of standards

When an organization defines policies and the actions required to adequately protect against threats, it is not always appropriate to design and develop custom-built procedures for this purpose. Besides the obvious cost of designing, testing, and implementing particular functions, it is virtually impossible to ensure that they provide an adequate level of protection. This is

why it is important that organizations use standard, system level enforcement methodologies.

As indicated earlier, threats can arise from a variety of sources and motivations. This makes it essential that policy enforcement systematically cover all the possible access points in order to ensure that the policy is effectively being implemented.

Another advantage of standards is that they are generally more predictable, therefore, making it easier to gauge their strength. Having a known level of security is critical in performing accurate risk assessments. In addition, they can help protect against potential litigation in cases where specified levels of protection are required, such as with government assets. Types of security assessments will be covered in Chapter 2, “The systematic approach to managing security” on page 11.

Finally, interoperability is a requirement for most enterprise environments. It is critical that a security solution protect resources on all platforms within an organization. A solution that does not interoperate with tools and products on other platforms can very easily leave some policies unenforced. Therefore, it is important to consider the capability to operate with other industry products as a critical requirement for most heterogeneous environments.

1.6 Related information

The following links can provide more information about various security concepts. Please also refer to Appendix E, “Related publications” on page 353 for further references.

1.6.1 Web links

IBM SecureWay FirstSecure – This is the home page for IBM FirstSecure products. It also contains a library section with links to various white papers and other documentation:

<http://www.software.ibm.com/security/firstsecure>

Open Group Portal for Security – Is a listing of links to various areas. Open group membership is required for access to some areas and documents:

<http://www.opengroup.org/security>

CERT Coordination center – Is an organization that works with the Internet community to raise awareness of computer security issues. There are some

good references for those who want to learn about various security issues. Some tools and case studies are also available:

<http://www.cert.org>

W3C Security Resources – This is the W3C page with security links and information:

<http://www.w3.org/Security>

The Internet Security Conference (TISC) – Is a useful page of links covering many security-related issues:

http://www.tisc.corecom.com/security_links.html

Security Search – Is a Web-based security search engine:

<http://www.securitysearch.net>

1.6.2 Discussion groups

The following Usenet discussion groups may provide a forum for discussion of security-related issues as well as links to other related discussion groups:

`comp.security.misc`

`comp.security.unix`

1.7 Summary

This chapter shows that it is important to establish a set of comprehensive security policies that can be enforced within the scope of an organization's existing network. Security is a requirement for e-business solutions; security solutions should allow for a maximum of freedom, while at the same time provide an adequate level of safety. Security should not be considered as or implemented in such a way that it limits the purpose of an organization's primary business.

It is important that organizations can rely on comprehensive and integrated security solutions that cover the broad range of requirements and yet adhere to existing and emerging standards. IBM SecureWay FirstSecure provides such a solution that includes components for all areas of architected network and computer security.

Chapter 2. The systematic approach to managing security

The previous chapter described the importance of setting security policies that adequately balance the threats and risks that exist in modern distributed computing environments. Various methods have been invented, and products have been developed throughout the industry to fight such potential attacks and preserve the integrity of a computing system. Some popular examples are anti-virus software, firewalls, and various authentication methods.

Anti-virus software runs on a computer, and it scans memory and executable files for known viruses. Firewalls connect two networks, usually a corporate's internal (secure) network and the external (non-secure) Internet, in such a way that potential intruders from the non-secure network cannot gain uncontrolled access to resources on the secure network. Authentication methods are necessary to ensure that people (or services running on computers) are who they claim to be.

It is important to adopt a systematic approach to security. One of the major concerns that emerges with security solutions is the overall complexity and cost. In fact, the following inhibitors to deploying security solutions are frequently mentioned:

- Security is too complex.
- Security policies are becoming impossible to implement.
- The total cost of security is escalating.
- Security topics are stopping e-business initiatives.

These statements are evidence of an inadequate approach to solving security problems. What is needed is a solutions framework that includes the major disciplines of IT security as part of an overall architecture rather than a set of unrelated products solving point problems. These areas include:

- Comprehensive security policies, their proper definition, handling, and implementation.
- Secure boundary services to securely connect computer networks.
- Intrusion detection and immunity to detect potential threats.
- A public key infrastructure to securely identify and handle identities.
- Toolkits that provide hooks to the security infrastructure for custom-built applications.

An information system framework or architecture is defined by the following:

- A set of components

- The interfaces of the components
- The protocols used to communicate with the components
- The structure formed by the interrelationships among the components

A framework helps organize the typical *point solutions* approach by:

- Categorizing security functionality
- Placing security component technologies into appropriate categories showing which categories are required in which environments
- Showing how the individual components are used to solve specific protection problems in each environment

In short, a framework defines a standard set of security components and a structure describing the relationships among them (Figure 2). In addition, the framework integrates countermeasures to provide the protection needed to support the security policy.

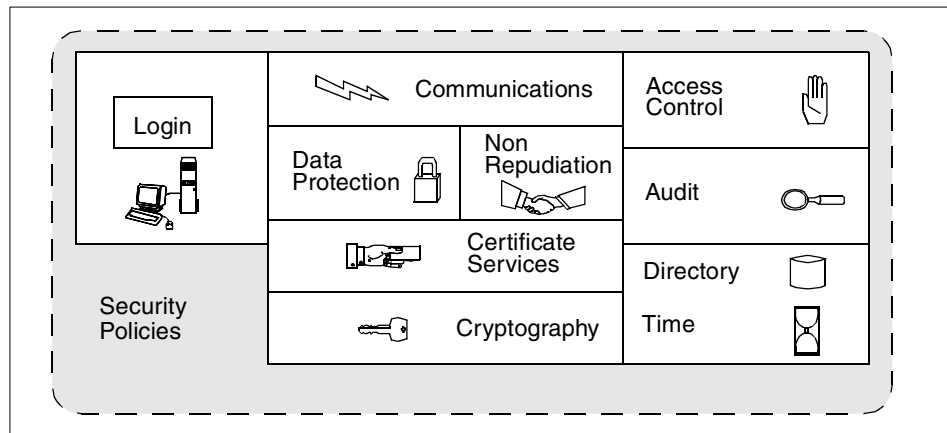


Figure 2. Functional elements of a security framework

This chapter discusses the disciplines of such an architectural approach to enterprise security and introduces the IBM SecureWay FirstSecure framework approach to implementing such an end-to-end architecture.

If an organization's security is based on a well-planned overall architecture, it is possible to provide layers of fine grained control so that the goals of security management can be applied and easily managed. One way of depicting this concept is through the idea of a layered approach, which is represented in Figure 3 on page 13; however, this illustration represents only part of the complete security mechanisms.

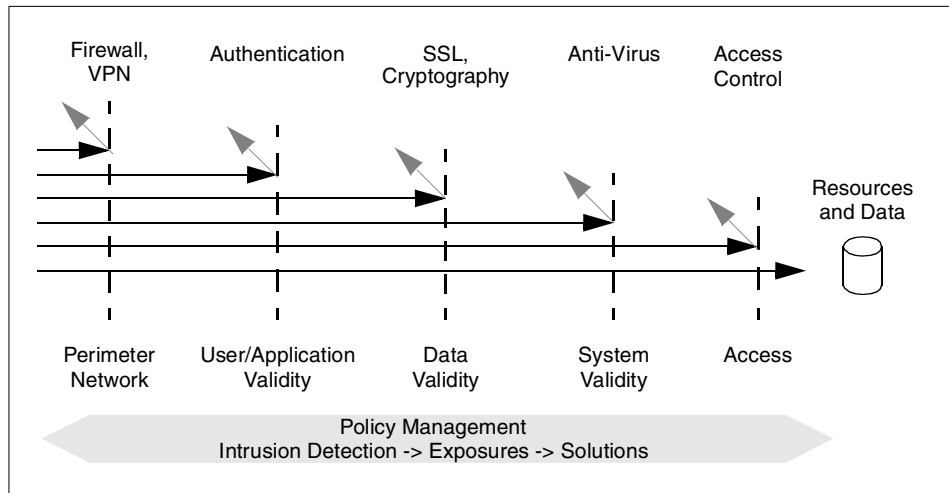


Figure 3. Layers within a security architecture

2.1 Security with security policies

An organization may employ a number of different network security products, such as firewalls, intrusion detection software, anti-virus solutions, and public key systems to protect the resources upon which it relies on to achieve its business goals and functions.

In most instances, separate network security products require a set of product-specific security policies that regulate how various security requirements (authentication, data protection, authorization, and so on) can be satisfied by that product. However, implementing and supporting many such individual products, which are not designed to fit into an overall framework, may lead to:

- Increased cost of specialized skills required.
- Increased potential of human error when translating product-specific security policies into overall enterprise security policies.
- Creation of a “weak link” in the overall security when the combined set of specific products does not completely cover the spectrum of overall security requirements as defined in an enterprise policy definition.

For these reasons, corporate security policies, supported by comprehensive tools, are key for an enterprise-wide security solution.

2.1.1 Assessing and defining an enterprise security policy

Before considering the details of a solution, this section briefly introduces the development of enterprise policy definitions. These policy definitions are the result of comprehensive analysis and risk assessment in the context of the organization's business function, where many technical, business, and legal issues must be considered during this process.

Typically, the process of security policy definition follows these stages: Assessment and planning, architecture and design, and, finally, implementation. The following discussion concentrates on generic, Internet-related assessments and planning tasks. All the techniques listed would not necessarily apply in any specific situation, although there should be at least one technique that is useful for any given situation. It is not the intent of this chapter to serve as a guide for performing these activities. If the expertise is not readily available within an organization, the assessment and planning activities should be performed by a security expert. In many cases, this may be a consultant hired to oversee the design and implementation of a comprehensive enterprise security system.

2.1.1.1 Security assessments and planning

For a thorough solution, utilization of best practises and proved procedures for defining policy should be discussed. Some of the key industry standards for such a collection of documented methodologies includes the British Standard Code of Practise, BS7799 and ISO 7498-2, that can be accessed, for example, at the following Web sites:

<http://www.bsi.org.uk>

<http://www.iso.ch>

Assessment and planning may include some of the following activities to define policy and procedures for an enterprise:

- *Security workshops* – The activities of a workshop include understanding the business activities that are using critical internal and external connections, understanding the key threats/perils related to these activities, understanding the nature and priority of the organization's specific security requirements, understanding and analyzing the security implications of the network topology, identifying and analyzing the key security components in the network design, identifying and analyzing the security characteristics of key applications related to the external connections and business activities, and, finally, discussing the security implications of future business plans and any impacts they might have on the current network topology and components.

- *Information asset profile* – The first stage is to identify the organization's critical assets, identify their owners and custodians, determine who depends on these assets and how they are used across the organization, classify the security requirements for protecting these assets consistent with the business needs, and relate these security requirements to the organization's key business issues. Interviews with key business and IT managers in the organization are conducted to understand what the critical business assets are and the nature and severity of any security risks and exposures to those assets. The risks examined are:
 - Impact to the organization if critical information gets into the wrong hands (*confidentiality*).
 - Impact to the organization if the wrong information is used (*integrity*).
 - Impact to the organization if critical information is not available for use when needed (*availability*).
- *Security health checks* – This method identifies both the strengths and weaknesses in the organization's IT security controls. Typically, consultants conduct interviews with key managers and staff members in the organization to understand what security controls are in place in the following ten management areas: Policy, organization, personnel, physical controls, asset classification and control, system access control, network and computer management, business continuity, application development and maintenance, and compliance.
- *Security process assessment* – Compared to a security health check, this activity goes into sufficient depth to verify that each control selected has the right processes in place to implement the control. The review will be conducted against an agreed, predefined information security standard or code of practice using interviews with appropriate staff within the organization. Security consultants will also verify the accuracy of the answers given by reviewing the actual processes and related documentation. This is done by inspecting examples of the process deliverable or outputs provided by the individuals who are responsible for executing the processes being reviewed.
- *Application security assessment* – During this process, an in-depth, end-to-end review of the business application will be conducted looking at the application's architecture, design and function, its development and maintenance processes, its operational processes and technology components including the platform it runs on, the networking services used, and any data base or operating platforms services used.
- *Network security assessment* – This typically includes a technology and management analysis. The technology review component consists of

intrusion tests and configuration analysis that gives the involved personnel a thorough understanding of the strengths and weaknesses of the internal network components. The management review component consists of interviews with administrators and management and reviews of documented security policies, standards, and processes.

- *System security assessment* – This process typically assesses each component's (hardware and middleware) mechanisms for identification and authentication, access control, confidentiality, integrity, non-repudiation, audit, and alert in the context of the organization's documented policies, standards, and processes.
- *Site security assessment* – Site assessments will determine if procedures have been implemented to ensure a secure business environment for all employees and other persons working in the facilities. In addition, emergency plans covering anticipated emergencies and catastrophes have been established, and plans that adequately address the protection of people and assets should be available. Analysis of procedures that have been implemented to report and analyze security incidents, bring them to closure, and prevent reoccurrence will be considered. There should be effective management processes to protect proprietary information and assets from unauthorized disclosure, modification or misappropriation, and, finally; there is a process in place to provide management with a validation that the security controls within the scope of this engagement are operating effectively.
- *Internet security assessment* – This is designed to help an organization to minimize the risk of a hacker causing damage to the network. This assessment provides a comprehensive review of the Internet solution on both a technical level and a management level. The technology review, consisting of multiple intrusion tests and configuration analysis, gives a thorough understanding of the strengths and weaknesses of an Internet solution. The management review consists of interviews with administrators and management and of reviews of security documentation. This provides an organization with insight into how the organization is prepared to handle the security of the solution over time.
- *Ethical hacking* – Ethical hackers can simulate a real intruder's attacks but in a controlled, safe way. Consultants can tell what they find and what can be done to fix any problems and issues.

At the conclusion of each of these processes, reports are prepared and delivered to the organization. This information forms the basis for further architecture design.

2.1.1.2 Architecture and design

Once the security business and technical goals are gathered and assessed in terms of security capability, an overall security architecture can be designed. Some of the activities involved in this process include the following elements:

1. *Security policy definition* – First, the security policy should be developed. This definition document should contain at least the following items:
 - A definition of information security with a clear statement of the organization's goals including a list of resources (subjects and objects) that are subject to security policies.
 - An explanation of specific security requirements including:
 - Any relevant legislative or contractual requirements relevant to the organization
 - A plan for security education, virus prevention and detection, and business continuity
 - A definition of general and specific roles and responsibilities for the information security program
 - The process for reporting suspected security incidents
 - The process, including roles and responsibilities, for maintaining the policy document
 - A set of rules that govern rights and permissions of subjects and objects
2. *Security standards definition* – The intent of this process is to develop a detailed work plan and, on a continuing basis, to ensure that all work performed is designed to satisfy the organization's needs. To ensure the standards developed during this activity meet the business needs and can be realistically implemented, security standards should be applied together with "best practices" selected from industry standards for commercial environments in areas, such as organization, personnel, physical controls, asset classification and control, network and computer management, business continuity, and compliance.
3. *Security process development* – The security experts might first review and assess the security processes that have been selected at one or more location. The assessment will identify both the strengths and weaknesses in the processes as they are currently implemented. Using this information, the responsible experts define and formally document new processes that will meet business needs in terms of efficiency, effectiveness, and adaptability by using a formal process management methodology.

4. *Enterprise security architecture* – This assessment provides a comprehensive, standards-based framework for managing an organization's security program consistent with its business objectives.
5. *Internet security architecture* – This process provides a comprehensive, standards-based framework for managing an organization's Internet security controls consistent with business objectives.
6. *Secure solution design* – The assessment team should now have the necessary information required to design a secure e-business solution that addresses the organization's needs in one or more of the following areas: Simple access, external publishing, private publishing, production access, e-commerce, collaboration, and hosting.
7. *Detail design* – As in other I/T projects, the final step includes product evaluation and selection and implementation. The individual products are implemented to form a comprehensive solution.

At the conclusion of each of these processes, appropriate reports would be prepared that then form the basis for further security implementation. After the above steps are complete, the detailed plans should provide a clear picture of the organization's security goals, procedures for implementing them, and the products needed to attain these goals.

2.1.1.3 Security architecture implementation

After the overall security architecture is formulated, the next step is to implement the appropriate policies, configurations, and procedures associated with the individual products identified in the Secure solution design. This is discussed in greater detail in subsequent chapters.

2.1.2 Centralized policy implementation

This section discusses the implementation of security policies within an organization. There are numerous advantages to utilizing a standard policy service to manage authorizations. Some potential advantages may include:

- Reduced cost of application development
- Reduced total cost of ownership and management of separate authorization systems
- Ability to leverage existing security infrastructures
- Enable newer and different kinds of applications
- Enable a higher degree of consistency
- Enable a quicker time-to-market
- Capability to share information securely

- Concentrated design and management skills

With proper planning, this should allow the development of a comprehensive set of policy definitions that can be applied across the network (Figure 4).

Policies are the rules that define how subjects (for example, users) can access objects (such as applications). Subjects can be defined and managed individually or, for easier management and improved scalability, organized in groups of subjects for which the same set of rules apply.

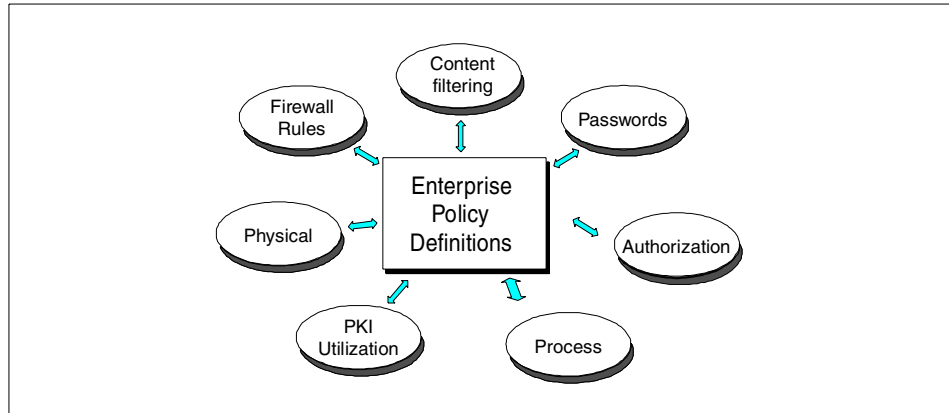


Figure 4. Aggregation of security policies

For example, some possible cases where access policies would be applicable are:

- An employee accessing an internal Web server.
- A customer accessing a corporate's external Web server.
- A user accessing a program that can be launched by a specific Web server.
- Employees initiating Telnet or FTP sessions to other computers outside the company.
- Software programs attempting to access other servers/programs in the enterprise.

In order to simplify the definition, management, and enforcement of security policies, an easy-to-use tool should be readily available. Such a tool ideally provides the following high-level functions:

- A single management console for enterprise-wide policy management and a single point of management for security policies

- Policy-based authentication and authorization to Web-based (HTTP) and non-Web-based (Telnet, SMTP) application servers
- APIs for custom-written applications to make use of the functions
- Security provisions to protect the policy manager itself

The following section considers some other elemental security concerns and abstract methods of solving security issues.

2.1.3 Policy-based authentication and authorization services

When an enterprise policy has been defined and documented, the authorization elements can be applied to a server that performs both course and fine grained authorization (or access) control.

A tool that incorporates policy-based security services is the key element in a security architecture and framework. The IBM SecureWay FirstSecure framework provides such a *Policy Director* that is the central component of the framework.

Authentication and authorization in most systems, both legacy and new, is tightly coupled to individual applications. Developers typically build applications over time to serve their business needs. Many of these applications require some specific form of authentication and authorization. The result is often a wide variety of applications with different implementations. These proprietary authorization implementations require separate administration, are difficult to integrate, and result in higher costs of ownership.

Chapter 4, “The Policy Director (PD)” on page 53, discusses the architectural elements of the Policy Director and integration strategies into other FirstSecure elements. Chapter 10, “Deploying FirstSecure in practical scenarios” on page 271, considers the practical view, which includes other FirstSecure elements being used in conjunction with the Policy Director. The main thrust of these scenarios is to provide guidance that spans many FirstSecure elements, not simply installation guides for individual point products.

2.2 Secure boundary services

The computing environment of an organization usually can be divided into different parts based on security needs. One part may contain employees with personal computers who need to access various resources to do their respective jobs. Another part may contain the Internet Web servers, and a

third section could contain the vital business servers. A functional view may depict all these computers as connected together, but from a security perspective, it is necessary to restrict the possible interactions between machines within the internal network.

A common approach for meeting this type of security requirement is to network computers with similar security requirements together and to establish security enforcement points between these sub-networks. These security enforcement points, called *firewalls*, provide controlled channels through which data traffic can flow. This ability to specify where the traffic flows allows an organization to apply policies at that point of control.

One might say that the function of a firewall is to connect secure and non-secure networks. However, use of a firewall can assist in security administration as well. For example:

- A firewall can allow outbound connections from the secure network to the non-secure network. Since users on the secure side are generally trusted, it may be adequate to use an easier to administer weak authentication technique, such as IP address checking or unencrypted passwords, for these users.
- However, in cases where users on the non-secure network are allowed to access certain services within the secure area, it is critical that only strong authentication is used to prevent threats, such as IP address spoofing or password eavesdropping. Here, the firewall allows services to be made publicly available (for example, mail delivery) but still remain secure and protected.
- Data connections over the non-secure network can be protected from eavesdropping through the installation of appropriate encryption.

A secure firewall can also examine the content of the traffic passing through the controlled ports. Potentially damaging applications and data, such as certain ActiveX controls, Java classes, or viruses, can all be prevented from passing through the boundary server into the secure network by use of a content filtering program.

In summary, a secure boundary services system should include the following functions:

- Deny or allow traffic depending on its direction and some weak authentication
- Deny or allow traffic depending on its direction and some strong authentication

- Permit access to selected services depending on its direction
- Detect and block transfer of malicious code
- Detect and block electronic mail (including attachments) that pose a threat of any kind
- Maintain application independent secure data transfer channels over the non-secure network (VPN, virtual private networks)
- Means to totally block access to services depending on the direction
- Deny everything that is not explicitly permitted

Note

Firewall outbound connections consist of traffic that is *initiated* on the secure (protected) side and whose peer is on the non-secure side. It does *not* specify the direction of the data. For example, downloading of a Web page from the Internet through the firewall uses an outbound connection, but data flow both ways.

User rights and their authentication should be managed from the same place, as should other security related tasks, where the boundary servers would also have access to the central policy data.

In practice, a firewall might connect one or more secure networks, one or more non-secure networks, and one or more demilitarized zones (DMZs). A DMZ is a special isolated network that is neither part of the secure or non-secure network. The resources within the DMZ provide services to users on the non-secure net but need to communicate with the secure internal network too. Since accesses from the non-secure network means that systems in a DMZ are at a higher risk of getting compromised, traffic from the non-secure network to these machines must pass through a firewall. Then, all traffic from the systems within the DMZ to the secure network also must pass through a firewall.

Because several different encryption, authentication, and filtering tasks may be deployed at a single network boundary, it might not suffice to implement all these tasks on only one machine. Cases where it may be advisable to spread the load over multiple hosts include performance requirements, platform differences, or the intent to keep the firewall itself as simple as possible. Therefore, as more hosts are added to improve functionality, it is essential that the new traffic corridors equally enforce the organization's security policies.

2.3 Immunity against intrusion

Intrusion immunity encompasses the technology that either detects attacks on a network (intrusion detection) or potential threats, such as viruses or trojan horses, that have already entered a system. The latter is performed by anti-virus programs.

An *intrusion detection* program is similar to a burglar alarm for computer systems. Running in the background, an intrusion detection program can monitor system activity for evidence of an attacker either trying to break into a computer or misuse the resources. If employed to protect a single host machine, it is called *host-based intrusion detection*. When monitoring network traffic, this type of intrusion detection system is referred to as *network intrusion detection*.

Network intrusion detection systems are usually capable of understanding traffic from the lowest IP level up through application protocols. They can interpret traffic data, such as source and destination addresses, protocols and services, time of day, interval, weak authentication, and packet integrity. By analyzing this information, they can detect a wide range of attack forms from low level denial of service or rerouting attacks up to brute force password guessing or unsolicited domain name server (DNS) zone transfers. In reaction to suspicious activity, they can generate alerts, reports, and when appropriate, application interfaces to the firewall's programs are set up. They can even arrange for cutting the suspicious connection via the firewall program or other means.

2.3.1 Traditional security and intrusion immunity

When a correctly configured security environment is already deployed, it may seem unnecessary to apply an intrusion immunity system. While there is no simple answer, the following thoughts should help to understand its advantages.

- The main concern of application server platforms and applications usually is function, not security. While some basic level of security can be attained with proper measures during installation (see 9.1.1, "Hardening the operating systems" on page 218), it is often not possible to improve security of services, such as NFS (Network File System) or POP3 (post office protocol), when they cannot be disabled.
- Security enforcement points, such as firewalls, have good, but rarely comprehensive, security features. If they only implement a subset of an ideal security policy, there is a good chance that an intrusion detection

system – as long as it does not use the same algorithms – may cover another subset, thus, reducing the remaining gap.

- Filters, authentication software, and other security tools usually provide comprehensive logging of many events but do not interpret that data at a higher level or watch for repeated failures and other attack fingerprints. For example, a firewall filter might log an incoming ping request but would not recognize that it is an oversized packet (ping of death), which may not cause any harm, but is often the first of a series of different attack attempts. A network intrusion detection system might trigger an appropriate event because it is attempting to analyze the data rather than just logging it.
- In a heterogeneous enterprise security environment, there probably are no enterprise wide log and event management centers where data are combined data from different sources, thus, allowing the recognition of “distributed” abnormalities. However, a distributed network intrusion detection systems may circumvent this shortcoming by watching several parts of the I/T infrastructure (while, of course, adding another source of log data and alerts).
- A carefully implemented intrusion detection system may not be visible to a hacker (as opposed to a firewall acting as mail gateway). This allows it to be free of any server functionality that might act as a starting point for attacks, thus, making the intrusion detecting system itself very secure.
- Last, but not least, if an intruder succeeds in getting control over a server or firewall machine and disables all components that are installed to uncover his or her activities, a network intrusion detection system may still monitor the traffic and warn of the abnormalities.

For these reasons, intrusion immunity must be seen as a valuable complement to a traditional security infrastructure.

2.3.2 Protection against computer viruses

While network intrusion detection is useful for controlling the dangers of direct attacks via the network, *computer viruses* are a threat to computer security that are potentially much more damaging.

A computer virus is a piece of code that can nest within a program without affecting its primary function or destroying it. It becomes active when its hosting program is executed. Viruses can enter a network via infected program code on diskettes, CDs, tapes, or network data transfer, such as E-mail. Once within the network or computer, they can further intrude on other systems via downloads or e-mail attachments.

Once a virus is on a host in the network, it can further spread using the same mechanisms as for intruding, or via internal communication ways, such as disk shares. It infects other programs it gets access to and, at some time, starts its specific malicious action, such as destroying stored information or clogging highly important network traffic corridors. Because of the danger of viruses, efficient measures must be taken to prevent them from intruding and spreading.

Virus scanning programs detect the presence of a virus by scanning program files for particular byte patterns. To be effective, this kind of check should be applied to all imported code imports, particularly from sources that may be considered suspicious (that is, code entering from non-secure networks and code installed from any storage medium, such as floppy disk or CD-ROMs, including backups). For further security, all files on all hard disks should be scanned regularly.

It is difficult to prevent unknown viruses from entering the protected network. However, with adequate protection, they can be quickly identified when they start their activity, especially if trying to infect other program files or hard disk boot records. Therefore, an efficient anti-virus program should be updated regularly to ensure that the virus data are current, and the program must be able to detect these kinds of events on all threatened hosts.

Because vendors' programs use different virus detection algorithms, it may provide an extra measure of security by simultaneously employing multiple anti-virus programs from different vendors. Therefore, even if a virus goes undetected by one program, it may not be able to hide from another. This may provide extra protection at the network boundaries and the workstations.

2.4 An infrastructure for public key technology

The explosive growth of the Internet has changed many basic assumptions regarding communication and business. The adoption of Internet technologies such as Virtual Private Networks (VPNs), intranets, and extranets, offers an attractive alternative to traditional methods of communication. However, organizations looking at taking advantage of Internet openness should understand the inherent security risks, such as unauthorized user access, data tampering, and eavesdropping.

What are eavesdropping and tampering?

Eavesdropping: Information remains intact, but its privacy is compromised. For example, someone could learn a credit card number, record a private conversation, or intercept classified information.

Tampering: Information in transit is changed or replaced and then sent on to the recipient. For example, someone could alter an order for goods or change a person's resume.

In order to utilize the full potential of the Internet, a strong security solution is essential. This solution must begin with strong authentication of users, provide encryption of all traffic from the application to the user, and control access to all information. Secure e-mail, Web access, e-commerce, VPNs, and extranets require strong security that provides confidentiality, authentication, access control, data integrity, and accountability. However, it also needs to establish, maintain, and protect a trust relationship for trusted e-business. *Certificates* and *public key cryptography* are emerging as the preferred enablers of strong security.

Public Key Infrastructure (PKI) is the term generally used to describe the mechanisms, entities, policies, and relationships that are employed to retrieve cryptographic keys and to associate public cryptographic keys with their owners. It consists of the programs, data formats, communication protocols, institutional policies, and procedures required for enterprise use of public-key cryptography.

The use of a PKI allows confidential and trusted electronic data transfer between computers around the world in a secure and transparent context.

Some basics about public key cryptography are covered in Appendix A, "Introduction to cryptosystem" on page 307.

2.4.1 PKI description and capabilities

A Public Key Infrastructure (PKI) consists of the technology and protocols, services, and standards for managing public keys. The main services a PKI offers are based on key registration (issuing a new certificate for a public key), certificate revocation (deleting a previously issued certificate), key selection (obtaining a party's public key), and trust evaluation (determining whether a certificate is valid and what operations it authorizes).

The policies and procedures required to manage keys are critical components of a PKI implementation. Key management involves:

- Verifying the identity of an individual when issuing keys
- Revoking, expiring, and renewing keys
- Key recovery, allowing the legitimate recovery of keys that have been lost or become unavailable

A public key infrastructure provides the technical framework necessary for implementing these basic security capabilities: Authentication of users, authorization to access systems and resources, digital signatures, and encryption of communications.

User authentication provides proof that “you are who you say you are”; digital signature identifies the originator of a message; encryption hides message content by encoding it and access authorization answers the question: “What can this user do and see?” Access authorization ensures that users are able to access only those applications and data to which the applied policy gives them legitimate rights.

2.4.2 PKI components and functions

A typical PKI consists of five basic components. Of these, three components carry out the core public key management functions: The *Certificate Authority* (CA), the *Registration Authority* (RA), and the *Certificate Repository* (CR). The remaining two components represent the entities that use certificates to enforce the security of their business: The *Subscribers* (also called *end entities*) and *Application/Users* (also called *relying entities*). These components are illustrated in Figure 3. Notice that this figure also shows multiple subscribers and application/users in the lower half. The figure also contains a Key Recovery Service component that might be part of a PKI, but it is not normally considered a basic component and, therefore, not further explained in this chapter.

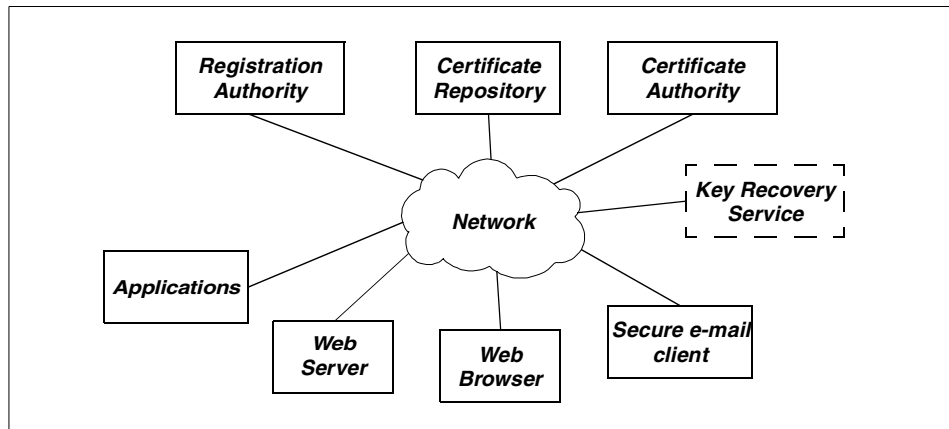


Figure 5. PKI components

The following sections explain the basic PKI components in more detail.

2.4.2.1 Certificate Authority (CA)

The Certificate Authority (CA) is the component that is responsible for the actual management of certificates. When the CA creates a certificate, it adds a unique “CA signature” to a subscriber’s public key, which ensures the integrity of the certificate. It cannot be modified without invalidating the signature; therefore, any certificate with a valid CA signature is sure to have the same contents as when it was signed by the CA. The CA is also responsible for revoking certificates that are no longer valid. An organization’s specific policies may allow automated certificate creation and revocation or may require that the creation or revocation of a certificate be performed by a particular person. This is sometimes referred to as the Registration Authority or RA.

The following example outlines the basic procedure for setting up a secure communication between two hypothetical users, Anne and Joe. Figure 6 diagrams the basic procedure for a Certificate Authority. The numbers in the figure correspond to the steps listed below.

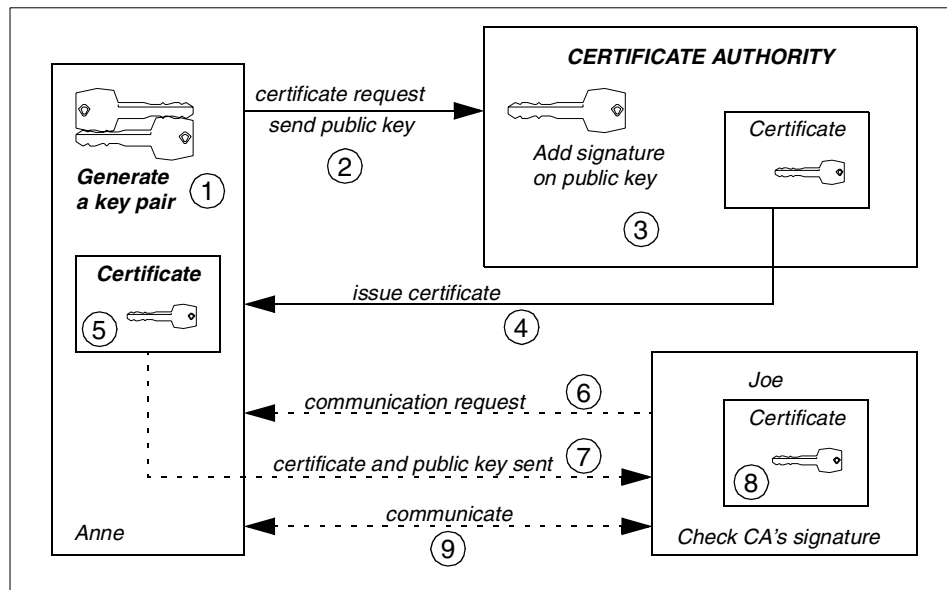


Figure 6. CA basic procedure

Suppose that the two users (Anne and Joe) want to establish a secure communication, and that Anne needs to hold her own certificate. The following steps outline the basic procedure for issuing certificates:

1. Anne generates her own key pair (even if the generation process could be performed by the CA).
2. Anne requests a certificate from the appropriate Certificate Authority and also sends her generated public key. This information is encrypted with the CA's public key.
3. The Certificate Authority decrypts the data and verifies Anne's identity. Depending on the policies in effect, the certificate authority may even require Anne to prove her identity in person before agreeing to issue the certificate. This identity verification is sometimes handled by a separate Registration Authority. Then, upon positive verification, the certificate is built by the CA, which adds to Anne's public key all the other information needed, including signing it with the CA's private key.
4. The Certificate Authority then issues the certification and sends it to Anne by encrypting the data with her public key.
5. Anne receives the certificate, decrypts the data, and stores her certificate.

Once Anne holds her certificate, she can handle the secure communication with Joe. The basic procedure is as follows:

6. Joe sends a communication request to Anne.
7. Anne transmits her certificate to Joe (containing her public key).
8. Joe stores Anne's certificate.
9. Joe and Anne can now start a trusted communication using the public key stored in her certificate.

Note

The way Joe gets Anne's certificate can vary. In fact, the certificate can be stored in a publicly accessible place from where it can easily be retrieved without asking the subscriber for a copy. It is important to understand that a certificate is not a secret at all; therefore, it is a good design to store certificates in public directories for easy retrieval.

Certificate chaining

In some organizations, a CA can either publicize its public key or provide a certificate from a higher level CA attesting the validity of its public key. This last solution is the base for a hierarchy of CAs. The use of a CA's hierarchy could be due to the certificate base (that maybe is too large for a single certificate authority (CA) to maintain). Also, there may be geographical separations between organizational units, or different issuing policies may need to be applied to different sections of the organization.

An example of a CAs hierarchy is illustrated in Figure 7. The delegation of the responsibility can be pursued by setting up subordinate CAs. The X.509 standard includes a model for setting up a hierarchy of CAs. In this model, the *root CA* is at the top of the hierarchy and issues a self-signed certificate. The CAs that are directly subordinate to the root CA have CA certificates signed by the root CA. CAs under the subordinate CAs in the hierarchy have their CA certificates signed by the subordinate CAs, and so on.

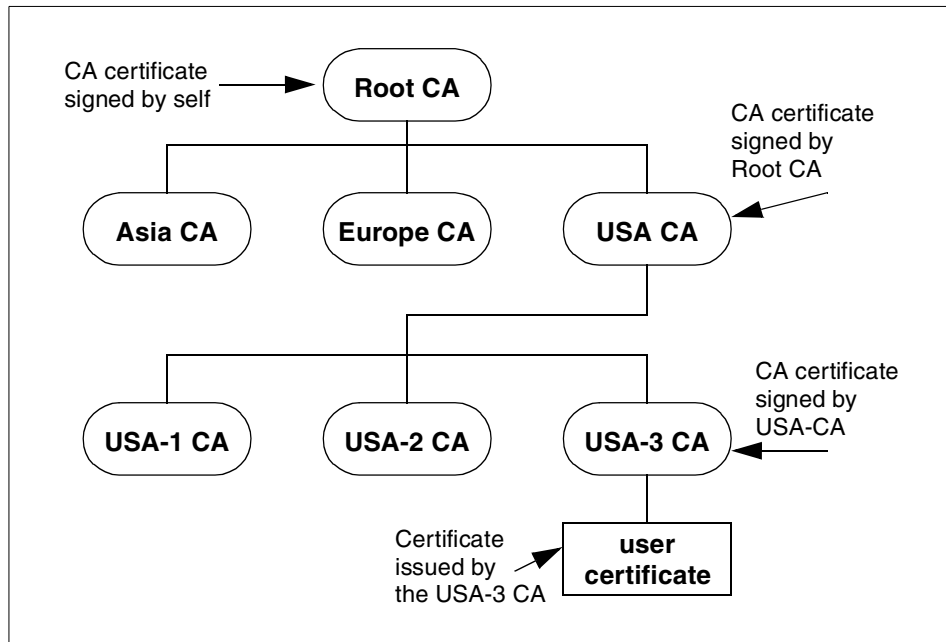


Figure 7. Example of a hierarchy of certificate authorities

A certificate chain consists of a certificate, the certificate of the CA that signed the certificate, the certificate of the CA that signed the CA certificate, and so on. A certificate chain ends with the CA certificate of the root CA.

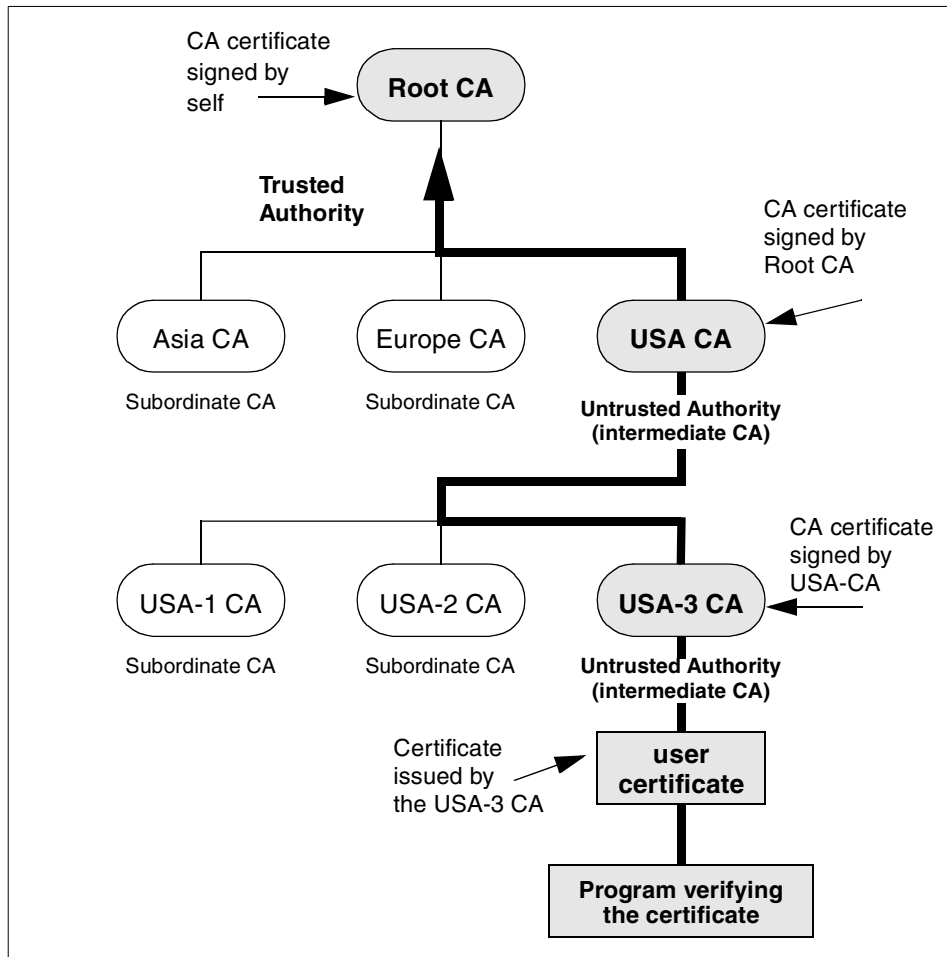


Figure 8. Example of a certificate chain

As illustrated in Figure 8, a certificate chain allows the tracing of a path of certificates from a branch in the hierarchy to the root of the hierarchy. In a certificate chain, the following occurs:

- Each certificate is followed by the certificate of its issuer.
- Each certificate contains the name of that certificate's issuer, which is the subject name of the next certificate in the chain. In the example shown in Figure 8, the CA certificate for USA-3 CA contains the name (signature) of the authority, USA CA, who issued the certificate. USA CA is also the subject name of the next certificate in the chain (the CA certificate for USA CA).

- Each certificate is signed with the private key of its issuer. The signature can be verified with the public key of the issuer's certificate, which is the next certificate in the chain. In the example above (Figure 8), the public key in the CA certificate for USA CA can be used to verify the signature in the CA certificate for USA-3 CA.

The root authority's certificate is self-signed. That is, it is signed using the private key corresponding to the public key in the certificate. A self-signed certificate has no authentication value on its own.

Because root CA certificates are self-signed, you must only load these certificates from a trusted source.

Evaluating a certificate may require searching through a long chain of CAs. Ideally, every CA in the path would support any policy requirements associated with the validation. Such policy-based certificate path validation is not yet a standard part of CA functionality. However, some vendors are beginning to support it, and work is proceeding to define standards in this area.

Certificate Revocation Lists (CRLs)

Certificate Revocation Lists (CRLs) provide a centralized source for accessing information about the validity of a certificate. Certificate Authorities regularly publish lists of certificates they have issued but have been revoked. The CRLs are ideally placed at easily-accessible points on the Internet so that a PKI-enabled application can check whether the certificate has been revoked. If a certificate is listed in a CRL, the application will know that the certificate has been revoked; otherwise, if the certificate is not listed, the application can normally assume that the certificate is valid and is safe to accept.

The most obvious problem with a list approach is the manageability of a very large number of certificates as the CRL grows. Some other approaches have been tried: The most common is splitting the CA's lists into smaller sublists, which is currently available with v2 CRLs (X.509 v2 CRL, or RFC 2459). Also, "delta" lists of CRLs and techniques for online verification are other approaches that are being investigated by certificate issuers, and standards are being worked on.

2.4.2.2 Registration Authority (RA)

The Registration Authority (RA) is an optional PKI component that can be used to manage identity verification as a part of the certification process. It is considered to be subordinate to the CA. A user requesting a certificate may be required to prove his or her identity in accordance with the policies of the

organization issuing the certificate. (One possible point of confusion is that an RA may also be the person who performs the identity verification; in this book, this person will be referred to as the RA Administrator.) For example, to issue a certificate, a user may be required to physically go to an RA Administrator and show the identification as required by the organization's policies. Then, the RA Administrator can approve the certificate request using the RA program, after which, the certificate file is sent to the user.

The RA program provides a point of isolation where an organization's user authorization policies can be enforced, therefore, providing only the functions required for creating, approving, changing, and revoking certificates. In larger networks with many certificates in use, there may be more than one RA in use.

2.4.2.3 Certificate Repository (CR)

Another PKI component is the Certificate Repository, which is a simple directory service that allows the certificates and CRLs to be stored in and retrieved upon demand. This component is often implemented as part of a general-purpose directory service, such as LDAP (see Appendix B, "Introduction to LDAP" on page 321), or it might be a PKI-specific function possibly included as part of the CA component.

2.4.2.4 End entities

The subscriber (end entity) is the holder of a certificate. In accord with an organization's local policy, the certificate functions in binding a key pair with an identity. The identity could be that of a human user, or it could be that of an information resource (for example, an application server). The policies that surround the management of certificates and the PKI functions carried out by the other PKI components are all focused on providing the subscriber with certain security services. Often, mutual interaction is needed between multiple human users (for example, e-mail exchange), between human users and information resources (for example, client/server applications), or between multiple computer processes (for example, virtual private network endpoints). In these cases, each entity act as both the subscriber role and as the application/user.

2.4.2.5 Relying entities

The applications and users that use certificates to interact with the subscribers that own the certificates will use them to obtain whatever security services are appropriate given the purpose of the certificates. The term "applications/users" is used because both applications (for example, Web servers) and human users (for example, e-mail correspondents) can obtain security services from certificates. The applications/users are also known as

the “relying entities” since they rely on certificates to know the identity of the subscriber. As mentioned before, in interactive applications, it is typical for the client (user) and the server to each play both the subscriber role (for example, be the owner of a certificate) and the application/user role (for example, rely on a certificate).

Besides these five components, if the PKI provides key recovery, there may also be another component called *key recovery service*. Key recovery is an advanced function required to recover data or messages when a key is lost.

In summary, the most important PKI functions are issuing and revoking certificates, creating and publishing CRLs, storing and retrieving certificates and CRLs, and providing trust and key life-cycle management.

Enhanced or emerging functions include time-stamping and policy-based certificate validation. In regard to the time-stamping function, it is important to highlight that in addition to the content and authenticity of a transaction. The exact time of the transaction can be important. For instance, a transaction may have to be submitted by a certain time to be valid. The solution to having trusted transactions is to combine digital signatures with time-stamps. This can be accomplished by having a PKI providing a time-stamping service.

2.5 Security toolkit for custom-built applications

The availability and proper function and implementation of an overall security framework is crucial for an organization. This includes vendor products and custom-written applications. It is very important that custom-written applications can leverage the security framework without “reinventing the wheel”. A comprehensive security framework, therefore, offers the necessary tools for the development of such applications.

The goal of a security *software development kit* (SDK) is to provide developers with an effective way to implement advanced security in networked applications. The use of a toolkit reduces the total cost of development by substantially shortening the development cycle.

The introduction of new technologies, the huge installed base of existing products and procedures, highly impacts the company’s environment and operations. A security toolkit could help minimizing this impact while implementing an application or an infrastructure.

The development kit that enables companies to customize and expand on their security implementation may also address standard security application

protocol interfaces (APIs) for building and deploying secure applications and managing the security environment.

2.6 Conclusion

The security disciplines mentioned in this chapter constitute the foundation of the IBM SecureWay FirstSecure framework. IBM SecureWay FirstSecure represents a security architecture and an integrated set of applications that enable customers to safely run their computers, networks, and e-business applications. These applications generally allow for the location, connection, and securing of information resources and overcoming the challenges of creating an infrastructure for identification and the integration of security applications. Such applications are based on commonly agreed upon industry standards defined by such groups as: IETF (Internet Engineering Task Force) and the Open Group and FIPS (Federal Information Processing Standard).

The IBM SecureWay portfolio encompasses several elements:

- IT security consulting
- Security services
- Embedded security
- Security gateways
- Secure software servers
- Cryptography
- Internet-based commerce (e-commerce)
- Directories/access control
- Security administration
- Network security
- Secure applications

The SecureWay architected approach addresses complexity, policy, and total cost of ownership by utilizing three groups of capabilities: Protection, direction, and detection. These elements are considered in the security requirements for e-business application life cycles (Table 1).

Table 1. Security requirements for business application life cycle

Stage	IT requirement	Security support
Evaluation	Which technology or product is the best?	Assurance: How secure is it? How can we be sure?
Implementation	How can this application be rolled out?	Administration: How will users be enrolled and the system be configured?

Stage	IT requirement	Security support
Deployment	Is the system ready to go on-line?	Availability: Will it be up and running and support the required transactions?
Transactions	What is the transaction risk? How will we mitigate it?	Authorization: How can unauthorized use be prevented?
Audit	What happened?	Accountability: How do we figure out who did what?

The goal is to avoid creating single islands of micro-solutions, but instead to follow an architecture. This approach makes clear what security problems are being addressed, provides guidance to designers and implementers, and establishes a baseline for implementations. The IBM SecureWay FirstSecure security architecture solves the challenges of enterprise security and is composed of five inter-related elements (Figure 9).

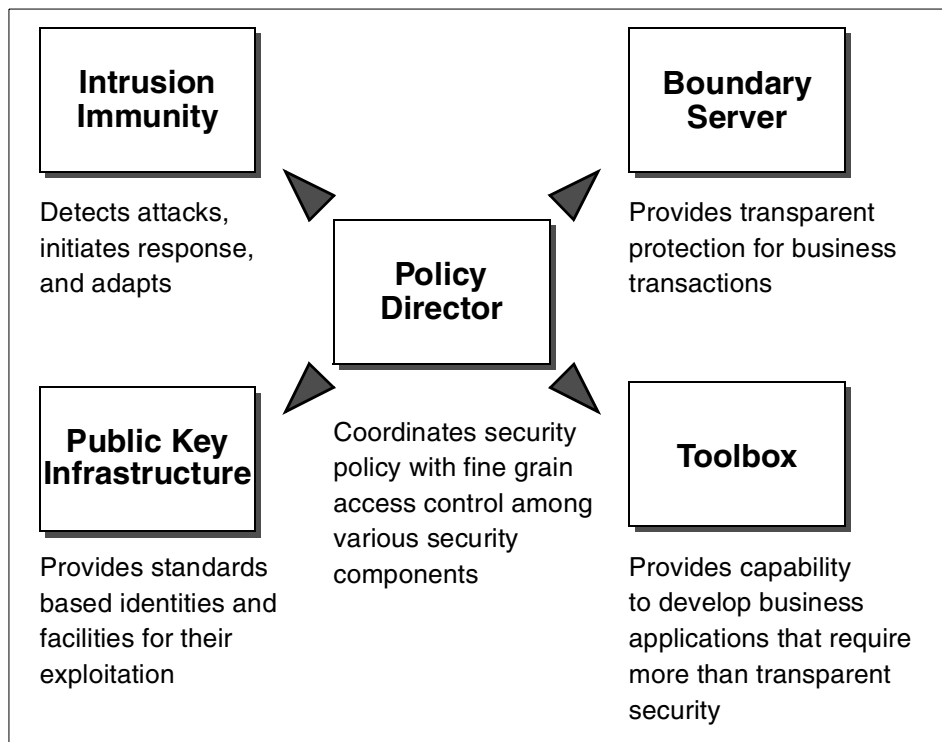


Figure 9. FirstSecure overview architecture

Part 2. IBM SecureWay FirstSecure - Concepts and building blocks

Chapter 3. The IBM SecureWay FirstSecure building blocks

The IBM SecureWay FirstSecure is a security framework that has been designed and developed to provide a comprehensive, yet flexible, solution for enterprise security. IBM SecureWay FirstSecure can provide scalable, end-to-end security for an organization desiring to implement an e-business strategy.

The IBM SecureWay FirstSecure is composed of five building blocks that interlock to create a secure network using state-of-the-art technology. These components are:

- *Policy Director* – The central component that coordinates security policies and provides fine-grained access control across the network.
- *SecureWay Boundary Server* – Provides the security necessary to allow an organization to confidently open the gateway between networks.
- *Intrusion Immunity* – Detects potential or actual attacks over a network or in systems and alerts administrators of such attacks.
- *Trust Authority* – Provides the essentials for a Public Key Infrastructure necessary for secure communication over the network and gives an organization the capability to use certificates for user authentication.
- *Toolbox* – Provides the tools necessary to create or migrate business applications to utilize the FirstSecure security framework.

Each of these components takes advantage of industry standards in order to provide a proven, secure solution. Some organizations may have invested many dollars in an incomplete security solution already. The various components of IBM SecureWay FirstSecure can be installed independently, thus, allowing a staged transition from any existing security solution to a more comprehensive security framework. This flexibility can reduce total cost since security can be implemented in a planned and controlled sequence.

The five IBM SecureWay FirstSecure building blocks are briefly introduced in this chapter and then further detailed individually in Chapters 4 through 8.

3.1 The Policy Director

The Policy Director is the central point for authentication and authorization in the FirstSecure architecture. It is a high-performance network application that provides state-of-the-art authentication, authorization, data security, and resource management capabilities. The Policy Director can be used in conjunction with standard Internet-based applications to create highly secure,

scalable, and well-managed access for Web-based (HTTP) and other TCP/IP-based network resources. This extends easily to a secure environment for sharing information on a computer network.

Policy Director is designed to allow authorized users the appropriate level of secure access to network resources and to deny access to unauthorized users. User identities are established through mutual authentication between client and server. Access to Web-based resources is granted (or revoked) based on the user's identity and detailed access control lists (ACLs) that protect the resources.

Enterprises that have intranet and extranet users who require a high-level of security when accessing intranet resources over the public (unsecured) Internet will benefit from Policy Director. Security policy is consistent, regardless of whether the user is working on a firewall-protected LAN or accessing the intranet from a public Internet Service Provider (ISP) account.

The Policy Director, as part of the IBM SecureWay FirstSecure framework, can be thought of as having the following characteristics:

- It is a central point of management for enterprise-wide security policies including users, protected network resources, and access control lists. It is centrally managed and (therefore) easy to administer.
- It is a standard, high-performance authentication and authorization authority for external and internal users and resources. It is application-independent and uses a standard authorization coding style that is language-independent (Authorization API).
- It consists of several components that can run on separate machines to best meet specific requirements. It provides a high level of security for its own, internal operations between the components.
- It acts as a Web server with a high level of policy-based authentication and authorization security. This high level of security can be extended to standard Web servers that may already be in place.

3.1.1 Functions

IBM SecureWay Policy Director provides a complete authorization solution for Web, client/server, CORBA, and legacy applications:

- *Fine-grained access control to Web servers* – Policy Director considers each server, file system directory, file, or Web page and executable program to be separate objects, allowing management down to this level as necessary.

- *Coarse-grained access control to Telnet and FTP servers* – Policy Director can secure TCP services, such as Telnet and FTP, across any number of servers.
- *Manage PD servers* – Centralized management can be accomplished by having one Policy Director Management Console manage several PD replica servers and/or completely separate servers that are logically connected, also known as being *junctioned*. This provides a single object space within which an operator can easily manage multiple servers, and users are insulated from the details of the topology.

3.1.2 Points of integration with other FirstSecure elements

Policy Director integrates with other elements of IBM SecureWay FirstSecure and acts as the central, fine-grained, access control manager for:

- *Firewall* – Firewall proxy users can be defined using the Policy Director. When LDAP is being used as the User Registry, the IBM Firewall, as part of the SecureWay Boundary Server (SBS) building block, retrieves the proxy user information from that registry and allows for a single point of user administration.
- *Trust Authority* – Certificates created and managed by Trust Authority, the PKI component of the IBM SecureWay FirstSecure, can be used for user authentication with Policy Director.
- *Toolbox* – Enterprise business applications that need to access other network-based resources can have their authorization controlled by APIs to interface with Policy Director.

3.2 The SecureWay Boundary Server

The SecureWay Boundary Server is used within the IBM SecureWay FirstSecure framework to implement the security policy on security enforcement points. It consists of the following software components:

- *IBM Firewall* – This software is installed on the dedicated firewall machine and contains IP level traffic filters, circuit level gateway (socks 5), and application proxies for SMTP, HTTP, FTP, and Telnet. It also provides many more functions, such as VPN (virtual private network), log management, and different types of authentication.
- *ACE/Server* – The ACE/Server from Security Dynamics allows strong authentication with SecurID tokens. It provides an alternative authentication method for environments where this is a requirement.

- *MIMESweeper* – The MIMESweeper, from Content Technologies Ltd., consists of two packages, the *MAILsweeper* for SMTP protocol and the *WEBSweeper* for HTTP protocol. MIMESweeper provides content screening for SMTP traffic by keyword, filters junk mail, and can restrict mail transfers depending on user, size, and content class. It restricts HTTP and FTP traffic by URL and can detect and block transfer of confidential data.
- *SurfinGate* – The SurfinGate, from Finjan Software Ltd., component can detect and block malicious JavaScript code, Java applets, and ActiveX controls.
- *Norton AntiVirus* – The Norton AntiVirus Suite (which is actually part of the intrusion immunity component) adds anti-virus measures for SMTP.

A combination of these components allows you to build a highly secure network boundary to protect the internal assets from outside threats.

3.2.1 Points of integration with other FirstSecure elements

The SecureWay Boundary Server integrates with other elements of IBM SecureWay FirstSecure:

- *Policy Director* – Firewall proxy users can be defined either locally or by using the Policy Director Management Console and a centralized User Registry.
- *Intrusion Immunity* – Intrusion Immunity agents can strengthen the security of the Boundary Server by detecting any potential and actual attacks to the Boundary Server machine(s) and dynamically create deny filter rules on the firewall.

3.3 Intrusion Immunity

IBM SecureWay FirstSecure provides a dual approach to the problem of *intrusion immunity* through the combination of intrusion detection and anti-virus protection. Both systems work together to protect from threats as they occur. This intrusion immunity works like a security alarm, taking action when a threat is detected.

IBM SecureWay FirstSecure provides network-based protection against threats through security agents that can be deployed wherever they might be a need, as, for example, when an organization's internal network is connected to the Internet. When a potential attack is detected, the agent can determine, based on the policies in effect for that organization, the appropriate response to that threat. If an immediate response is warranted, notification is sent to

the central security server, and appropriate actions are then taken depending on the nature of the attack.

3.3.1 Tivoli Cross-Site for Security

Tivoli Cross-Site for Security, as part of IBM SecureWay FirstSecure, is a network intrusion detection solution that allows remote agents to be deployed wherever needed. This allows real time inspection of network traffic passing between secure and non-secure areas.

If abnormalities are detected that suggest an attack is happening, the Intrusion Detection agents can gather information about these events centrally and trigger alerts that show up on a Tivoli Enterprise Console (if available). Tivoli Cross-Site for Security is able to recognize a wide range of different attack forms, such as IP port scans and floods, the "ping of death," detection of protocol specific information requests, and even login attempts. This is achieved via three different components that work closely together.

- The *Security Agents*, sometimes referred to as clients, implement the security policy. To do this, they directly monitor the network traffic, and when a potentially threatening event is detected, these agents determine whether an alert should be created based on the policies in effect.
- The *Management Server* provides a central location for intrusion-related security policies and their maintenance. The management server also distributes relevant policy information to the various agents. This server receives information about alerts and events from the security agents and then determines which actions to take.
- The *Management Console* provides the graphical user interface for controlling the management server. It provides functions for controlling the deployment of security policies and accessing event reports.

This distributed concept provides the scalability that allows a single intrusion detection system to monitor even complex networking environments.

3.3.2 Norton AntiVirus Suite

Virus protection is an important requirement for any networked computing environment. The Norton AntiVirus Suite provides comprehensive virus protection for clients and servers as well as e-mail content checking to prevent the delivery of potential viruses. Administrative tools are also provided to help manage virus definition files so that a knowledgeable user can perform simple updates of virus definition files. And, in the event that the Norton AntiVirus program does detect a virus, it provides three key functions for stopping the continued spread of that virus:

- Quarantine of the infected files in a safe location for further examination
- Scan and deliver any suspicious files to the Symantic AntiVirus Research center for analysis
- Scan other files to determine if a virus has infected other files, potentially leaving the virus still within the system

The Norton AntiVirus components supported within IBM SecureWay FirstSecure are:

- Desktop solutions for Windows 95/98 and Windows NT 4.0
- Server solutions for Windows NT 4.0, Novell NetWare, Lotus Notes, and Microsoft Exchange
- NAV for Internet e-mail gateways for NT
- NAV AntiVirus for Firewalls with a plugin for the IBM Firewall
- Administration solutions with Norton System Center and Network Manager

3.3.3 Points of integration with other FirstSecure elements

Although there is no direct interaction of Intrusion Immunity with other elements of the IBM SecureWay FirstSecure, Intrusion Immunity can greatly enhance the overall level of security of all components by passively monitoring and detecting any potential attacks.

3.4 Public Key Infrastructure

The IBM SecureWay FirstSecure Public Key Infrastructure (PKI) building block provides an integrated security solution allowing complete management of digital certificates. It can be defined as a trusted infrastructure for issuing and managing certificates, for developing high-trust applications, and for building policy-driven trust (see also 2.4, “An infrastructure for public key technology” on page 25).

IBM SecureWay FirstSecure PKI provides two levels of functions:

- Complete life cycle management of digital certificates, providing the capability to request, renew and revoke certificates, a registration authority to approve certificate request, and a certification authority to create and publish digital certificates and CRLs
- Enhanced registration capabilities for enabling businesses to register their trusted e-business entities online

The component is based on the Internet Engineering Task Force's (IETF) Public Key Infrastructure working group's specifications to ensure interoperability between different PKIs. The reference implementation for these standards, also known as Jonah, was jointly developed by IBM, Lotus, and Iris, and is available at the following WEB site:

<http://www.mit.edu/pf1>

The PKI builds on the Common Data Security Architecture (CDSA), a standard supported by the Open Group. CDSA can support multiple trust models, certificate formats, cryptographic algorithms, and certificate repositories. Because the reference implementation is based on a standardized framework, there is the possibility to choose between different trust models and key algorithms. Using the PKIX reference implementation as the base, the IBM SecureWay FirstSecure PKI has extended the functionality to support other requests for certificate services.

This component provides a set of software that enables the creation, management, storing, distribution, and revocation of certificates based on public key cryptography.

In the IBM SecureWay FirstSecure Framework, the PKI is provided through the *IBM SecureWay Trust Authority 3.1* product.

Trust Authority provides Internet applications with the means to authenticate users for trusted communications. Trust Authority supports the complete life cycle of digital certification including user enrollment, the initial issuance of digital certificates, certificate renewal, and certificate revocation. It also supports the immediate publication of certificates and certificate revocation lists, therefore, allowing applications to readily ascertain the authenticity of requests.

3.4.1 Points of integration with other FirstSecure elements

IBM SecureWay Trust Authority integrates with other elements of IBM SecureWay FirstSecure:

- *Policy Director* – Certificates created and managed by Trust Authority can be used for user authentication with Policy Director.
- *Toolbox* – The APIs that come with the SecureWay Toolbox can be used by application developers to develop their own applications that leverage a Public Key Infrastructure.

3.5 The SecureWay Toolbox

The toolbox component of the IBM SecureWay FirstSecure Framework is a *software development kit* (SDK) for developing secure applications for enterprises and for customizing or expanding on their security implementations. It provides the basic building blocks needed to develop secure business applications.

The toolbox provides support for stand-alone security application development and integrated application development through interfaces provided by the FirstSecure applications. Standardized interfaces ensure that appropriately developed applications can access cryptographic services and security management. The toolbox provides standard interfaces that security providers can use to add new cryptographic services, such as encryption algorithms, trust model services, and security data store services.

The toolbox contains the *IBM KeyWorks Toolkit* and the *IBM Key Recovery Service Provider Toolkit* (see 8.2, “Toolbox security services” on page 193 for more details). These two toolkits require detailed advanced designer or programmer knowledge of various security functions. The toolbox also provides application developers with standard *application programming interfaces* (APIs) for rapid building and deploying of secure applications and for managing the security environments (see Chapter 8, “The SecureWay Toolbox (TB)” on page 191).

The toolbox also includes thorough documentation for some component APIs. The documentation relates to each toolkit component and is available in a form consistent with that employed by each of the supported operating systems.

The main Toolbox objective is to provide a standard API set for each component that removes the requirement for the API user to be intimately familiar with the details of the underlying security mechanism.

3.5.1 Points of integration with other FirstSecure elements

Being a programmer's toolbox only, the SecureWay Toolbox does not directly interact with other components of IBM SecureWay FirstSecure. However, custom-written applications can, through the use of the SecureWay Toolbox, interact and utilize functions of the Policy Director and Trust Authority (PKI).

3.6 The system environment

The previous sections in this chapter introduced the building blocks of the IBM SecureWay FirstSecure framework and product offering. Although not considered a building block of the IBM SecureWay FirstSecure framework, it must be noted, at this point, that the FirstSecure framework must be supported by some very basic configuration guidelines that apply to every distributed systems platform to ensure security. Though the details are beyond the scope of this book, three such basic areas of concern are mentioned below. Please refer to the applicable product documentation and/or to appropriate literature for more details. Please also refer to 9.1, “Pre-installation considerations” on page 217.

3.6.1 Hardening the system platform

A security installation can only be as secure as the foundation it stands on. In the case of IBM SecureWay FirstSecure, the important parts of that foundation are the underlying operating system platforms. Many different operating system vulnerabilities have been discovered, and others will continue to be discovered. These vulnerabilities can be caused by commonly occurring situations, such as a programming error (*bug*), a misconfiguration, or even be the result of inherent protocol weaknesses. In any case, an important step in reducing the risk of getting attacked is to disable any network services that are not required for the key tasks to be performed. Doing so is called *operating system hardening*.

Operating system hardening should be applied to all security-relevant machines but is a must for all machines directly connected to a non-secure network. Some more details about operating system hardening can be found in 9.1.1, “Hardening the operating systems” on page 218.

Besides hardening according to common sense and rules, it is important to follow manufacturers support bulletins and other information that might help to counteract newly discovered security flaws. Other sources of related information are the CERT security alerts that can be found at:

<http://www.cert.org>

3.6.2 Network name resolution

The proper setup of the network name resolution, usually done by one or more Domain Name Service (DNS) servers, is another crucial part of the overall security setup, especially when an organization’s internal network is being connected to the external Internet. Internal network names and addresses should never be exposed to a corporate’s external network, except

for, and only for, the servers that need to be accessible from the external network. This is especially important as there might be a misconception about the systems that are installed on an isolated network between the internal and the external network (also called the demilitarized zone, or DMZ). In fact, some servers in the DMZ, such as systems needed for intrusion detection, do not need to be known outside an organization.

3.6.3 Physical design

The physical systems and network design supports a corporate security policy by limiting physical and network access to critical resources. Any system that is critical for security should be physically installed in an access-controlled room. The network layout and router configuration should be done in a way that they do not allow malicious network traffic tracing and connection attempts. The most popular counteraction, in terms of layout, is the use of a firewall between an organization's internal and the external network. Firewalls, however, are sometimes also used to secure critical parts of an internal network.

Special attention should also be spent on data backup and archiving. For example, system backup tapes contain all the data that are stored on a system and, if not properly handled, can be examined by an attacker on a separate system without any means of detection.

3.7 Bringing it all together

In every organization, there are different existing heterogeneous computing environments. FirstSecure elements can be integrated into such environments individually or in myriad combinations. In order to illustrate one complex environment that utilizes all of the FirstSecure elements and several different typical legacy system platforms, Figure 10 shows a complex network and system environment. Chapter 10, "Deploying FirstSecure in practical scenarios" on page 271 will further illustrate and discuss real-life scenarios.

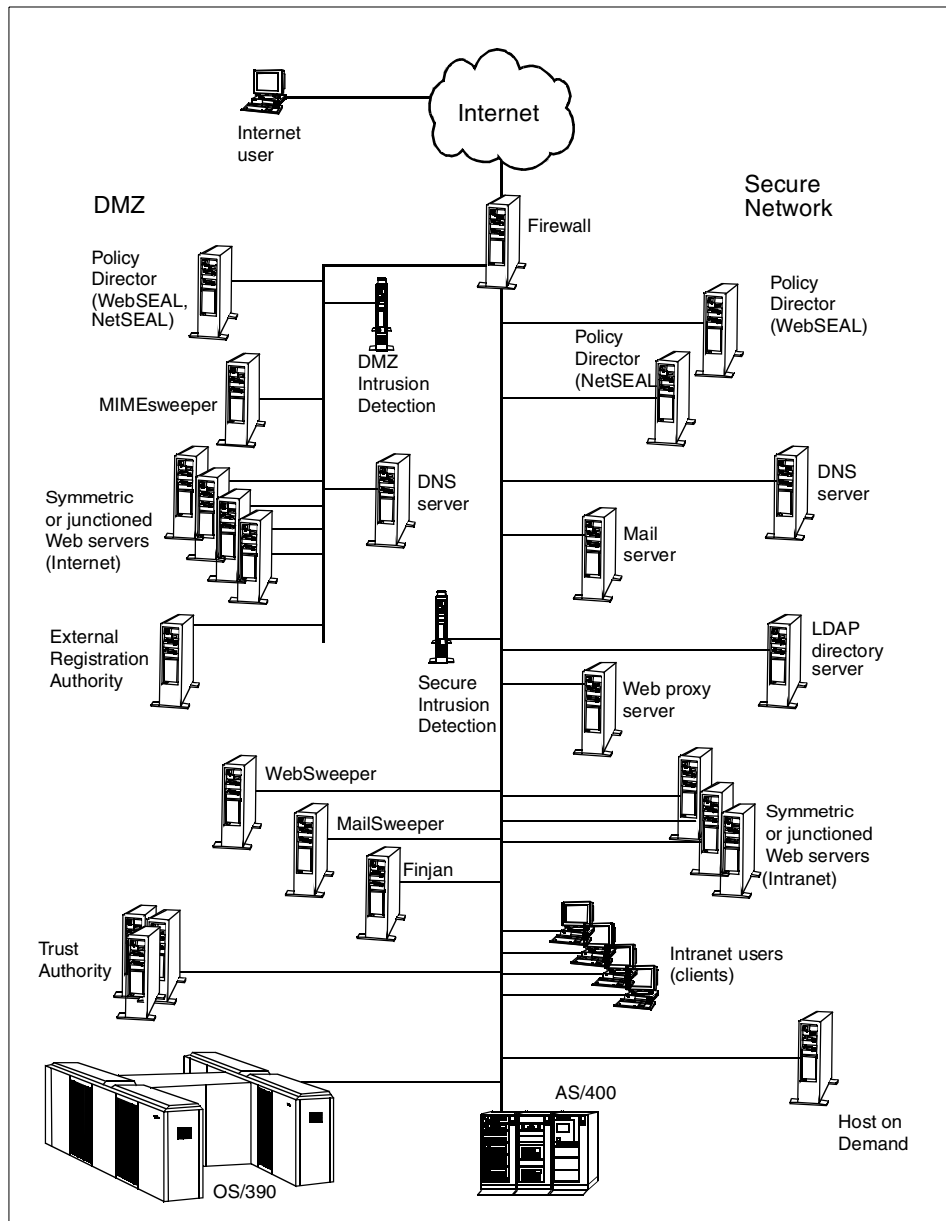


Figure 10. Network environment with IBM SecureWay FirstSecure

In addition to the functionality of IBM SecureWay FirstSecure, Version 2, as described in this redbook, IBM also announced some *statements of directions*, which are provided for your convenience here:

CORBA Support – IBM plans to deliver a Policy Director module that will provide single sign-on and authorization management for enterprise CORBA environments. It is planned to support secure interoperability between Object Request Brokers (ORBs) available from Inprise and IONA, providing users with the flexibility in their development and deployment of CORBA applications. IBM plans for the module to provide a CORBA Security Service to implement controlled delegation. This function is intended to provide the ability to pass user and server information with a transaction request, enabling extremely fine-grained access control.

MQSeries Support – IBM plans to deliver a Policy Director module that will provide customers with the ability to create authentication policies, data protection policies, and authorization policies for MQSeries.

Tivoli Integration – IBM plans to provide support that will enable customers to use Tivoli User Administration and Tivoli Security Management to manage Policy Director users and access control policies.

Enhanced Single Sign-On support – IBM intends to enhance Policy Director's Web credential mapping service to support PKI and other stronger authentication systems.

SecurID Token Support – IBM plans to enhance Policy Director to provide support for authentication using Security Dynamics SecurID tokens without requiring any customer level customization.

AIX 4.3.3, Solaris 2.7 – IBM intends to provide support of FirstSecure on AIX 4.3.3 and Policy Director support on Solaris 2.7.

Chapter 4. The Policy Director (PD)

The IBM Policy Director is the core element of the FirstSecure framework for providing centralized authentication and authorization services to access enterprise resources. The Policy Director provides standard authentication and authorization processes for business network systems without replacing existing systems. Because of the key role of authorization in Web based business applications, the Policy Director is considered to be in a central position as illustrated in Figure 11.

Policy Director integrates with existing legacy and emerging intranet infrastructures to provide a centralized policy management capability. This chapter discusses important concepts necessary to understand, plan, and implement Policy Director.

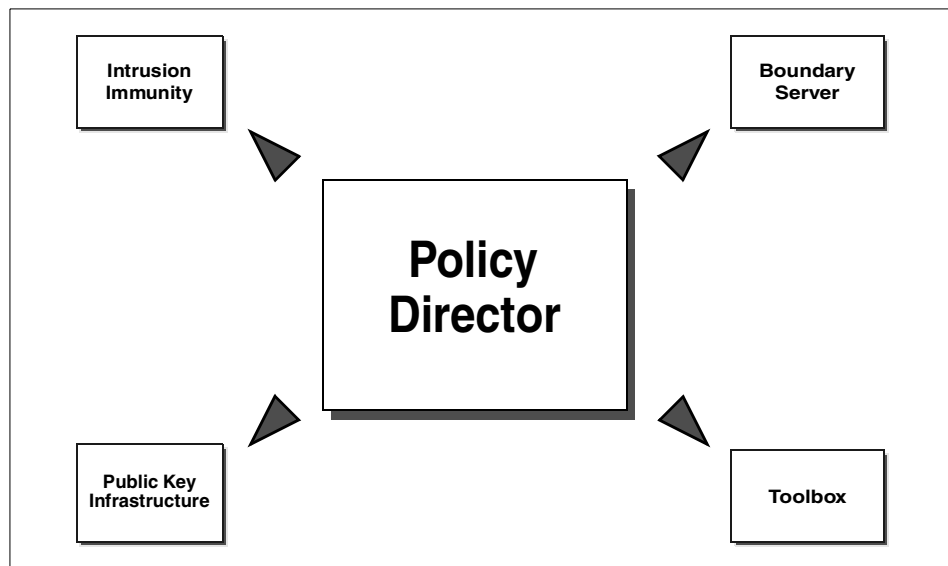


Figure 11. Policy Director in the FirstSecure framework

Policy Director enhances the ability to manage access to network resources as well as centrally define the authorization characteristics attributed to these resources. This work is performed on a *Management Console* which includes a graphical user interface (GUI), associated with a Policy Director server.

When requesting a service from a network server, such as a Web server, client computers access the Policy Director system transparently for authentication, and if they have the appropriate authorization to access the

resource, they are connected to it. The target resources may be a Web page, a CGI application, or another application that is accessed through a more comprehensive framework, such as IBM WebSphere, which utilizes CORBA technology. In either case, logging may be defined and applied by the Policy Director server.

The major functions of the Policy Director consist of authentication, secure communication, authorization, Web single sign-on, and centralized management. These functions are provided by the following core components:

- A *User Registry*, either held in an LDAP directory (see list of prerequisite services) or on a DCE Security server.
- A *Management Server*, utilizing the user registry and an ACL database.
- A *Security Manager*.
- An *Authorization Server*.
- A *Management Console*.
- A *Credential Acquisition Services (CAS) Server*, which is optional.
- A *Standard Web Browser* (with or without SSL support) or *NetSEAT* client.

In addition, the following are prerequisites providing essential services for the Policy Director components:

- *Distributed Computing Environment (DCE)*, which includes a DCE Security Server and a DCE Cell Directory Server.
- An *LDAP* (Lightweight Directory Access Protocol) server, which is optional.

LDAP is not required, but highly recommended. The IBM Policy Director can be run using the DCE support supplied with the product. However, there are advantages to using LDAP as we will see later.

If LDAP is chosen, it requires a suitable *Web server*, such as the IBM HTTP Server.

These components together protect the key application resources that businesses depend on and will be examined in greater detail throughout this chapter. Installation considerations beyond the Policy Director Installation Guide are discussed in 9.2, "Policy Director" on page 221.

Before we consider applying Policy Director, we will describe the architecture of the system.

4.1 Policy Director architecture

Policy Director is a set of interoperating programs that work together to provide authorization control to network computing resources. In its simplest form, Policy Director can be considered a server on a network that is utilized to provide access control to other information repositories. Figure 12 illustrates this and includes an illustration of the different protocols used to communicate between Policy Director and other servers. Some of these services are implemented by the Distributed Computing Environment (DCE), which is discussed in more detail in 4.1.7, “Distributed Computing Environment (DCE)” on page 58. Software services are programs (also called *daemons* on UNIX systems or *services* on Microsoft NT) that can exist on a single computer or be spread out among several computers in the network for efficiency reasons.

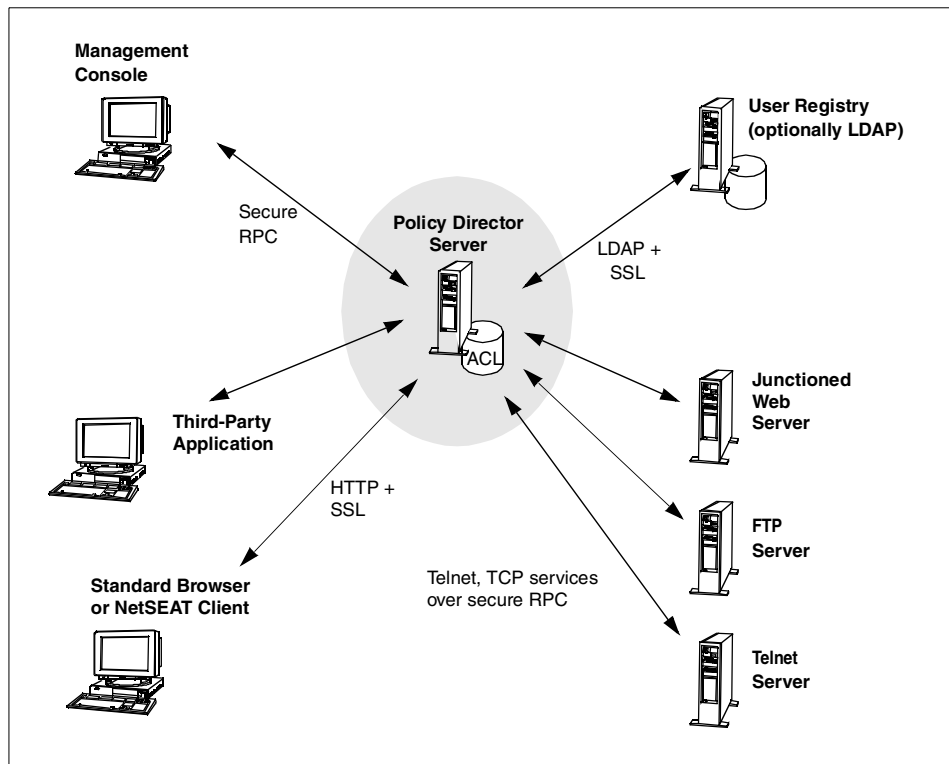


Figure 12. Relationships between Policy Director elements and network services

The Policy Director authorization service depends on two databases containing the information necessary for the authorization, decision-making

process. These two databases are the *User Registry*, which may be held in an LDAP directory or on DCE, and the *Authorization Policy Database*, which is stored as a local database file on the Management Server. Administration of the servers and databases is performed through the Policy Director Management Console.

4.1.1 User Registry

The User Registry contains an account entry for all valid users (principals) who participate in the secure domain. This database is used in two important areas:

- Defining user information for authentication services.
- Defining the groups to which the principals belong and the roles the principals can assume. This information is considered during an authorization decision.

The User Registry is held in either an LDAP directory (provided by the IBM SecureWay Directory Server) or on the DCE Security Server, called the *DCE registry*. LDAP is assumed to be the User Registry database at installation time unless you specify otherwise. The use of an LDAP-based registry has a number of advantages over a DCE registry, such as its scalability and better interoperability features. Also, it can be assumed that further integration work within IBM SecureWay FirstSecure will focus on LDAP rather than DCE. The use of LDAP, however, does not mean that DCE is not required. DCE is required as a security framework for the Policy Director components to work together.

4.1.2 Management Server

The Management Server maintains the Master Authorization Policy Database for the secure domain. The Management Server replicates this master Authorization Policy Database throughout the secure domain and is responsible for updating the replica databases whenever a change is made to the master database.

The Management Server also maintains location information about the other Policy Director and non-Policy Director servers operating in the secure domain.

4.1.3 Security Manager

The Security Manager applies access control policy based on information from an Authorization Policy Database. The Security Manager includes a

NetSEAL component for coarse-grained TCP/IP access control and a WebSEAL component for fine-grained HTTP and HTTPS access control.

4.1.3.1 NetSEAL

NetSEAL provides a Virtual Private Network (VPN) solution for securing TCP/IP communication. NetSEAL performs access control based on the destination port and identity of the client. NetSEAL is the security solution for authorizing and securing traditional network services, such as Telnet and POP3, as well as various application packages, for example, database systems and network management tools.

4.1.3.2 WebSEAL

WebSEAL is a high-performance, multi threaded Web server that accepts HTTP, HTTPS, and NetSEAL client requests. WebSEAL manages access control for such resources as:

- URLs
- URL-based regular expressions
- CGI programs (in PERL, C, C++)
- HTML files
- Java servlets
- Java class files

Although WebSEAL can function as a normal Web server, it is typically used for authentication and authorization services to other, so called junctioned, Web servers.

4.1.4 Credentials Acquisition Server

The Credentials Acquisition Server, or CAS, is essentially a database that allows the mapping of user identities from a non-Policy Director format into a recognized Policy Director format. It has a key role to play in certain kinds of authentication as will be discussed in 4.2.1, "Authentication" on page 61. It is also used to allow the expansion of Policy Director to accept authentication information from other security systems.

4.1.5 Management Console

The Management Console is a graphical Java application used to securely manage security policy and all Policy Director components. From the Management Console, an administrator can add, modify, or delete users and groups and apply ACLs. The Management Console also allows for managing the user registry and the primary Authorization Policy Database. The Management Console uses either the NetSEAL client (on Windows NT, 95

and 98) or a DCE client to perform these management tasks through secure communications channels.

4.1.6 DCE Security Server

The DCE Security Server is a server that maintains critical information about the Policy Director environment and manages the secure authentication amongst the Policy Director components. DCE also provides secure RPC communications between many of the Policy Director components. If desired, it can also be used to store the User Registry as outlined in the preceding discussion.

DCE provides for replication of the Security Server to prevent a single point of failure. The Master Security Server is then responsible for updating all replica servers (and their copy of the registry) whenever a change to the master registry occurs.

DCE, as an underlying framework, is more than just a Security Server; it is an important enabling technology that is explained in more detail in the following section.

4.1.7 Distributed Computing Environment (DCE)

Policy Director specific services utilize DCE to facilitate secure communication between its functional elements on a server as well as between other management console(s) and servers running HTTP, Telnet, or other TCP services that this Policy Director server is managing. This layered relationship is illustrated in Figure 13.

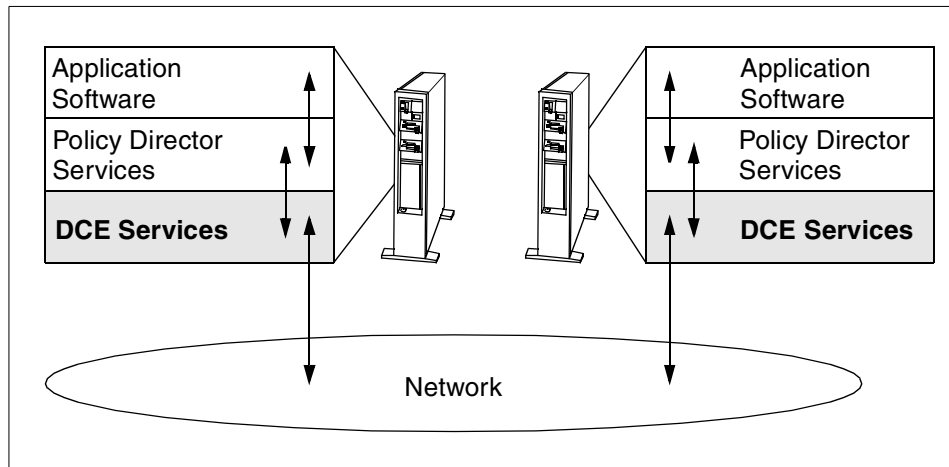


Figure 13. Policy Director software services layers

Before continuing, we discuss DCE in the context of securing elements of Policy Director. DCE is a vendor-independent, highly secure platform that provides basic security functionality to applications and other security products, such as IBM Policy Director. DCE is standardized by The Open Group (<http://www.opengroup.org>) and is available as separate products from all major vendors. DCE, however, is bundled with IBM SecureWay FirstSecure and does not need to be ordered separately. For more information beyond the short introduction given here, please refer to the respective product documentation or to The Open Group. IBM DCE, as included with the Policy Director, comes with a comprehensive set of online documentation that can optionally be installed.

DCE is a key element that is used to increase the trust between Policy Director server(s), Management Consoles, and other servers that the Policy Director is protecting.

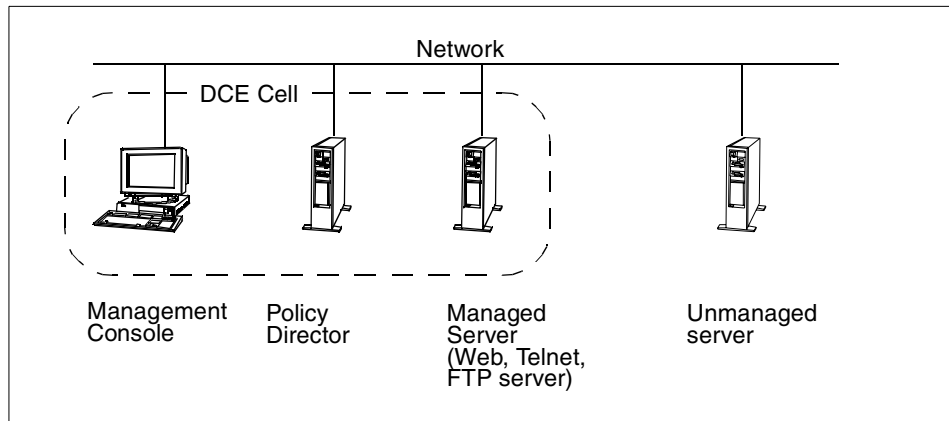


Figure 14. DCE cell supporting Policy Director

In DCE, the environment that is protected is called a *cell*, which defines a secure DCE domain (see Figure 14). A DCE cell includes all systems that are configured to be part of that cell. DCE uses and supports Kerberos authentication, used to provide mutual authentication and, ultimately, trust between computers in a DCE cell. Policy Director uses this Kerberos-based service for the various authentication processes between its components, for example, between remote management consoles and the Policy Director Management Server.

After successful authentication, clients and servers can encrypt their communications using secure DCE RPC (Remote Procedure Call) to assure privacy and data integrity as they go about their business. Client applications, such as HTML browsers, Telnet, or other TCP applications, can access a Policy Director secure domain by using the Policy Director NetSEAT client as a proxy.

In the FirstSecure framework, a new DCE cell is typically configured for use by the Policy Director. Because of its underlying services, the DCE cell (a DCE cell consists of at least a DCE Security Server and a Cell Directory Server) needs to be in place before Policy Director can be installed and configured. DCE, however, is utilized by Policy Director transparently to the user and administrator and does not normally require any special attention.

4.1.8 Bringing it all together

The interrelationships between some of the components outlined above are illustrated in Figure 15, which represents a refined picture of the illustration in Figure 12 on page 55.

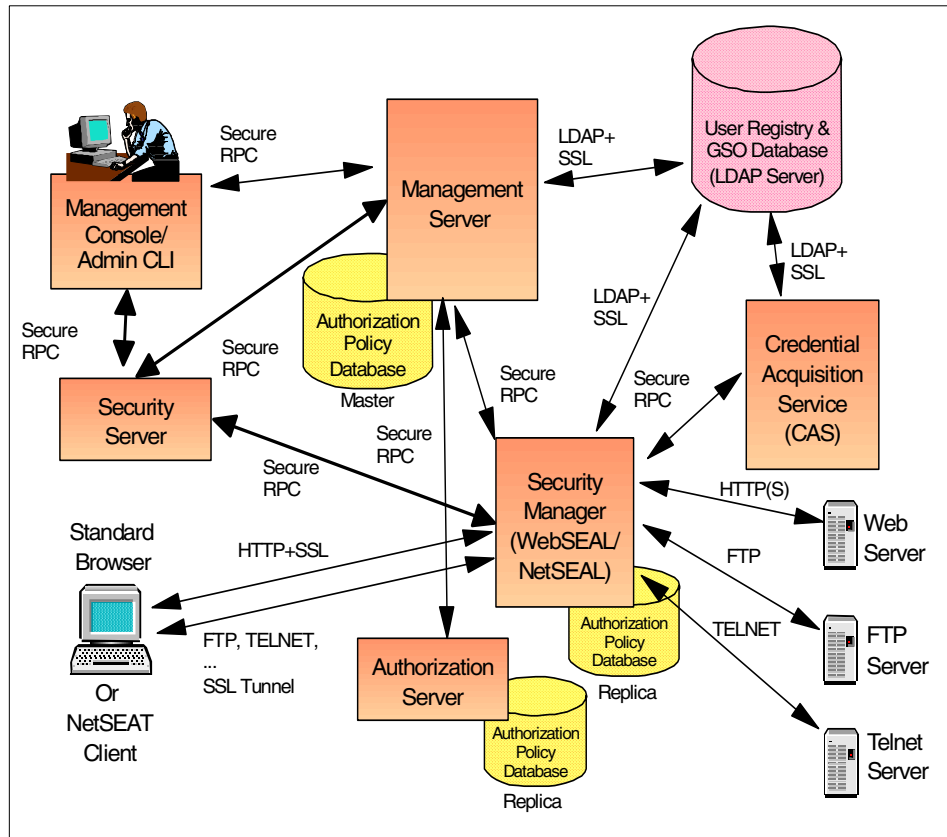


Figure 15. Policy Director architecture

4.2 Authentication and authorization

At an abstract level, Policy Director is composed of a set of services that interoperate to provide the two principal functions of *authentication* and *authorization*.

Figure 15 on page 61 illustrates the overall architecture of Policy Director and the communications that take place between the various components. Refer to this diagram as an aid in the discussions that follow.

4.2.1 Authentication

Authentication refers to the process of identifying a principal (user or process) requesting access to a secured resource and returning the *credentials* for that principal. Credentials are used by Policy Director in the authorization phase

(see next section). By default, in the FirstSecure framework, authentication is assumed to be provided using information held in an LDAP directory available on the network. This is not the only possibility; the User Registry can, as we have mentioned before, be held in DCE. It is even possible to use authentication information held in other security environments, such as OS/390 RACF, by using the mechanisms provided by the Credentials Acquisition Server (CAS). However, in the FirstSecure framework, and for maximum scalability and flexibility, LDAP is the preferred, scalable mechanism.

Policy Director can support three kinds of authentication:

- *Certificate-based authentication* – For users coming in through a standard SSL-enabled browser, whether within a company's intranet or from the external Internet.
- *User ID and password* – For user ID based authentication.
- *Kerberos* – For network authentication using secure DCE Kerberos.

SSL can support server-only or server and client side authentication, and it supports the X.509v3 certificates issued by IBM SecureWay Trust Authority and IBM Vault Registry as a standard.

User ID and password can be used in basic authentication or forms-based login mode over SSL.

Kerberos is used for authentication amongst the Policy Director servers and between the servers and the Management Console. It is also possible to use Kerberos for user ID/password authentication if the User Registry is held in a DCE database.

4.2.1.1 An authentication scenario

A user, whether inside or outside the organization's secure domain, wants to access a secured resource within the secure domain. The user in our scenario, is attempting to access a URL or a Web-enabled application using an SSL-enabled browser. We will use this basic scenario to discuss the ways in which Policy Director authenticates a user. We will also briefly mention the differences if the user is not using SSL.

Before attempting to follow this scenario through, if you are unfamiliar with the principles of SSL handshaking, please refer to Appendix A, "Introduction to cryptosystem" on page 307, or to the relevant sections in the *IBM SecureWay Policy Director Administration Guide*, available on the product CD-ROM or at:

<http://www.ibm.com/software/security/policy/library>

In our scenario, the user's first access point into the secure domain is WebSEAL by accessing a URL that points to it. There are several ways in which Policy Director can authenticate a user via WebSEAL. The important point is to take the information that the user sends to WebSEAL and obtain from it a recognizable Policy Director user identity, known as the Policy Director *Distinguished Name*, or DN. This DN will then be used to obtain the relevant credentials.

The possible authentication methods arising in this scenario are:

1. A user typing in a user ID and password via WebSEAL. This further divides into:
 - *Basic authentication*, the user types a user ID and password in a browser pop-up dialog window.
 - *Forms-based authentication*, where the user types a user ID and password into an HTML form provided by WebSEAL.

This latter method allows a user to log out of the requested application session without closing down their Web browser. See later discussion for details.
2. Full client-side certification using X.509 certificates. This requires the use of a CAS or another credentials acquisition and mapping process. The CAS may require customization to support certain authentication methods.

The following discusses these authentication methods in greater detail. As the two variations of user ID/password authentication are largely identical, we will deal with those together and point out the differences.

User ID/password authentication

When the user requests access to the secured resource, the following steps take place:

The client (on the user's machine) requests a connection to the WebSEAL server by requesting access to a URL that points to that server. If this is a request to a secure site using SSL, the WebSEAL server returns its server-side certificate. If the client accepts the server's certificate (which is most often done automatically by the browser), the secure SSL connection is established between the browser and the server.

The server then issues a challenge. This challenge prompts the client browser to present a request to the user to supply a user ID and password. In basic authentication, this will just be a prompt. In the case of a forms-based login, an HTML form is produced by WebSEAL and displayed by the browser asking for the same information.

Once the user ID and password are entered, WebSEAL then uses these to access the User Registry and obtain the user's credentials.

The major difference between basic authentication and forms-based login is that, in basic authentication, the user ID and password are cached in the user's browser to be used for each subsequent access request. This means that the user cannot actually log out unless the browser is closed. In the case of forms-based login, the user ID and password are not cached in the browser, and the user can log out using the `pkmslogout` command without closing down the browser. The `pkmslogout` command runs on the server, and is executed typically when the user clicks on a "log out" link from within the application.

SSL ensures secure transmission of the logon data (user ID and password) between the browser and the server. Without SSL, an attacker could easily intercept and copy this information.

X.509 client-side certificate authentication

In the case of X.509 client-side certification, the flow is rather different. The user's machine (the *client*) will have its own certificate to provide the user's identity.

The client requests a connection to the WebSEAL server. The server presents its own certificate, which is then used to encrypt subsequent transmissions. Once the client has accepted the server's certificate, the server requests a client certificate. The client sends its own certificate to the server. The server receives the certificate, and now it calls upon the CAS to do some work for it.

The CAS takes the certificate and strips out the Distinguished Name (DN) contained in the certificate. *This is often different to the DN held for that user in Policy Director*, as the client may use its certificate for more than one purpose or may have several certificates for access to several applications. The job of the CAS is to return to WebSEAL a DN that will be recognized by Policy Director, that is, a DN that matches an entry in the LDAP User Registry.

If the CAS finds a Policy Director DN mapped to the certificate DN in its mapping database, it returns this to WebSEAL. Otherwise, it returns the DN provided on the certificate.

WebSEAL then uses this DN to access the LDAP User Registry and obtain the credentials of the user associated with the certificate presented.

The CAS functions as a mapping agent. It also serves another important security purpose. It checks the Certificate Authority (CA) *Certificate Revocation List* (CRL) to ensure that the certificate presented has not been revoked.

Kerberos

DCE authentication using Kerberos can, in some circumstances, be used to authenticate a user login via user ID and password. This would only be used if GSSAPI (Generic Security Service API) tunnelling was required. In order to use this authentication method, two things are needed:

1. The User Registry must be held on DCE.
2. The user must be logging in from a NetSEAL client.

This authentication is used when NetSEAL is the entry point, such as when the user has requested FTP or Telnet access.

These methods are discussed in detail in the *IBM SecureWay Policy Director Administration Guide*, available on the product CD-ROM or at:

<http://www.ibm.com/software/security/policy/library>

Credentials and the Credentials Acquisition Service

The Policy Director Credentials Acquisition Service (CAS) allows authentication and mapping of user identity information (such as the X.509 certificate) to a Policy Director user identity. The Security Manager can then retrieve credentials for this Policy Director user identity from its LDAP User Registry. Credentials generally contain the user's identity and a list of the groups to which a given user belongs. This information can be shared amongst the Policy Director servers or accessed via an API.

Policy Director represents credentials with an internal data structure called an EPAC (Extended Privilege Attribute Certificate). This is an internal structure that the user does not need to know about. However, its existence has implications for the deployment of Policy Director.

One of the integration and interoperability considerations for Policy Director revolves around the concept of mapping this EPAC information to other systems that have different representations for users and credentials. This mapping is required when connecting to software systems in other computing environments, such as OS/390 and IBM WebSphere applications, to name only two examples.

The Policy Director Credentials Acquisition Service is a customizable component that can be used to extend the standard authentication mechanisms that are supported by WebSEAL. To put this more simply,

imagine a company's IT organization has legacy systems in which they have stored all their user information. The organization wants to use Policy Director as their central access control system. By using CAS, they can continue to use the information repository they have already established. CAS will, under the covers, convert the information to a suitable format for Policy Director. If the organization does not want to use CAS, the same end can be achieved by writing and installing a suitable custom application.

Another application for the CAS, as already explained in "X.509 client-side certificate authentication" on page 64, is to extract user identity information from X.509 certificates and convert it to a format understood by Policy Director.

4.2.2 Authorization

Authorization refers to the process of determining a user's rights to perform an operation on any given resource. Authentication provides Policy Director with credentials (the information held, for example, in the LDAP User Registry). The credentials are then used to determine what authorizations the user has.

Only authenticated users/clients can participate in gaining access to secured resources. A client request for a protected object is handled in the following manner:

1. Depending on the implementation, a secure communication channel is established between the client application and the Policy Director server (WebSEAL or NetSEAL, see explanations of these applications later).
2. The client authenticates to the Policy Director to establish proof of identity. Based on this identity, the Security Manager (WebSEAL, NetSEAL, or a custom enforcement agent) retrieves the user's credentials. Credentials define the groups and organizations to which the client belongs and the roles the client can assume.
3. For each access request, an authorization check takes place against a centrally managed and replicated *Authorization Policy Database*. This database contains the policy rules or templates known as *Access Control Lists (ACLs)*. Access control lists are enforced based on the client's credentials.
4. If permissions on the object are compatible with the user's credentials, Policy Director (or a custom enforcement process) then grants the access.

Authorization is probably the largest role played by Policy Director. This book addresses the elements of authorization in some detail. Full details of

authorization tasks and capabilities can be found in the *IBM SecureWay Policy Director Administration Guide*, available on the product CD-ROM or at:
<http://www.ibm.com/software/security/policy/library>

4.3 Authorization: Objects, ACLs, and enforcers

One of the most important concepts to understand before undertaking the tasks of authorization and administration is the concept of the *Object Namespace*.

This is referred to throughout this and other publications as *object space*, *namespace*, *object namespace*, and other permutations of these phrases. All of these refer to the same basic concept. There is a more detailed discussion of the Policy Director namespace in 4.7, “Policy Director namespaces” on page 96.

The resources that can be secured using Policy Director are represented in Policy Director object namespace as *objects*. When you apply authorization policy to a resource, you achieve this by setting up a *policy template* in the Authorization Policy database, and then applying that template to the resource in the object namespace.

Policy templates are also known as *Access Control Lists (ACLs)*, which are explained in some depth later in this chapter. ACLs are where an organization’s security policies are defined to Policy Director.

Authorization is achieved by matching a user’s credentials to an ACL. This is carried out by the *authorization evaluator*. The decision of the authorization evaluator has to be enforced, and the agent that enforces this is known as the *policy enforcer*. Two policy enforcers are WebSEAL and NetSEAL.

The remainder of this section briefly describes some of the main elements of Policy Director involved in authorization and administration including WebSEAL (see below), NetSEAL (see 4.3.2, “NetSEAL” on page 71), and NetSEAT (see 4.3.3, “NetSEAT” on page 71).

4.3.1 WebSEAL

The Policy Director WebSEAL Server is a high-performance, multi-threaded secure authentication and management HTTP 1.1 Web server. WebSEAL (as well as NetSEAL) can be installed on the same system as one or more of the Policy Director servers, but generally it would be preferable to put these components on a separate system for performance and security reasons. Here, we have presented a simplified model, and we will only discuss utilizing additional flexibility when we are discussing load balancing. Please refer to

10.1, “Policy Director” on page 272 for a guide to the elements of DCE and Policy Director in a practical scenario. In addition, the *IBM SecureWay Policy Director: Up and Running*, helps you understand how to install each element.

The Policy Director WebSEAL Server can be used to service Web requests from secure NetSEAT clients or standard browsers and can be used to service normal (unsecured) HTTP requests as well as secured HTTPS requests. The Policy Director WebSEAL Server supports the CGI 1.1 interface, server side applications, and auditing.

The information that a user accesses can be held on the WebSEAL Web server itself, or more commonly, will be held on *back-end servers*, which are protected by the WebSEAL server (in this case, the WebSEAL server is acting as a *proxy* to the back-end servers). The Web-based information supplied by a collection of Policy Director WebSEAL Servers and third-party (back-end) Web servers can be configured into one logical Web object space through the use of *smart junctions*. Smart junctions allow you to mount the information on one Web server into the Web object space maintained by a Policy Director WebSEAL Server. The back-end servers are, therefore, called *junctioned servers* (see Figure 12 on page 55).

This capacity to exploit smart junctions to manage multiple servers is one of the unique capabilities of Policy Director. Not only can the Web pages residing on the Policy Director server be managed, but *the complete Web page file systems of other Web servers* can be connected, or junctioned, to the Policy Director Web space. The root directory of the other servers are mounted at any point in the Web space.

This feature allows Policy Director to manage authorization of multiple servers from one graphical user interface.

Also, Web users do not realize that they are referring to information stored on different back-end Web servers. WebSEAL creates a unified namespace for all Web objects and creates a logical namespace of *all* Web servers. For instance, the following three Web sites seem to be located on the same server, but they could actually be located on three different, junctioned servers:

```
http://www.mycompany.com/engineering  
http://www.mycompany.com/marketing  
http://www.mycompany.com/sales
```

WebSEAL also provides a solution for authentication to the junctioned Web servers should they require a user to authenticate. Policy Director with WebSEAL can authenticate a user to a junctioned Web server even if the user

ID and password are different from the ones used to authenticate to WebSEAL. This functionality is called *GSO* (Global Sign-On). The authentication to the junctioned Web servers is transparent to the users once a user is authenticated to WebSEAL (if not, WebSEAL prompts the user for a user ID and password for this primary authentication). Through the Policy Director Management Console, the administrator defines a user to be a GSO user and, at that time, assigns a secondary user ID and password to the user's record for back-end authentication.

Figure 16 illustrates the logical relationship between several servers that are managed by Policy Director junctions.

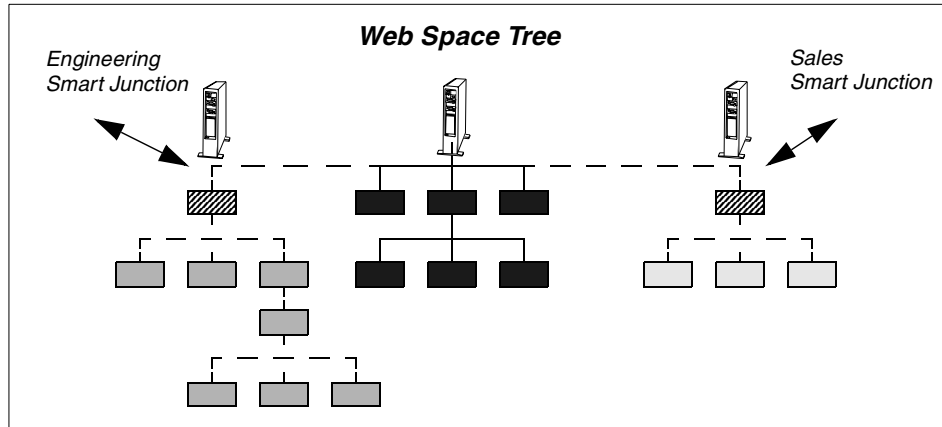


Figure 16. Logical smart junctions

Figure 17 illustrates how the Policy Director Management Console expresses junctions.

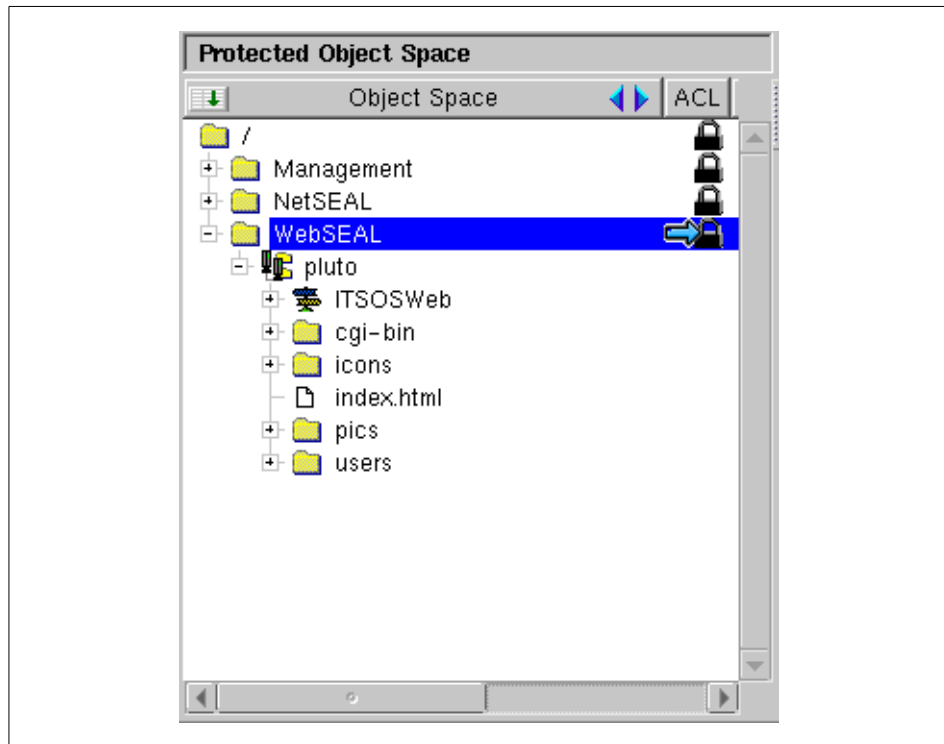


Figure 17. Smart junctions as viewed through the Management Console

In this example, the WebSEAL server is on a system called *pluto*. We have added ITSOSWeb, which is a file system holding many Web pages on a completely separate server, as a junctioned server into the object space represented by *pluto*. No user accessing ITSOSWeb would ever have any clue that the Web server they were accessing was not *pluto* itself.

Not only can back-end Web Servers be added to the Web object space in a way that is transparent to the end-user accessing the Web resources, but an organization can configure multiple Web servers, which are mirrored images of each other, into the same Web object space. WebSEAL will then load-balance between these multiple servers, and the users will be completely unaware of which system it is they are actually accessing.

We will return to configuring smart junctions later in this chapter.

4.3.2 NetSEAL

The Policy Director provides for the ability to include other TCP/IP servers in its protected domain, or cell. Such TCP/IP servers can, for example, be Telnet or FTP servers. This capability is packaged in NetSEAL, which operates as a port-request interceptor. Any attempt to access a NetSEAL protected server from the network invokes the Policy Director services for authentication and authorization on the *designated* Policy Director server (but only if the NetSEAL protected resource has been configured to *only* accept access requests from NetSEAL). The Policy Director server is designated by being configured in the NetSEAT client.

NetSEAL provides a built-in, coarse-grained access control manager. A user's ability to connect to a particular port on the server (for example, port 23 for Telnet) is controlled by NetSEAL.

When an authenticated connection is required, NetSEAL is always accessed using a NetSEAT client as a proxy. The two together provide a secure GSSAPI or SSL tunnel to support users accessing network applications. It is possible to access NetSEAL from a standard TCP client, but this will only provide an unauthenticated connection. We will first describe NetSEAT briefly and then outline the capabilities and uses of NetSEAT and NetSEAL working together.

Note that NetSEAL also supports its own kind of junctions. These are implemented using GSSAPI and provide secure communication between the NetSEAL server and the back-end server. For further details, see the *IBM SecureWay Policy Director Administration Guide*.

4.3.3 NetSEAT

NetSEAT is a set of Policy Director and DCE elements that are usually installed directly on a client system. It is lightweight and easy to install. It acts as a proxy between the client application and the Security Manager (WebSEAL or NetSEAL component). This capability allows for the inclusion of a client computer into a DCE cell, since NetSEAT functions as a DCE client, without installing DCE client code separately.

NetSEAT provides a secure tunnel end point for all network communications. The user's authenticated identity is passed over a GSSAPI or SSL created tunnel along with the original protocol request. Such a tunnel is used, for example, in NetSEAT's communications with NetSEAL.

An excellent example of a NetSEAT protected application is the Policy Director Management Console.

NetSEAT is a key enabler for NetSEAL. The next section looks a little more closely at the relationship between the two.

4.3.4 NetSEAL and NetSEAT working together

As mentioned, the communication between a NetSEAT client and a NetSEAL server takes place over a secure GSS (DCE) or SSL tunnel. NetSEAL protects the application server by utilizing Policy Director authorization capabilities.

An illustration of this working partnership is provided in Figure 18:

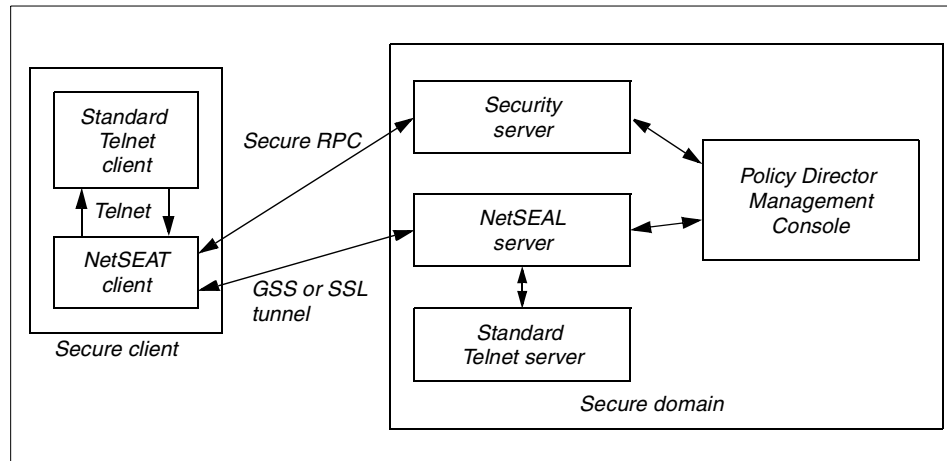


Figure 18. The combined NetSEAT and NetSEAL security model

As can be seen in Figure 18, the Telnet client (on the user's machine on the left) makes a request to access the Telnet server. This is intercepted by the NetSEAT client, which then authenticates to NetSEAL over the secure tunnel and checks the authorization permissions for the resource. If this process is successful, the Telnet client is allowed access to the server.

If GSS is used for secure communication, as in the scenario shown in Figure 18, then the NetSEAT client is part of the DCE cell that contains the secure components. If SSL is used instead, NetSEAT does not need to be part of the DCE cell. This may have an impact on the configuration of, for example, a firewall in the communication path, as explained in 10.1.2, "Technical implications" on page 277.

Figure 19 shows the simplest view of a NetSEAL server protecting the back-end network application servers. For simplicity, we have left out the

NetSEAT client, but be aware that the combination of NetSEAT and NetSEAL provides the fullest protection.

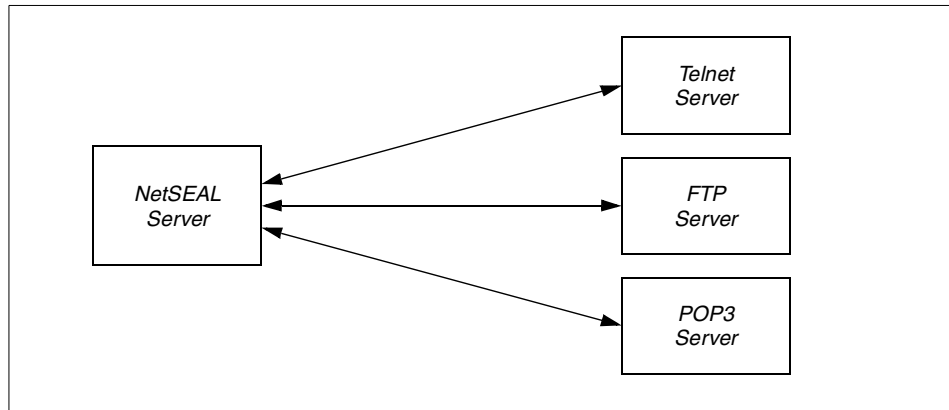


Figure 19. NetSEAL server protecting back-end application servers

However, what happens if the client is, for instance, in Japan, and the back-end servers are in the USA? This is a normal scenario in today's business world.

When the NetSEAT client is configured, there are several parameters that are configured through a GUI interface:

- Secure Domains
- NetSEAL Servers
- Host Security
- Subnet Security
- General

The *Secure Domain*, as its name suggests, is where you tell NetSEAT what secure Policy Director domains it will participate in and which Security Server to use (if DCE is being used).

NetSEAL Server refers to the NetSEAL tunnel destination. In other words, when a secure GSS or SSL tunnel connection from NetSEAT to NetSEAL is being configured, this specifies which NetSEAL server will be the other endpoint of that tunnel.

Host Security refers to the target NetSEAL server, or in other words, the NetSEAL server that is protecting the actual target application server you want to get to.

In the simple scenario shown in Figure 19, the NetSEAL tunnel destination and the target NetSEAL server are on the same machine. However, you can have many NetSEAL servers in a secure domain. By separating the tunnel destination and the target server, and putting these on two machines, it is possible to set up a secure communication channel across a Wide Area Network (WAN). Figure 20 illustrates this.

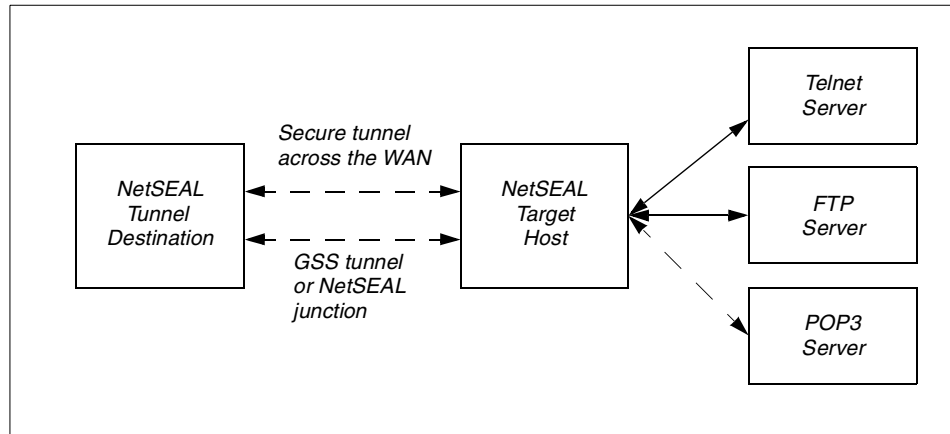


Figure 20. Using NetSEAL servers for secure communication across a WAN

NetSEAL servers communicate with each other using GSS tunnels or NetSEAL junctions. If we go back to our example, there can be a secure channel right from the NetSEAL client in Japan, across the WAN, to the NetSEAL server in the USA. And, the client is accessing a local NetSEAL server, while the back-end application servers are protected by a NetSEAL server local to them.

4.4 Users, groups, and security policies

This section describes users, groups, and security policies from an administration and management perspective. First, we offer an explanation of the concepts required to translate authorization policy into the system. This is followed by explanations of how an administrator works with ACLs, resources, and users.

Policy, in the context of authorization, essentially means the rules an organization applies with respect to which users are allowed to carry out which operations on which resources. In a large IT organization, this can appear a daunting task.

Security policy implementation is simplified with the aid of a graphical policy administration utility called the Management Console. This utility allows an administrator to apply authorization level security policy to defined network resources. It allows for very fine-grained, hierarchical definitions of the security policy. As we will see later, this allows for extremely granular authorization definitions without an administrator having to individually identify every relationship between a user and a resource.

Note that there is also an extensive command-line interface to carry out the tasks we will be describing later. Most activities can be carried out either through the command-line interface or through the Management Console. It is up to the administrator to decide which they are most comfortable with. See the *IBM SecureWay Policy Director Administration Guide* for a detailed description of the commands available.

Figure 21 shows the kinds of considerations a security administrator may have to take into account when setting up authorization policies.

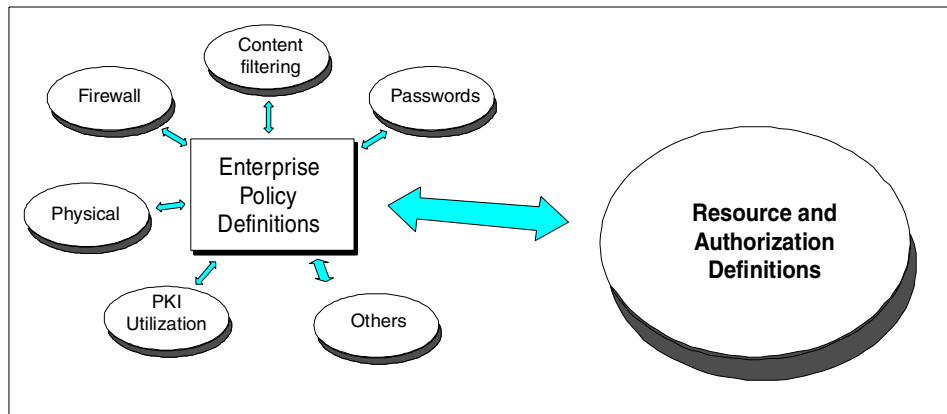


Figure 21. Aggregation of security policies - Focusing on authorization

From the definitions of the enterprise policy (see Figure 21), it is suggested that the following tables be designed and completed prior to beginning working at the Policy Director management console. Using a tool, such as a spreadsheet, to completely document the objects, permissions, and other information, will make the task of working with the management console much simpler. The examples included are very simplified, as specific needs will vary from enterprise to enterprise.

A user definition example worksheet is illustrated in Table 2.

Table 2. User definition worksheet

User ID	Comments and additional information for this user ID
alice	...
...	...

Table 3 provides an example of a group definition worksheet.

Table 3. Group definition worksheet

Group IDs	Member user IDs	Comments for this group
sales-team	alice bob	...
...

Table 4 provides an example of a resource definition worksheet.

Table 4. Resource definition worksheet

Resource Name	Comments and additional information
sales-db	DB2 database, on UNIX server xyz
...	...

Table 5 provides an example of an ACL definition worksheet.

Table 5. ACL definition worksheet

ACL name	User IDs/group IDs	Permissions
enable-sales	sales-team	R (ead) T (raverse) B (rowse) C (onnect)
...

Table 6 shows the mapping of ACLs to resources.

Table 6. Mapping ACLs to resources

ACL name	Remote Resource ID
enable-sales	sales-db
...	...

Although the latter two tables show the correct kind of layout to record any planned ACLs and the resources that can be applied to them, you may find it more useful to create a table like Table 7. Once you have understood the relationships of users, ACLs, and resources, you may find this table, Table 7, better reflects the way you think about security policies. You can then use the table to draw a plan of what ACLs are needed to cover all user/resource access combinations.

Table 7. User, ACL, and remote resource definitions

ACL name	User IDs Group IDs	Remote Resource ID	Permissions
enable-sales	sales-team	sales-db	R (ead) T (raverse) B (rowse) C (onnect)
...

If you are using the Global Sign-On (GSO), you should also set up a table such as the following (Table 8):

Table 8. GSO user definitions

Policy Director User ID / Password	WebSEAL proxy server name	GSO resource name	Application server User ID / Password
john/john	webproxy.it.com	appl.it.com	johnS/secured
...

If there are users who will be using GSO over smart junctions, the user IDs and passwords they use on the target systems will have to be entered into a special credential called a *resource credential*. This is entered through the Management Console as part of the `Users` view.

From here, the typical process of working with the Policy Director management console includes the following tasks:

- *Define who* (users and groups) will access these resources (employees, customers, partners).
- *Define what* are the objects requiring protection (Web, TCP ports, CORBA methods and interfaces).
- *Define how* these users will access the resources and what the rules protecting the objects will be (read, modify, administer, execute, and so on).

Policy Director essentially manages the mapping between users and groups to resources. The basic mechanism to represent access or authorization policy rules are the ACLs. An ACL is used to represent a relationship between a user, or a group of users, and a resource. This is represented in Figure 22.

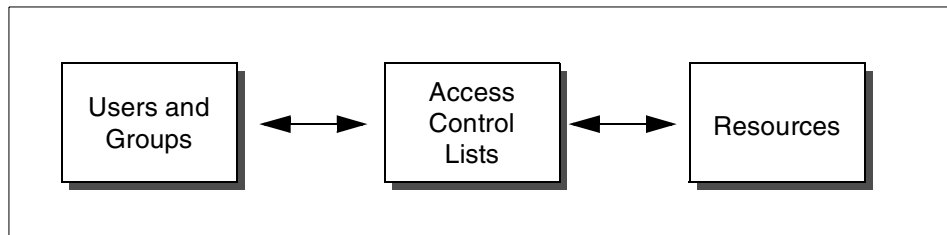


Figure 22. Relating users and resources via ACLs

The Management Console contains two significant views that are pertinent to our discussion. These are the *ACLs* view and the *Object Space* view. In the *ACLs* view, you can see what ACLs exist. You can look at an individual ACL and see what users/groups have been included in it and what permissions have been granted to each of these. In the *Object Space* view, you can get a hierarchical picture of the objects secured by Policy Director and see what ACLs have been applied to those objects. You can also open up the ACLs for inspection in this view. In the *Object Space* view, you can see exactly who is allowed to do what to which resources.

Note that the important point here is that an ACL is initially defined *without any mention of a resource*. It is only after an ACL has been defined that it is applied to a resource. And, any ACL can be applied to multiple resources. Because of this, and a feature known as *ACL inheritance*, it is possible to have very fine-grained access control to a large number of resources without the administrator having to individually set up and maintain every single relationship.

Most of the utilization of Policy Director will be to provide authorization control between Web browsers and Web pages, directories on an HTTP server, and/or programs that the HTTP server may launch or interface with. In terms of what a company is trying to achieve, this is what we call fine-grained access control. Coarse-grained access control with Policy Director refers to accessing a TCP service.

The sections that follow explain some of the administrator tasks in more detail.

4.5 Defining users and groups

One of the first tasks an administrator will carry out on the Management Console (remember that there is also a command-line interface for those who prefer this approach) will be to define users and groups. A screen capture of the user management interface is included here in Figure 23. There is a more detailed picture of the critical input areas further on. For now, simply look at the general console layout and notice the Action Tabs (Login, Users, Groups, and so on) at the top of the screen.

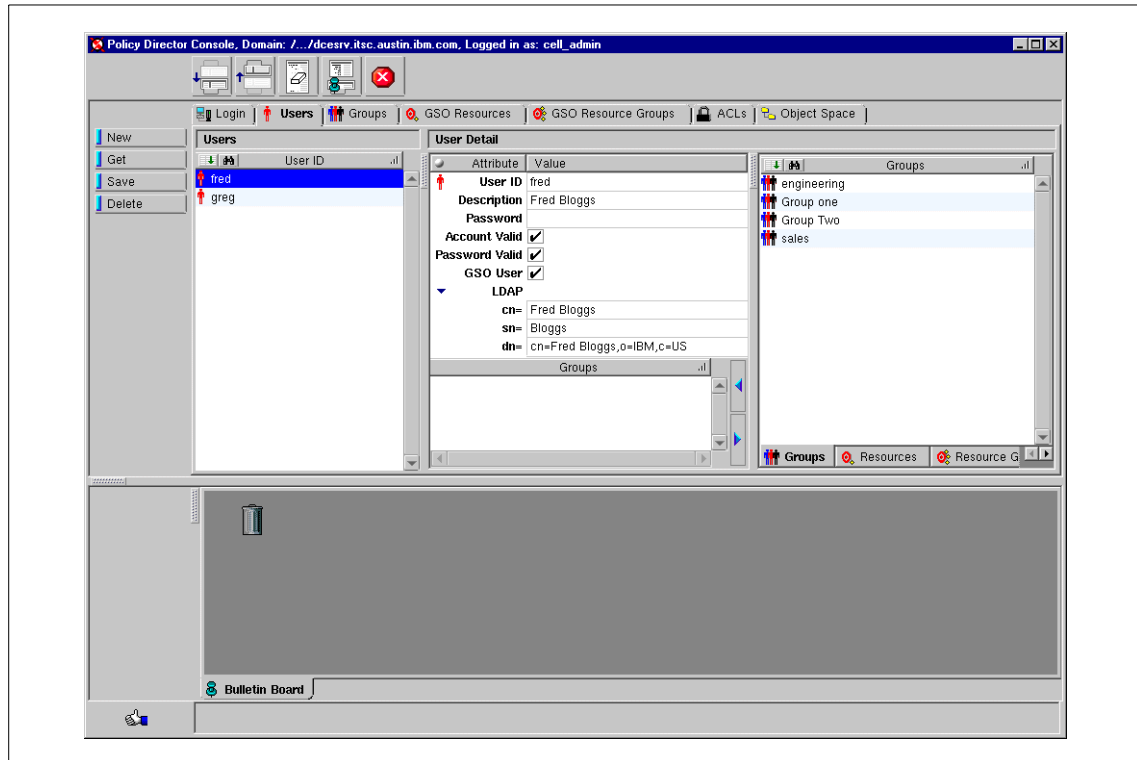


Figure 23. User management interface

In order to more clearly show the definition of a user, we have reproduced a partial screen capture of the critical area in Figure 24.

Figure 24 shows what we entered to create a hypothetical user from scratch. Note that we have defined this user as a GSO user by checking the relevant box.

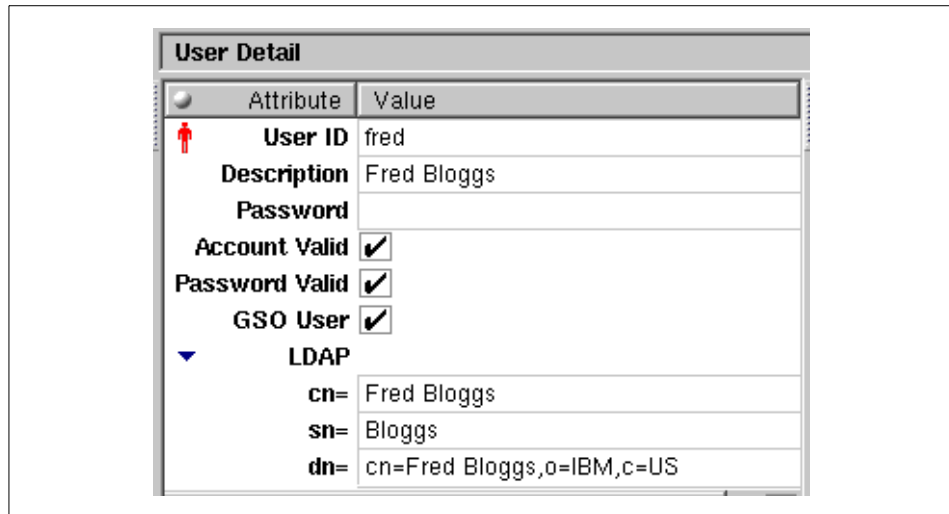


Figure 24. Defining a user

If configured as such, Policy Director exploits LDAP to store user and group information (see also Appendix B, “Introduction to LDAP” on page 321). User information maps into the ePerson schema of the LDAP directory. We do not need to understand this format in detail to carry out administration tasks. However, we need to understand how to enter the LDAP information for each user. The fields are as follows:

- cn** Refers to the common name for the user, for example Fred Bloggs.
- sn** Refers to the surname, Bloggs.
- dn** This is what is known in LDAP terms as a *Distinguished Name*, and this has to contain the common name and the information associated with *suffixes* that were set up when you configured LDAP prior to installing Policy Director (see *IBM SecureWay Policy Director: Up and Running*, for a detailed outline of adding suffixes).

In our example, we used the suffixes *o=IBM*, which stands for *organization equals IBM*, and *c=US*, which stands for *country equals US*.

As in most user definition schemes, users do not typically stand on their own in Policy Director but belong to a group. Defining groups is very simple. A picture of the Group Management interface is shown in Figure 25.

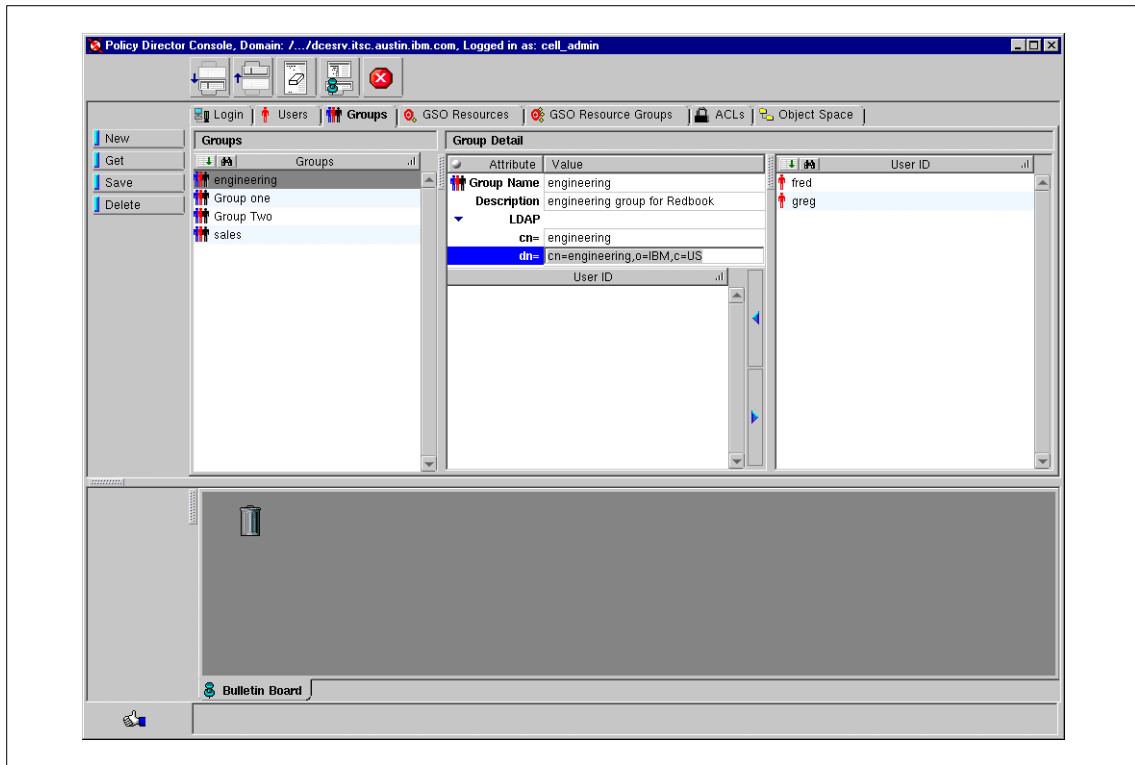


Figure 25. Group management

Note that the left most window pane contains a list of defined groups, while the right most window pane contains a list of the users available to populate the groups.

The next picture, Figure 26, shows a detail of a group definition.

Note that in the group definition, again there is a use of a Distinguished Name, and, again, this should contain values for the suffixes defined during LDAP customization.


Group Detail	
Attribute	Value
 Group Name	engineering
Description	engineering group for Redbook
LDAP	
cn=	engineering
dn=	cn=engineering,o=IBM,c=US

Figure 26. A group definition

Once a group has been defined and saved, users can be added to it from the Groups window by simply dragging and dropping users from the right most pane. This drag-and-drop method is used in many places in the Management Console and makes for easy authorization setup.

Managing users and groups is discussed in detail in the *IBM SecureWay Policy Director Administration Guide*.

4.6 Defining ACLs and attributes

This section describes ACLs and their attributes. Recall from our previous discussion (4.4, “Users, groups, and security policies” on page 74) that the ACLs are initially defined without reference to a protected resource. We will see later how an administrator applies a specific ACL to a resource.

When ACLs are applied to resources, they are applied in a hierarchical manner that mirrors the hierarchical relationship of the protected resources to each other. Using the management interface, ACLs may be copied and applied to other resources in the hierarchy and, subsequently, altered to suit the context. For this reason, ACLs are often referred to as policy templates. These templates will be discussed in greater detail in the following discussion.

Note

This redbook describes some of the predefined permissions and ACLs. In addition, Policy Director provides you with the flexibility to create your own permissions and ACLs. Please refer to the *IBM SecureWay Policy Director Administration Guide* for further information on this capability.

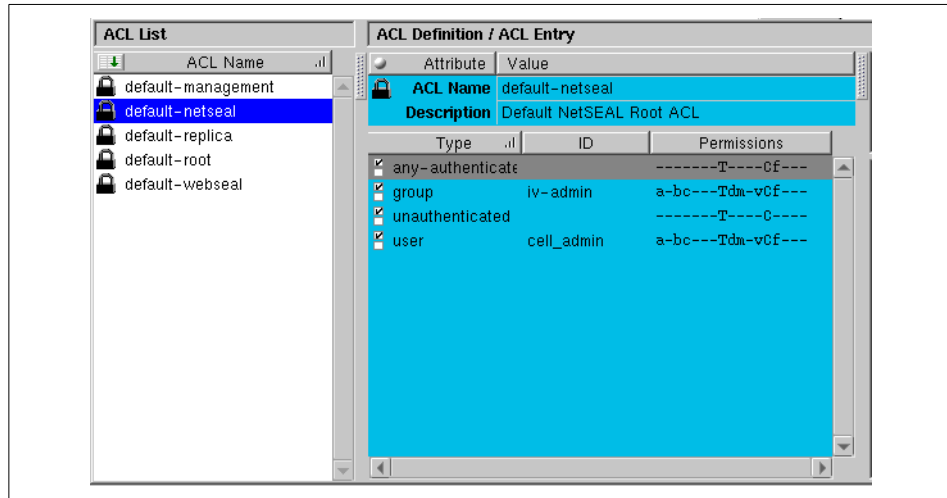


Figure 27. Detail of ACLs view in the Management Console

Figure 27 shows a detail of the ACLs view as seen in the Management Console. Note the five ACLs listed in the left most pane. These are the default ACLs supplied with Policy Director. These default ACLs can be copied and used as the basis of organization-specific ACLs. The `default-netseal` ACL is highlighted, which causes the details of this ACL to appear in the right-hand pane (for clarity, we have not produced the actual right most pane of the ACLs view here, though it appears in a later figure). The `default-netseal` ACL is quite basic and makes for a good example for discussion. We will refer back to this figure later.

4.6.1 Attributes

An ACL entry contains either two or three attributes: *Type*, *ID*, and *Permissions*. Whether there are two or three depends on the ACL entry type. You can see this in the preceding illustration (Figure 27).

Type refers to the entity category (user or group) for which the *Permissions* are being specified. The different types are: *User*, *group*, *any-authenticated*, and *unauthenticated*. *ID* (Identity) is the unique identifier (name) of the specific entity. So, for example, `cell_admin` is a specific instance of the entity category `user`. The ID attribute is not required for the any-authenticated and unauthenticated ACL entry types. *Permissions* refers to the set of operations permitted on an object by this user or group. Note that there is nothing on this view that tells you what object (protected resource) this ACL is being applied to.

The following illustration, Figure 28, outlines the mapping of ACLs and resources. The shaded boxes each represent a directory that has an ACL applied to it.

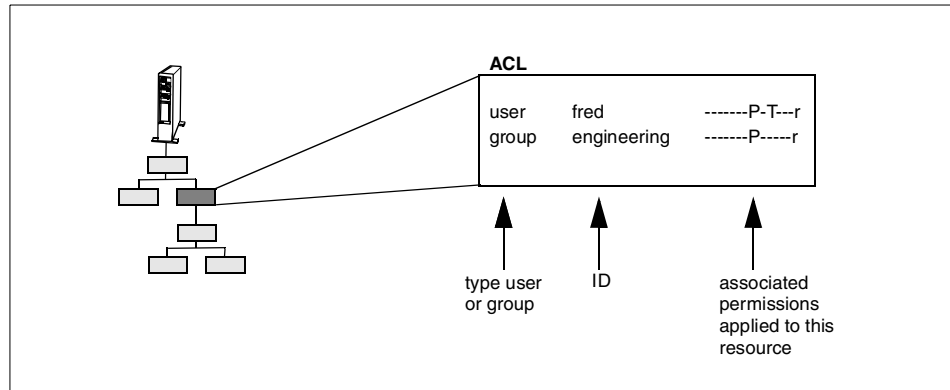


Figure 28. ACLs mapping user/group permissions to a resource

In the example (Figure 28), the user *fred* (type = user, ID = fred) has permission to read (view) the object that has been associated with the ACL containing this entry. The *r* permission allows the read operation. The *P* permission mandates that the communication channel use data encryption. The *T* permission enforces the traverse attribute (which is explained later, see Table 12 on page 101). *User* permissions over-ride *group* permissions, so that if the user *fred* belonged to the *engineering* group, he *would* be able to traverse (*T*) the object.

Most permissions dictate the client's ability to perform a specific operation on a resource. Several permissions impose certain conditions whenever an action on a resource is allowed. Examples of the latter can include forcing data encryption, requiring data integrity protection, writing a report record to the auditing service, or some external authorization condition.

The ACL *ID* is the unique identifier, or name, used to refer to a user or group entry type. IDs must represent valid users and/or groups created for the secure domain and stored in the Policy Director User Registry. The ID attribute is not used for the *any-authenticated* and *unauthenticated* ACL entry types because, by definition, these types would never contain a unique

person or group. A few examples of ACL types that have specific IDs defined are shown in Table 9:

Table 9. Some ACL examples

ACL Type	ACL ID
group	iv-admin
user	cell_admin
user	fred
group	engineering

Note that, in Table 9, we have used two IDs from our screen capture (Figure 27 on page 83) and have shown in the table that, if we want to, we could add a user and a group from our previous examples (Figure 24 and Figure 26).

The four ACL entry types are summarized in the following (Table 10):

Table 10. ACL types

Type	Description
user	<p>Sets permissions for a specific principal in the secure domain. The user entry type requires an account name (ID).</p> <p>The entry format is: user ID permissions For example: user Richard Roe-----r-</p>
group	<p>Sets permissions for members of a specific group in the secure domain. The group entry type requires a group name (ID).</p> <p>The entry format is: group ID permissions For example: group engineering -----r-</p>
any-authenticated	<p>Sets permissions for all authenticated users. No ID designation is required.</p> <p>The entry format is: any-authenticated permissions For example: any-authenticated -----r-</p>

Type	Description
unauthenticated	<p>Sets permissions for those users who have not been authenticated by the Security Server. No ID designation is required.</p> <p>The entry format is: unauthenticated permissions For example: unauthenticated -----T---r-</p> <p>Note: This ACL entry is masked (a bit-wise “and” operation) against the any-authenticated ACL entry to determine the permission set.</p> <p>For example, the following unauthenticated ACL entry: unauthenticated -----r- masked against this any-authenticated ACL entry: any-authenticated -----T---r- results in these permissions: -----r- (read only).</p>

4.6.2 Permission attributes

Each ACL entry contains a set of permissions describing the specific operations permitted on an object by the principal or group. This includes any restrictions on the type of access to the object, such as mandatory use of data privacy or integrity on the communications channel, audited access, and external (third-party) authorization requirements.

ACLs control protected resources by:

- Limiting a user’s ability to perform operations on protected objects
- Defining an administrator’s ability to change access control rules on the object and any sub-objects
- Specifying the Policy Director server’s ability to delegate user’s credentials

ACL permissions are context-sensitive. This means the behavior of certain permissions vary according to the region of the protected object space in which they are applied. For instance, the *m* permission has a different meaning on a WebSEAL object than on a Management object. On a WebSEAL object, this would simply allow the user to add an HTTP object to the Web object space. Depending on which kind of Management object it is applied to, this could allow a user to create a new Policy Director server definition or even create a new ACL.

In the previous figure, Figure 27, we were looking at the ACL for `default-netseal`. Notice that in the right-hand pane in that figure, `any-authenticated` is highlighted. That picture, however, does not show the full ACLs view on the Management Console. There is a further pane to the right of the panes shown, which was left out of Figure 27 for clarity. We have reproduced that additional pane here. In this pane, shown in Figure 29, the permissions for the `any-authenticated` type in the `default-netseal` ACL are detailed.

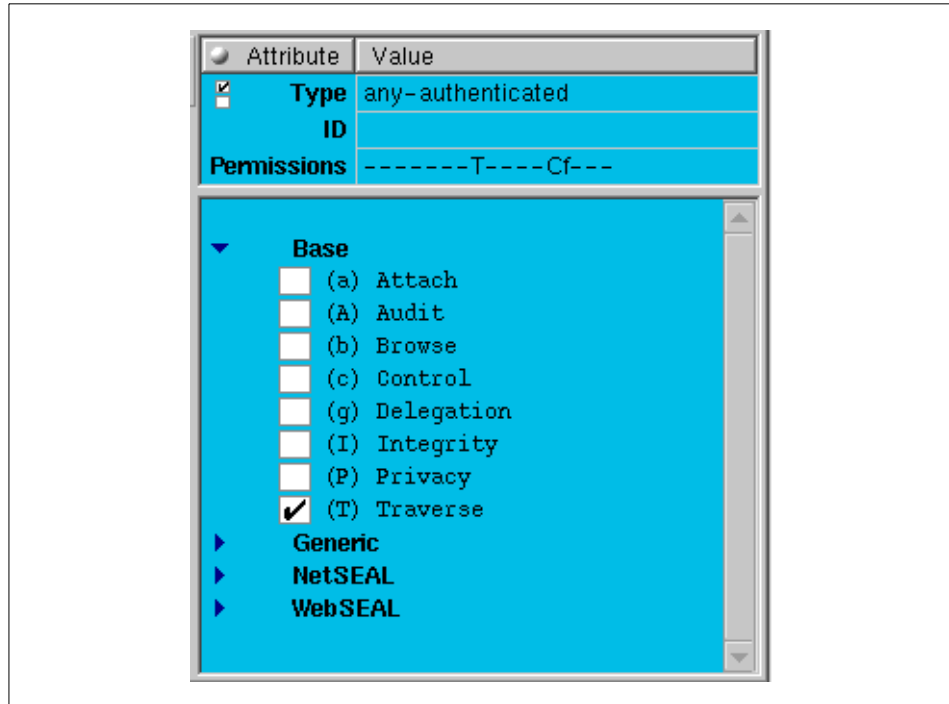


Figure 29. Details of permission settings

Note that this pane expands the rather terse line that summarized the permissions in Figure 28 on page 84. A detailed list of the permissions and what they mean is available in the *IBM SecureWay Policy Director Administration Guide*.

4.6.3 Standard permissions summary

The seventeen standard permissions are grouped into four categories and appear in the following order, as viewed in the management console, when used in an ACL entry. The permission symbols are listed below and are fully explained in 4.7, “Policy Director namespaces” on page 96.

Base: a A b c g I P T
Generic: d m s v
WebSEAL: l r x
NetSEAL: C p

Note that it is also possible to set up custom permissions to satisfy the individual needs of any particular organization. Refer to section 4.7.1.4, “The user-defined namespace” on page 100 for more details.

4.6.4 Utilizing ACLs as templates

This section describes ACLs and their application to objects in the protected namespace.

Policy Director protects network resources with access control lists (ACLs). An ACL is the set of controls (permissions) that specifies the conditions necessary to access a resource and perform certain operations on that resource. An ACL can also be thought of as a privilege or entitlement. ACLs:

- Hold the policy rules for the organization
- Are associated with objects in the object namespace
- Are explicitly applied or inherited

ACLs can be attached to more than one object. Some examples include: Hard-coded rules, external authorization capability, special secure labeling, and Access Control Lists (ACLs).

The following discussion shows how an ACL is associated with an object, using some screen captures for illustration. First, please refer back to Figure 28 on page 84 for a look at how ACLs appear in the ACLs view on the Management Console.

We now need to look at the *Object Space* view on the Management Console. Figure 30 shows the total view available to the administrator.

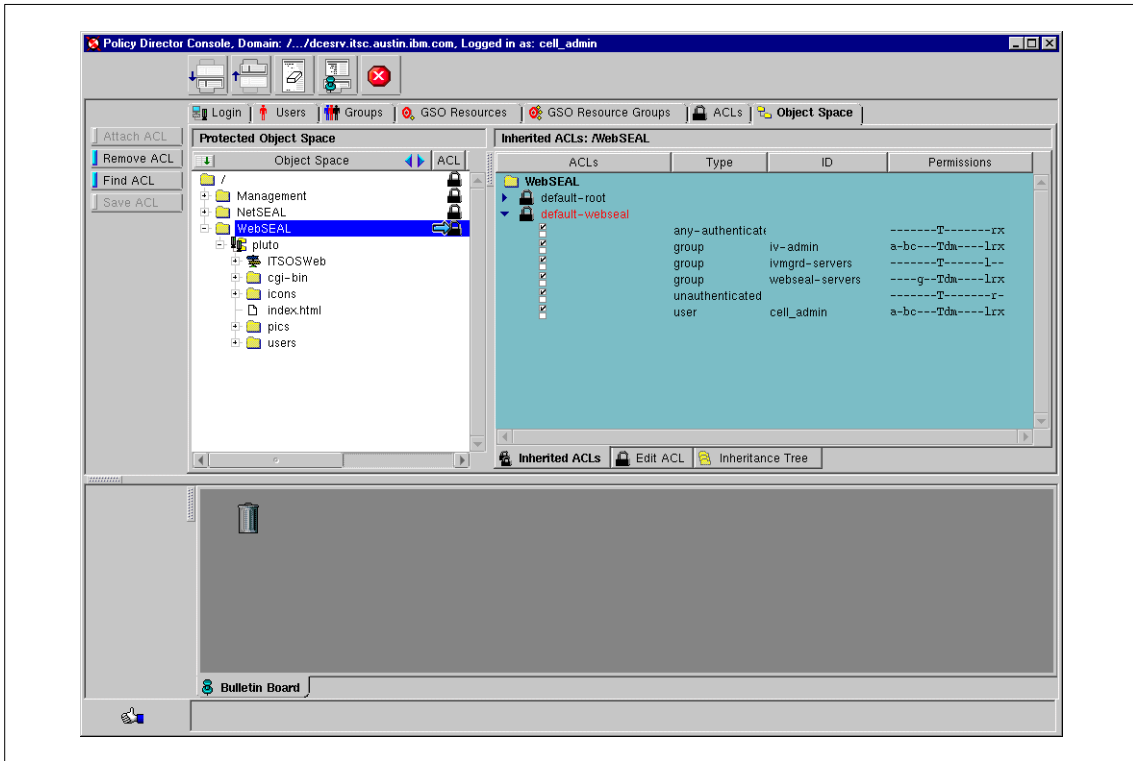


Figure 30. The Object Space view on the Management Console

Note that, on this screen, an administrator can view the contents of the object namespace in the left hand pane and that there is a display of something that looks very much like an ACL in the right hand pane. Note the padlocks icons that show up in both panes. These are the graphical representations of the ACLs. Let us look in a little more detail at what is in each pane.

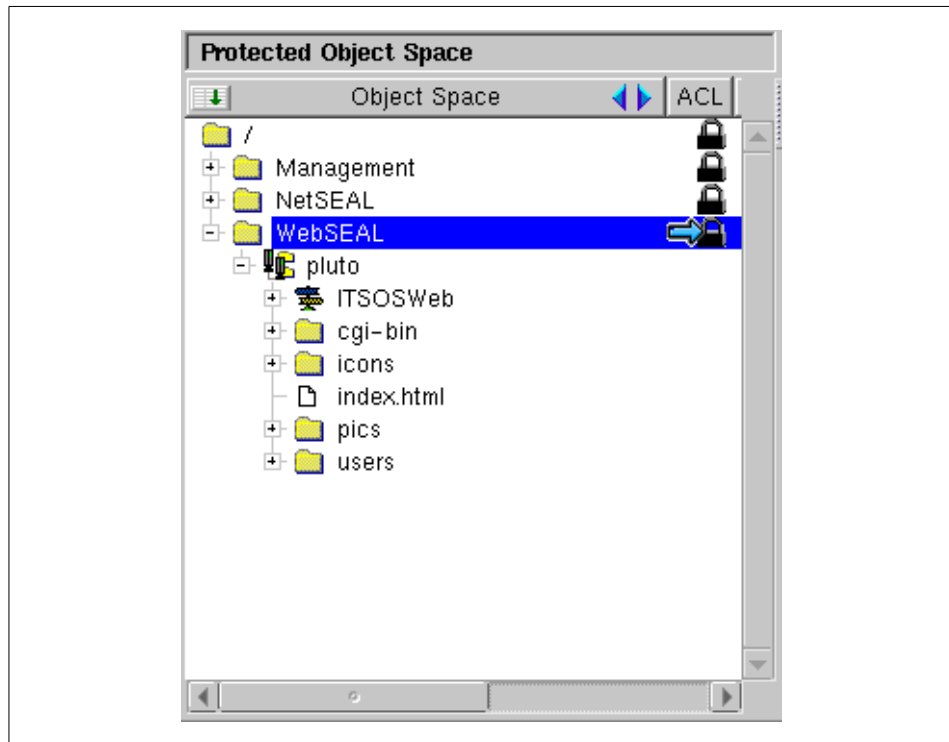


Figure 31. Detail of the objects displayed in the Object Space view

The figure above, Figure 31, shows us the resources present in our object namespace. Note that some of these objects have the padlock icons next to them. The presence of these icons indicate that an ACL has been *explicitly* associated with that resource.

Note also that the objects are represented in a hierarchy. This would represent some kind of real-life relationship between real objects. For example, the `cgi-bin` directory is found on the `pluto` host. This will be important when we are looking at ACL inheritance.

The next picture, Figure 34, is a detailed view of the right hand pane of the Object Space.

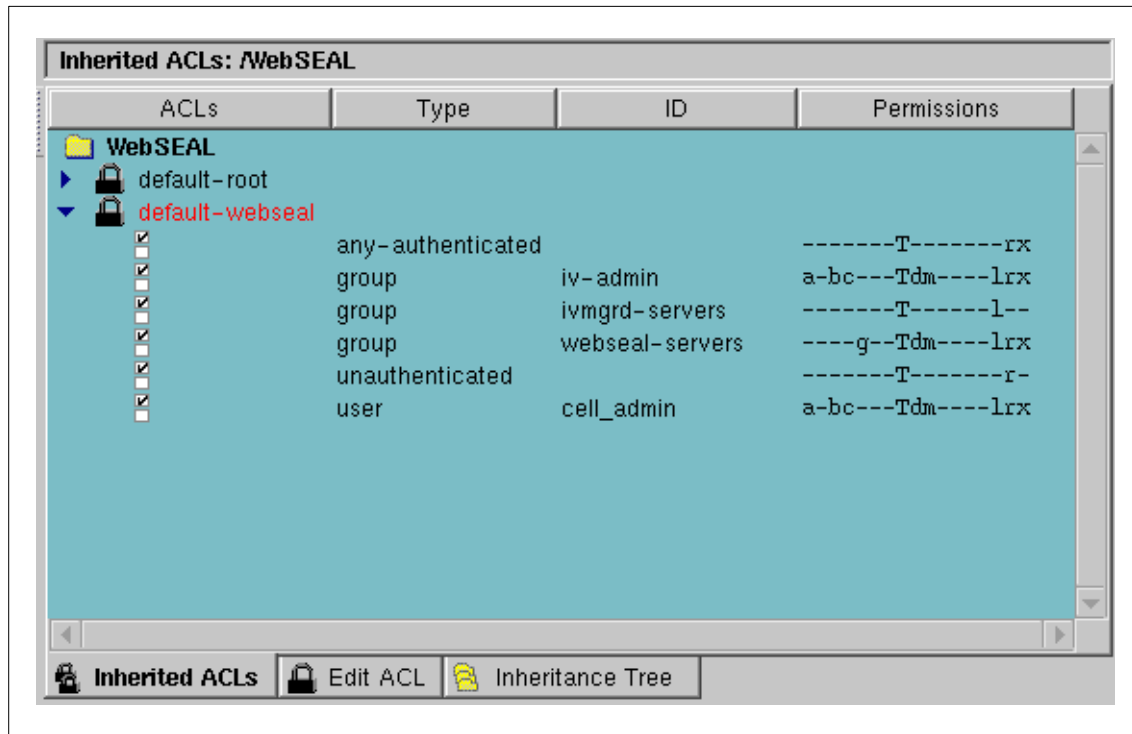


Figure 32. Detail of the ACLs displayed in the Object Space view

In Figure 31, the highlighted object was WebSEAL, representing the WebSEAL server function. Because of this, what is showing in the right hand pane (Figure 32) is the list of ACLs that have been applied to the WebSEAL object. There are actually two ACLs applied: `default-root` and `default-webseal`. Furthermore, `default-webseal` is opened up to let us see its contents. We can see that `default-webseal` has permissions set up for three groups: One real user and the two other types, `any-authenticated` and `unauthenticated`. We can also see that the permissions for each type to access the WebSEAL object are not all the same.

Why are there two ACLs here? This is because of the concept of ACL inheritance, which we discuss in detail in a following section. In our example above, the `default-root` ACL has been *inherited* by the WebSEAL object.

So how do the ACLs get set against an object? The process is easier to show than to describe. The first thing you have to do is produce a Management Console screen that has both the Object Space and the ACLs view open on it at the same time. How is this achieved? You may have noticed in some of the

full screen captures that there is an area below the main panes that typically contains a garbage can icon (see Figure 30). You may also have noticed an icon at the top of the screen similar to the following (Figure 33):

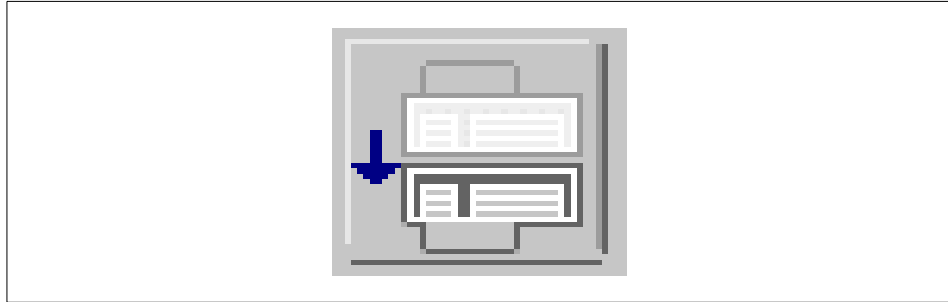


Figure 33. Push-down icon on Management Console

The Management Console screen is really in two halves: An upper half, which is where all the views we have been looking at are placed by default immediately after installation, and a lower half, which typically contains the garbage can and a bulletin board, but which is also available for use. When you click the icon shown in Figure 33, whatever view is active in the upper half gets moved into the lower half. In this way, you can see two views at once. Clicking it while on the ACLs view, for example, results in the following screen:

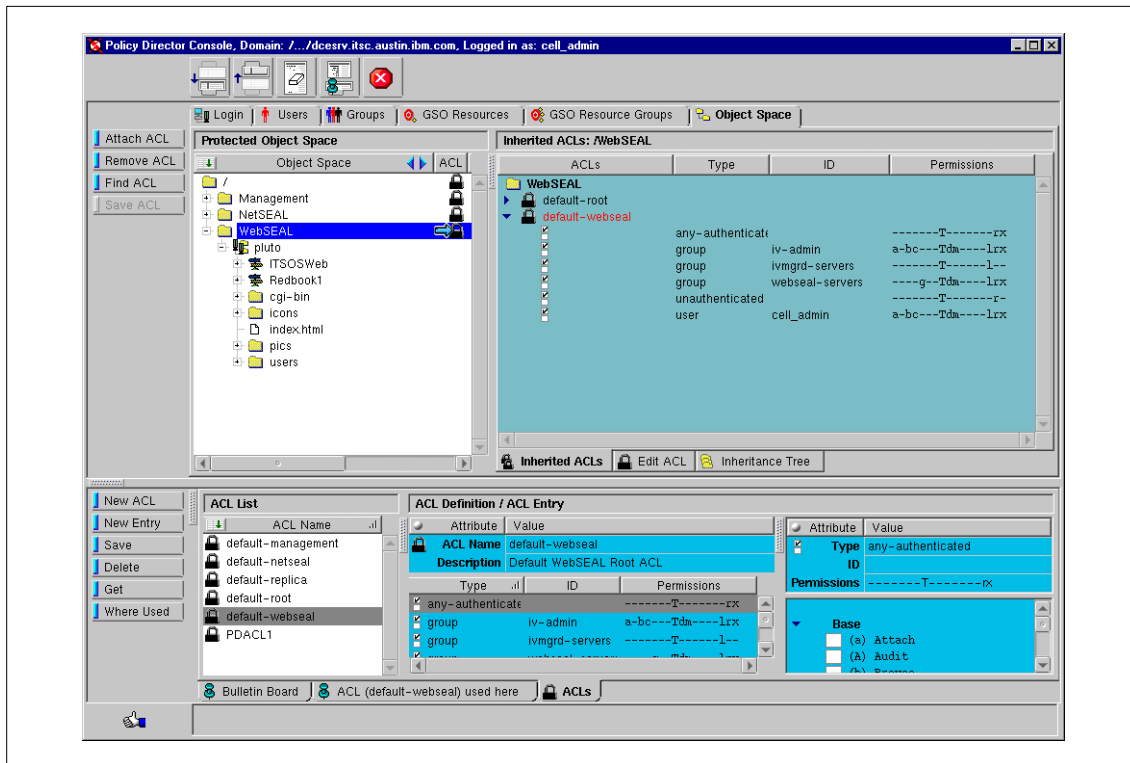


Figure 34. Result of pushing down the ACLs view

You can now see the Object Space view and the ACLs view on one screen.

Now all that needs to be done is to click on the padlock next to the appropriate ACL in the ACLs view (lower half) and drag-and-drop it over the respective resource in the Object Space (upper half) to apply that ACL to.

Now that the ACL has been applied to the resource, the ACL specifically controls:

- Who can access the resource.
- What operations can be performed on the resource.

When a user makes a request for a protected resource, the Policy Director Authorization Service compares the resource's ACLs with the user's credentials. The Authorization Service then permits or denies the user's request to access and/or perform specific operations on that resource.

We will now look at ACL inheritance.

4.6.5 Sparse ACL model: ACL inheritance

Through mutual client/server authentication and an authorization decision-making process, specific resources can be made available for general availability while restricting sensitive information to more secure access.

As we have seen, the Policy Director uses access control lists, or ACLs, to protect system and network resources. An ACL specifies the conditions necessary to access a protected network resource (who can access the resource and what operations can be performed on the resource). ACLs are either applied explicitly to an object in the namespace, or the ACL definitions are inherited from above in the hierarchy. This is illustrated in Figure 35.

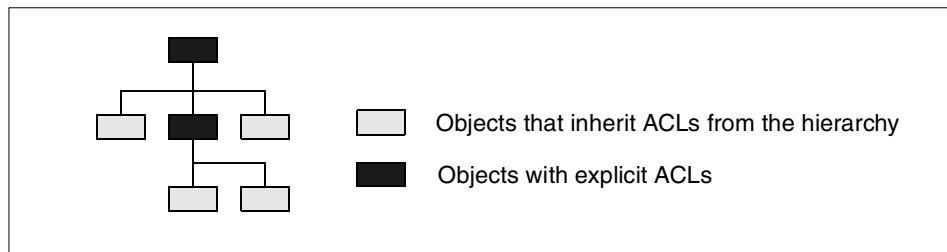


Figure 35. ACL inheritance

To secure resources in a protected object space, an Access Control List (ACL) must be attached to each object using either method.

Adopting an inherited ACL scheme can greatly reduce the administration tasks for a secure domain. This section discusses the concepts of inherited, or *sparse* ACLs.

Understanding the Sparse ACL Model

The power of ACL inheritance is based on the following principle: Any object without an explicitly attached ACL inherits the ACL of its nearest *container object* with an explicitly set ACL. In other words, all objects without explicitly attached ACLs inherit ACLs from container objects above them in the hierarchy with explicitly attached ACLs.

ACL inheritance simplifies the task of setting and maintaining access controls on a large protected object space. In a typical object space, you only need to attach a few ACLs at key locations to secure the entire object space, hence, a sparse ACL model. The combination of being able to grant many different kinds of access to an object to different groups and users, coupled with the

use of the sparse ACL model, provides very fine-grained access control with the least possible administrative effort.

A typical Policy Director namespace begins with a single, explicit ACL attached to the root container object. The root ACL must always exist and can never be removed. Normally, this is an ACL with very little restriction. All objects located in the namespace below inherit this ACL.

When a region or sub-tree in the namespace requires different access control restrictions, an explicit ACL must be attached at the root of that sub-tree. This interrupts the flow of inherited ACLs from the primary namespace root to that sub-tree. A new chain of inheritance begins from this newly created explicit ACL.

4.6.6 Evaluating an ACL

Policy Director follows a specific evaluation process to determine the permissions granted to a particular principal by an ACL.

4.6.6.1 Evaluating authenticated requests

Policy Director evaluates an authenticated user in the following order:

1. Matches the user ID with the ACLs user entries. The permissions granted are those in the matching entry. If successful, evaluation stops here; otherwise, evaluation continues to the next step.
2. Determines the group(s) to which the user belongs and matches it with the ACLs group entries: If more than one group entry is matched, the resulting permissions are a logical *or* (most permissive) of the permissions granted by each matching entry. If successful, evaluation stops here; otherwise, evaluation continues to the next step.
3. Grants the permissions of the any-authenticated entry (if it exists). If successful, evaluation stops here; otherwise, evaluation continues to the next step.
4. An implicit any-authenticated entity exists when there is no any-authenticated ACL entry. This implicit entry grants no permissions.

4.6.6.2 Evaluating unauthenticated requests

Policy Director evaluates an unauthenticated principal by granting the permissions from the ACLs unauthenticated entry.

The unauthenticated entry is masked (a bit-wise *and* operation) against the any-authenticated entry when permissions are determined. A permission for

unauthenticated is granted only if the permission also appears in the any-authenticated entry.

Since unauthenticated depends on any-authenticated, it makes little sense for an ACL to contain unauthenticated without any-authenticated. If an ACL does contain unauthenticated without any-authenticated, the default response is to grant no permissions to unauthenticated.

4.7 Policy Director namespaces

Namespaces refer to all the names and attributes of all the objects in the Policy Director environment. They can be thought of as a hierarchy of protected resources. System resources or actual Web pages, TCP ports, and CGI scripts are also included in the namespace. An item in the namespace is simply a protected object or logical representation of the resource.

The namespace is maintained by the Authorization Policy Database, an underlying element of Policy Director.

Figure 36 illustrates the complete suite of objects in the namespace.

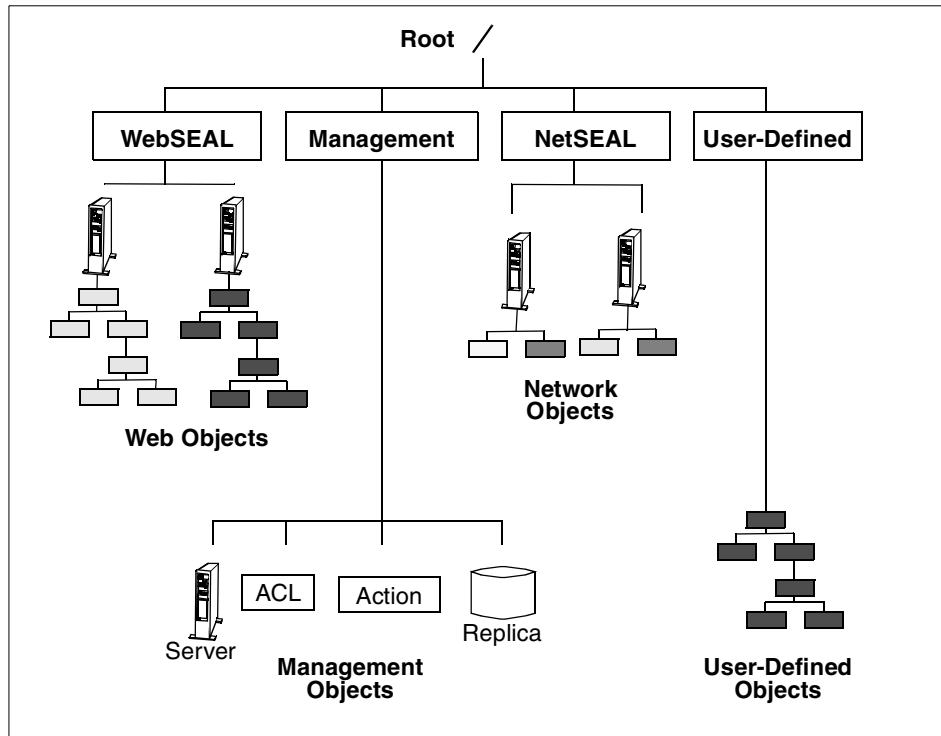


Figure 36. Policy Director namespace hierarchy

4.7.1 Standard namespace regions

As we can see in Figure 36, the namespace consists of a root container object, within which are several regions. Three of these regions come as standard: WebSEAL, NetSEAL, and Management. It is also possible to extensively extend the namespace region through the use of *User-Defined* objects, which have their own region. We will first look briefly at the WebSEAL and NetSEAL regions. Later, in 4.8, “Managing administrators within the namespace” on page 110, we will look at the Management region in the context of carrying out management administration. We will also discuss securing regions within the total namespace and look at the User-Defined namespace in some more detail.

4.7.1.1 Root (/) container object

The root object begins the chain of ACL inheritance for the entire protected object namespace. If no other explicit ACLs are applied, the root object defines (through inheritance) the security policy for the entire namespace. Traverse permission is required for access to any object below root.

4.7.1.2 The WebSEAL namespace

The following security considerations apply for the /WebSEAL container object:

- The WebSEAL object begins the chain of ACL inheritance for the WebSEAL region of the namespace.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire Web space.
- The traverse permission is required for access to this object and any object below this point.

/WebSEAL/<host>

This sub-tree contains the Web space of a particular Policy Director WebSEAL server. The following security considerations apply for this object:

- The traverse permission is required for access to any object below this point.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire namespace protected by the WebSEAL server (including resources on smart junctions, which we will explore later).

/WebSEAL/<host>/[<container>]/<file>

This is the resource object checked for HTTP access. The permissions checked depend on the operation being requested.

Section 4.7.3, “Extending the Web space - Smart junctions” on page 101, elaborates in more detail on smart junctions, a means of extending the Web space.

4.7.1.3 The NetSEAL namespace

The following security considerations apply for the /NetSEAL object:

- The NetSEAL object begins the chain of ACL inheritance for the NetSEAL region of the namespace.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for all NetSEAL protected services in the namespace.
- The traverse permission is required for access to this object and any object below this point.

/NetSEAL/<host>

This subtree contains all NetSEAL protected services on the particular server machine. The following security considerations apply for this object:

- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for all NetSEAL protected services on this machine.
- The traverse permission is required for access to any resource below this point.

/NetSEAL/<host>/<service>

This is the object checked for access to the protected service it represents. The permissions checked depend on the operation being requested.

NetSEAL specific permissions

Two permissions are relevant to NetSEAL and are described in Table 11.

Table 11. NetSEAL ACL permissions

Symbol	Access	Description
C	connect	Connect to a NetSEAL server.
p	proxy	Connect to the protected service that resides on a machine remote from this NetSEAL server.

The *proxy* permission applies to NetSEAL SSL junctions. This permission has to be set to allow a connection to a port on a remote machine.

These permission relationships are illustrated in Figure 37.

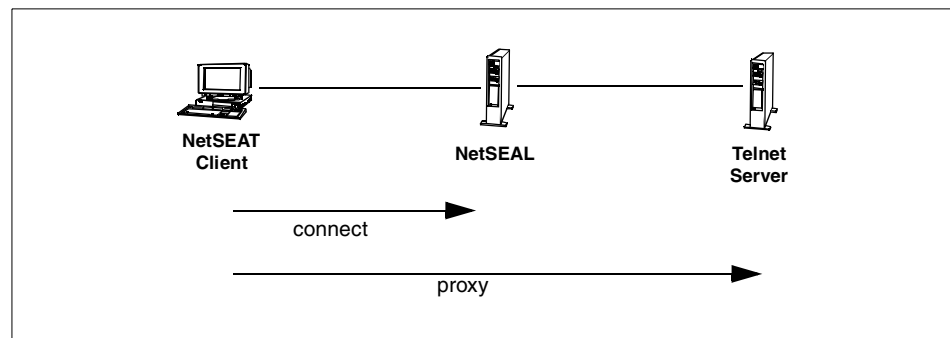


Figure 37. NetSEAL permissions

4.7.1.4 The user-defined namespace

Policy Director allows the extension of its authorization services to objects that belong in a third-party, or user-defined, namespace. The contents of the third-party namespace are defined through a special mapping file. The mapping file contains a list of all the resources that are to be defined, and for each object, the full path name must be given. An example of the contents of a mapping file is:

```
/Sales/figures/monthly  
/Sales/figures/quarterly  
/Sales/figures/daily  
/Inventory/shoes/Alabama  
/Inventory/shoes/Georgia  
...
```

You then specify in the configuration file, `ivmgrd.conf`, where this mapping file is to be found.

Once the objects are created in the namespace, all the standard Policy Director permissions can be applied. Custom permissions can also be set up to be applied to the objects in the third party namespace. Once set up, these appear and can be used in the ACLs view of the Management Console in exactly the same way as the standard permissions.

See the *IBM SecureWay Policy Director Administration Guide* for a detailed discussion of setting up third-party namespaces and custom permissions.

4.7.2 Securing regions of the Policy Director namespace

Figure 36 on page 97 illustrates the complete range of namespace objects in Policy Director. *Container objects*, or file system directories, represent specific regions of the protected object namespace and serve two important security functions:

- The container object's ACL can be used to define high level policy for all sub-objects within the region when no other explicit ACLs are applied.
- Access denial can be quickly applied to all objects in a region by removing the traverse permission from the container object's ACL.

Access conditions are generic permissions and apply throughout the namespace. The traverse is a generic permission, but its use really applies to container objects although it has to be present on both the container object

and the actual target object in order to allow access to the object. Table 12 describes the traverse permission.

Table 12. The traverse ACL permission

Symbol	Access	Description
T	traverse	Allows the requester to hierarchically pass through the object on the way to the requested object. It does not allow any other type of access to the object. Traverse is also required on the requested object itself.

Table 13 describes access condition permissions.

Table 13. Access condition ACL permissions

Symbol	Access	Description
A	audit	Causes the Policy Director server to write an audit record to the audit service whenever the object is accessed. This audits all access attempts including authorization failures.
I	integrity	To access this object, data integrity protection is required between the client and the Policy Director server.
P	privacy	To access this object, data encryption is required between the client and the Policy Director server.

The control permission is a powerful permission that grants ownership of the ACL. Control allows you to modify the entries in the ACL. This means you have the power to create entries, delete entries, grant permissions, and take away permissions.

Table 14 describes the control permission.

Table 14. The control ACL permission

Symbol	Access	Description
c	control	Ownership of the ACL template allowed to create, delete, and modify entries for this ACL.

4.7.3 Extending the Web space - Smart junctions

The WebSEAL server protects its own Web resources (URLs) and those of other Web servers placed logically underneath it in the object namespace hierarchy. As well as providing authentication and authorization services for those remote back-end Web servers, it protects their identity by masking their URLs (see the example in 4.3.1, “WebSEAL” on page 67), and it can provide

load balancing between identical back-end Web servers by directing user requests to the least-busy Web server.

This is all achieved through the use of *smart junctions*. Most of the configuration work for smart junctions can be carried out at the Management Console; some tasks have to be done on a command-line interface. An easy way to describe smart junctions is to show an example. The example that follows applies to a UNIX system, but the Windows commands would be essentially the same.

For our example, we will create an SSL smart junction that will utilize Global Sign-On (GSO) capabilities. This allows us to demonstrate both the use of the smart junction and the use of GSO in one example.

Step 1. Gather information

The first thing to do is to check what we have and what we need. The following needs to be known:

- What already exists in the object space? Especially, what WebSEAL servers?
- What is the fully qualified name of the remote system that is to be the back-end Web server?
- What name is to be assigned to this resource within Policy Director?
- What are the user IDs and passwords of the users on the remote Web server? Do all these users have a user record in Policy Director?

To see what is already in the object namespace, you can simply look in the Policy Director Management Console (see Figure 38).

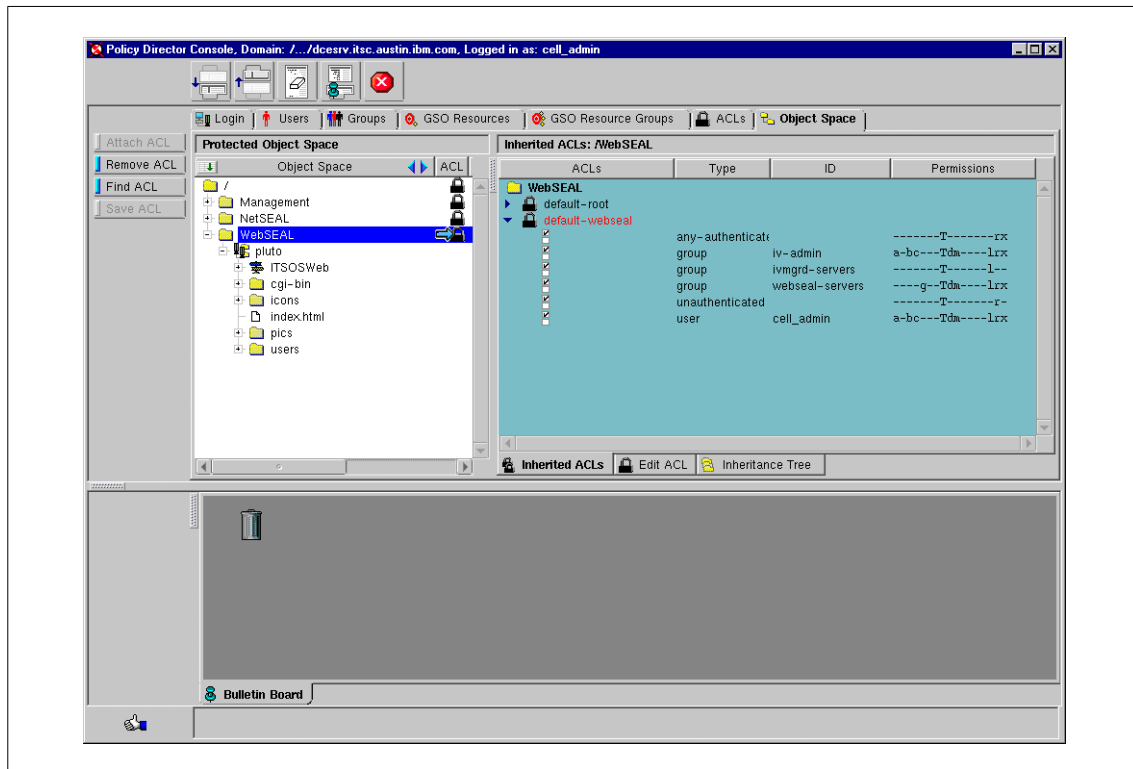


Figure 38. The Object Space view in the Management Console

Note that, in our scenario, we have one WebSEAL server called *pluto*. All our back-end Web servers will attach to *pluto*. The back-end Web server we will be attaching in our example is: www.mycompany.com

We have to decide what to call the resource in Policy Director. This name is only used in Policy Director, not in any other context. However, it may be helpful to choose a meaningful name. In our example, we decided to call the resource *Redbook*.

Finally, we have to look at the users of the Web service provided by the back-end Web server. These users are now going to be accessing this resource via WebSEAL. If the Web resource did not require any authentication, there would be no need to concern ourselves with this step. However, our scenario assumes that we are dealing with a user population that is accessing multiple resources through Policy Director. Each user may have several user IDs and passwords to access the variety of systems they need to carry out their daily jobs.

This is where Policy Director's GSO comes into its own. By the end of this example, you will have seen how a user can be given a Policy Director user ID and password, which Policy Director can then map to other user IDs and passwords for remote systems. Once the user has authenticated with Policy Director and requested the desired resource, Policy Director will then provide the back-end resource manager with the appropriate user ID and password for that resource, totally transparently to the user. The user now only has to remember their Policy Director user ID and password.

The last task in this step is to gather the user IDs and passwords of the users of the Web resource in question, and then check that each of these users has a user ID and password defined in Policy Director. If not, define these, and apply group memberships as outlined in 4.5, "Defining users and groups" on page 79.

In our example, we continue to use the user Fred. Fred was assumed to have a user ID on `www.mycompany.com` of `fred` and a password of `ut76Sdc7`.

Step 2. Name the resource and identify it to Policy Director

To identify the resource to Policy Director, use the GSO Resources view on the Management console. Figure 39 shows this view.

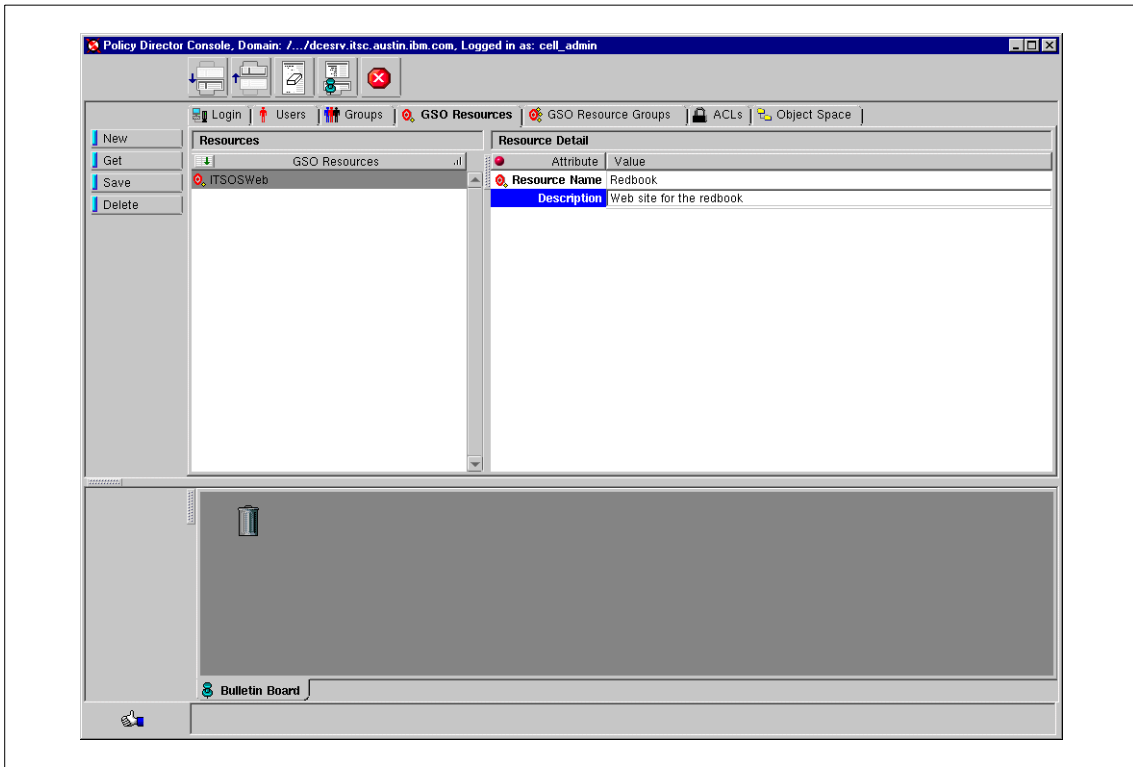


Figure 39. The GSO Resource name view on the Management Console

For clarity, the relevant detailed section of this screen is shown enlarged in Figure 40:

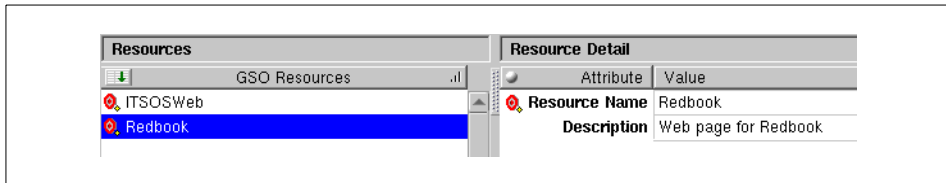


Figure 40. Detail of GSO resource definition

Step 3. Identify the mount point for the resource

At the moment, when the user accesses the desired URL on www.mycompany.com, he or she would type in something like:

`https://www.mycompany.com/Sales/monthly`

However, once `www.mycompany.com` has been smart junctioned with pluto, all the user's access URLs will begin:

```
https://pluto.mycompany.com/.....
```

It is important to decide on a sensible mount name. For our example we decided to call the mount point `Redbook1`. This allowed us to distinguish it from the GSO resource name, `Redbook`, but was similar enough to remind us of the relationship between the two. `Redbook` will be Policy Director's internal representation of the high-level object, `Redbook1`.

Step 4. Create the junction

It is at this point that the administrator has to go to the command-line interface. The actual creation of the junctions cannot be done from the Management Console.

In order to create a junction, the administrator has to be logged in as a DCE administrator. Issuing the command `dce_login` will cause a prompt to be presented where the administrator can log in as `cell_admin` (or whatever administration account there might be).

Once logged in, the administrator has to start the `junctioncp` utility. This is achieved by issuing the command:

```
# junctioncp -e <entry>
```

where `<entry>` is the host name of the WebSEAL server you will be adding the junction to. Figure 41 shows an example of doing this and what you can expect to see:

```
# dce_login
Enter Principal Name: cell_admin
Enter Password:
DCE LOGIN SUCCESSFUL
# junctioncp -e pluto
Attempting to bind to server at /./:/subsys/intraverse/secmgr/server/pluto
junctioncp> _
```

Figure 41. Starting the `junctioncp` utility

Once the utility is started, issuing the `help` command will show you all the commands available. We will be using the `create` command. Issuing `help create` would show you a list of all the options available with `create`.

The administrator now issues the `create` command:

```
# create -t ssl -b gso -h www.mycompany.com -T Redbook /Redbook1
```

where:

- t ssl** Determines the type of junction being created, in this case SSL.
- b gso** Specifies how authentication information will be passed to the back-end Web server.
- h <server>** Specifies the hostname of the junctioned server.
- T <resource>** Specifies the GSO resource you defined in Policy Director. This resource will be used later to hold the resource credentials for the user in which the server-specific user ID and password will be held.
- /<mount point>** Specifies the mount point on the WebSEAL server.

This results in the junction being created. Figure 42 shows this command entered and the result.

```
junctioncp> create -t ssl -b gso -h www.mycompany.com -T Redbook /Redbook1
Created junction at /Redbook1
junctioncp> _
```

Figure 42. The junction creation command

Step 5. Check the object in the namespace

The administrator should now be able to see the object in the namespace (though it will probably be necessary to use the refresh button).

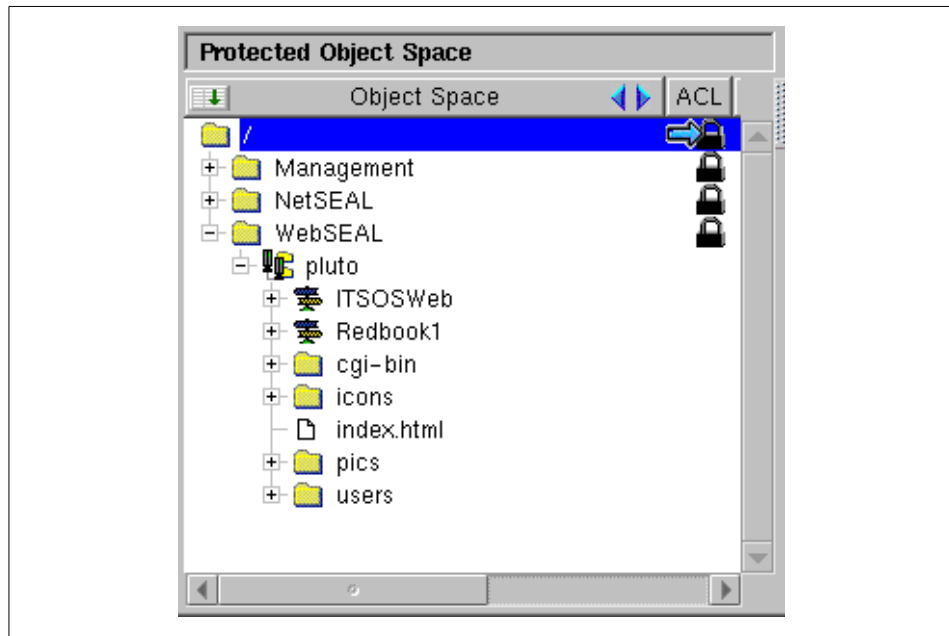


Figure 43. Detail of refreshed Object Space window

Redbook1 is now available underneath pluto. A user could now access `www.mycompany.com` by opening a browser and pointing at:

`https://pluto.mycompany.com/Redbook1`

Of course, we still need to think about authentication.

Step 6. Create the resource credential

The final step is to associate the user's Policy Director user ID with the user ID and password for the remote resource. This is achieved through a resource credential.

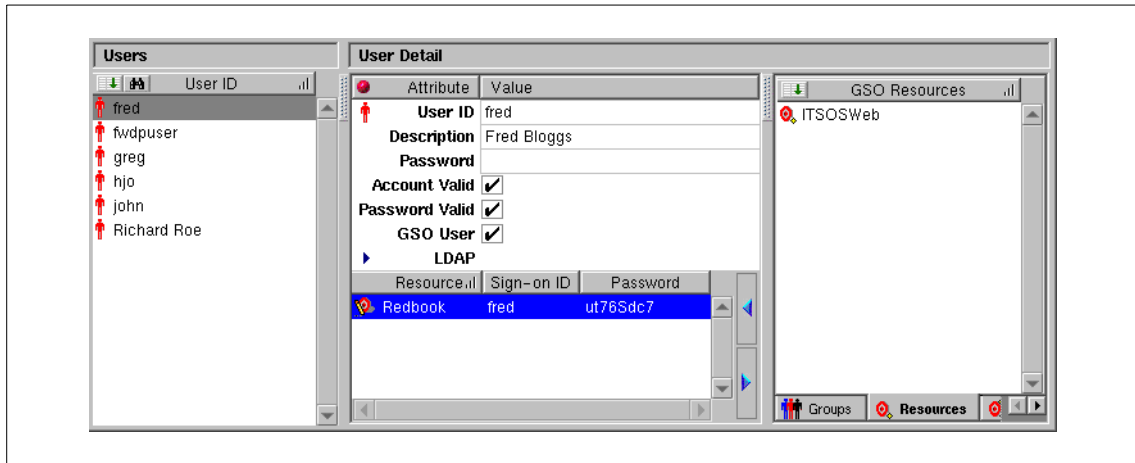


Figure 44. Assigning a resource credential

The above figure (Figure 44) is a detail from the Users view we have seen before. However, you will notice that `Redbook` has turned up on the screen.

Notice that the right hand pane now shows GSO resources. This pane can show Groups, GSO Resources, or Resource Groups depending on which of the tabs at the bottom of the pane is activated. Once the GSO resources are listed, an administrator simply drag-and-drops the desired resource over to the resource pane underneath the pane containing the user name. The administrator can then add the remote resource user ID and password in the fields shown.

The administrator simply repeats this last task for each user ID required, and the users can now log on to their desired Web application using their Policy Director user ID and password.

Extensive examples of creating smart junctions are included in the *IBM SecureWay Policy Director Administration Guide*.

The Object space and the query_contents program

You have seen that once the smart junction is created, it appears in the object space under the mount point assigned during creation. It is also possible to have the file system contents of the back-end Web server made visible through the Object Space view on the Management Console. This is achieved by installing the `query_contents` CGI script on the back-end Web server. This will then show the contents of the Web server as a list of objects within the container object represented by the mount point name. Once these objects are visible in the management console, they can be protected and authorized

like any other Policy Director object, and ACLs can be applied to them individually.

4.8 Managing administrators within the namespace

A key capability of Policy Director is the ability to delegate administration of select regions of the namespace to subordinate administrators. This section outlines the enabling concepts of the management namespace.

The following security considerations apply for the /Management object (see 4.7, “Policy Director namespaces” on page 96):

- The *management object* begins the chain of ACL inheritance for the management region of the namespace.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire management namespace.
- The traverse permission is required for access to this object.

/Management/Server

The /Management/Server container object of the Policy Director protected object namespace allows administrators to perform server management tasks (when appropriate permissions are set).

Server management controls are used to determine if a user has permission to create, modify, or delete a server definition. Server definitions contain information that allows other Policy Director servers, particularly the Management Server, to locate and communicate with that server.

A server definition is created for a particular Security Manager or Authorization Server as part of the installation process. The definition for a server is also deleted when the server is uninstalled.

The creation and deletion of the definition happens automatically; the installation administrator does not have to perform any special steps to create the definition. However, the administrator is granted modify (m) permission on the /Management/Server object in order to allow the creation of the definition during installation.

In addition, the administrator must have delete (d) permission on the /Management/Server object in order to delete the definition during uninstallation.

There are other operations that can be performed on a server definition:

- The user can view the definitions through the `ivadmin` utility. The user must be granted view (v) permission on the server object.
- The user can perform server management operations, such as start, stop, suspend, resume servers, or flush logs. The user must be granted server administration (s) permission on the server object.
- The user can modify parts of the definition using the `ivadmin` server modify command. The user must be granted modify (m) permission on the server object.

The permissions available for server management are listed and explained in Table 15.

Table 15. Server management permissions

Symbol	Access	Description
s	server	Perform server administration tasks (such as start, stop, suspend, resume).
v	view	List servers.
m	modify	Create a new server definition.
d	delete	Delete a server definition.

/Management/ACL

This object allows administration users to perform high-level ACL management tasks that can impact the security policy for the secure domain.

You must define ACL administrators in the `default-management` ACL object. See Table 16 for a list of the permissions associated with ACL management and what they mean. These permissions give the administrator power to create new ACL templates, attach ACLs to objects, and delete ACL templates.

Table 16. ACL management permissions

Symbol	Access	Description
b	browse	View the contents of the namespace below the object.
a	attach	Attach ACL templates to objects, remove ACL templates from objects.
m	modify	Create a new ACL template.
d	delete	Delete an existing ACL template. The ACL must contain an entry, with the control (c) permission, for this same user.

Symbol	Access	Description
v	view	List or view an ACL.

An ACL administrator cannot modify an existing ACL unless there is an entry in that ACL for the administrator containing the control (c) permission. Only the owner (permission c) of an ACL can modify its entries.

Note that the creator of a new ACL template becomes the first entry in that ACL with the *abcT* permissions set by default.

For example, if *cell_admin* is an entry in the *default-management* ACL with *m* permission, *cell_admin* can create a new ACL template. User *cell_admin* becomes the first entry in the new ACL with *abcT* permissions. The control permission (c) gives *cell_admin* ownership of the ACL and allows *cell_admin* to modify the ACL. User *cell_admin* could then grant administration permissions to other user entries in that ACL.

Ownership of the default-management ACL itself is given to both user *cell-admin* and group *iv-admin* by default.

Management/Action

This object allows administration users to perform ACL management tasks in a third-party namespace. The permissions available for action management are illustrated in Table 17.

Table 17. Action management permissions

Symbol	Access	Description
m	modify	Create a new action.
d	delete	Delete an existing action.

Policy Director provides authorization services to applications. Applications that are part of the Policy Director family include WebSEAL (for Web applications), NetSEAL (for TCP-based applications), and ObjectSEAL (for CORBA applications).

Third-party applications can make calls to the Policy Director Authorization Service through the Authorization API. Two necessary steps required to integrate a third-party application with the Policy Director Authorization Service include:

- Define the application's namespace
- Apply permissions on objects (resources) needing protection

The administrator of a third-party application namespace can use the `ivadmin` utility to define new permissions and actions. The administrator must have the `m` and `d` Management/Action permissions to create and delete these permissions/actions.

/Management/Replica

The `/Management/Replica` container object of the Policy Director protected object namespace controls the replication of the authorization database. High-level controls on this object affect the operation of the Management Server and the Security Manager(s) in the secure domain.

Replica management controls are used to determine what processes are allowed to read or update the master Authorization Policy Database in order for replication to take place properly.

Controls and associated permissions are listed and explained in Table 18.

Table 18. Replica management permissions

Symbol	Access	Description
v	view	Read the master authorization database.
m	modify	Authorize modification of the replica database(s).

All Policy Director servers maintain a local replica of the authorization database; this includes all security managers and authorization servers, which must be granted view (`v`) permission on the `/Management/Replica` object. The replication process requires that these processes be allowed to view and access entries out of the master Authorization Policy Database. The Policy Director installation automatically grants read permission to any server requiring access to the Authorization Policy Database.

Policy Director does not currently use the modify (`m`) permission in this context. The only way to modify the master policy authorization database is through the Management Console or the `ivadmin` utility. These tools are subject to other finer-grained checks. The modify permission is intended to be used in the future.

4.9 Accountability and monitoring

All accesses to WebSEAL, NetSEAL, and the Authorization Server can be monitored by the Policy Director's auditing facility, which generates an audit trail. Requests from the NetSEAL client can also be logged on a local file.

This section outlines some of the key logging and auditing files available.

Auditing is facilitated with several Policy Director and DCE files. The DCE processes are monitored using the auditing mechanism provided by DCE.

Policy Director server log files to consider include:

ivmgrd.log Management Server log
secmgrd.log Security Manager log
ivaclld.log Authorization Server log
nsid.log Directory Services Broker log

These log files can be found in the <install path>/<component>/log directories.

The DCE log files to consider include:

secd.log Security Server log
dced.log DCE Server log

Some Policy Director servers maintain their audit trail files in ASCII format. Configuration of general auditing is done in the iv.conf file, and the `audit_acl` permission controls specific auditing.

Each of the Policy Director servers has an audit trail file, in each case called audit.log, but held in its own log library. The following is an example of log files held in default libraries:

Management Server <install path>/ivmgrd/log/audit.log
Security Manager <install path>/secmgr/log/audit.log
Authorization Server <install path>/ivaclld/log/audit.log

Policy Director audit files for WebSEAL are:

wand_audit_log WebSEAL Audit Trail File

WebSEAL also has three standard logs that record activities rather than events:

wand_request_log
wand_agent_log
wand_referer_log

Some of the DCE audit trails are in binary format and require the use of a DCE command (in the `dcecp` utility) to read them:

```
dcecp> audtrail show
```

Such files include:

sec_audit_trail
sec_audit_trail.md_index
central_trail
central_trail.md_index

Please refer to the *IBM SecureWay Policy Director Administration Guide* and the appropriate DCE documentation (available online after installation) for detailed description and interpretation of these auditing and logging capabilities.

4.10 Scalability and availability

Policy Director can be incorporated into a large organization in a variety of ways to service high volumes of traffic. LDAP is used to store user and group information because of its vast scalability and flexibility – a key capability required for enabling large enterprise e-business applications.

Many servers involved in performing Policy Director functions are replicatable for scalability and availability reasons:

- DCE and LDAP directories can be replicated.
- The WebSEAL proxy can be replicated to offer higher availability by supporting the same Web pages to end users.

With smart junctions, backend Web servers can also be replicated for better load balancing of incoming client requests. The Web servers, which must be mirror images of each other, are mounted on the same junction point.

There are two approaches to deliver increased scalability:

- Utilizing Policy Director to manage multiple servers.
- Utilizing the IBM SecureWay Network Dispatcher to manage multiple Policy Director servers.

Policy Director can be configured to load balance between several back-end servers. Both WebSEAL and NetSEAL can provide this capability.

4.10.1 Replicated Policy Director

The Management Console and the Policy Director Authorization Service can be replicated to increase availability in a heavy-demand environment.

The master Authorization Policy Database, containing the ACL and credential information needed for authorization, is automatically replicated. Every application that uses the Authorization Service (WebSEAL, NetSEAL,

third-party applications) has its own local copy (replica) of the master. This relationship is illustrated in Figure 45.

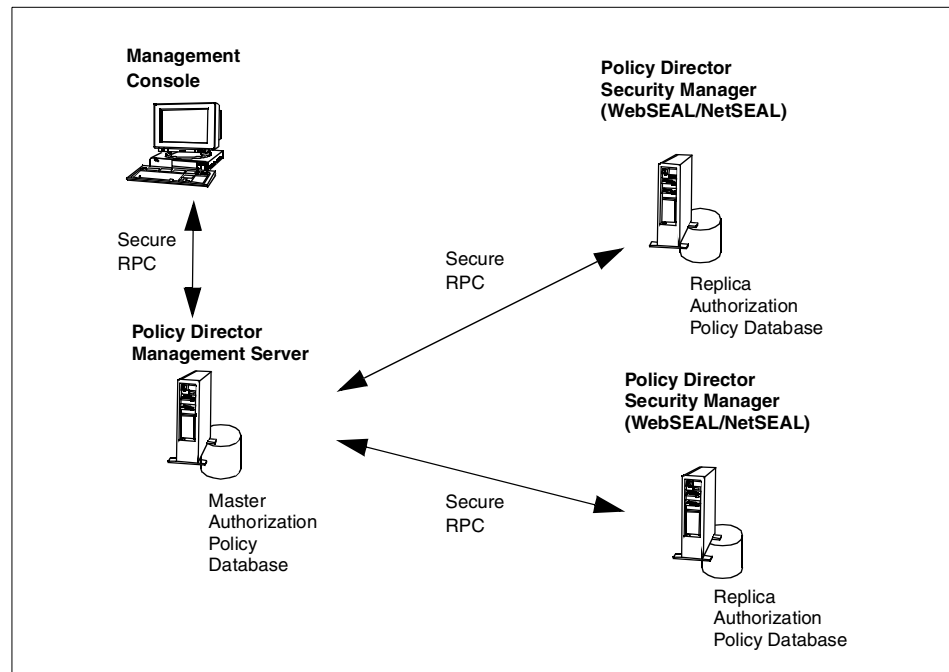


Figure 45. Separating Policy Director services across machines

The Authorization Policy database is cached in memory. An application server, such as WebSEAL, always obtains a fresh copy of the database at start-up. The database replica is actually cached in the internal memory of the authorization process associated with the application. Security administrators use the Management Console and/or the `ivadmin` utility to make changes to the security policy. The Management Server (`ivmgrd`) is responsible for updating these changes in the master Authorization Policy database. The Management Server then updates all replicas of the policy database.

In addition to these update notifications from the Management Server, the application servers also check the version of the master Authorization Policy Database at regular intervals to ensure they have not missed an update notification. If an update notification or pull update fails to reach a server, a log entry is created. In both cases, a retry mechanism also ensures the update happens in the future.

The cached authorization policy information results in improved system performance. For example, when WebSEAL does an ACL check, it checks the ACL in its own cached version of the database. WebSEAL does not have to access the network to obtain this ACL information from the master database. The result is very fast response times (performance) for ACL checks.

4.10.2 Utilizing the IBM SecureWay Network Dispatcher

In addition to using Policy Director to load balance back-end servers, the IBM SecureWay Network Dispatcher can load balance between multiple Policy Director servers to reduce potential bottlenecks accessing Policy Director services (see Figure 46). In this example, both Policy Director servers are cross-managing back-end servers - not only for load balancing, but also for high availability. In this configuration, user requests are sent by SecureWay Network Dispatcher to the least-used Policy Director server. Policy Director then makes the connection to the appropriate back-end resource if the authorization checks pass. These connections are illustrated with solid lines. As a further illustration of flexibility, Policy Director can load balance between multiple sets of servers to increase performance and provide another level of high availability. Policy Director keeps track of a count and the state of TCP port connections to determine the server to pass the request to.

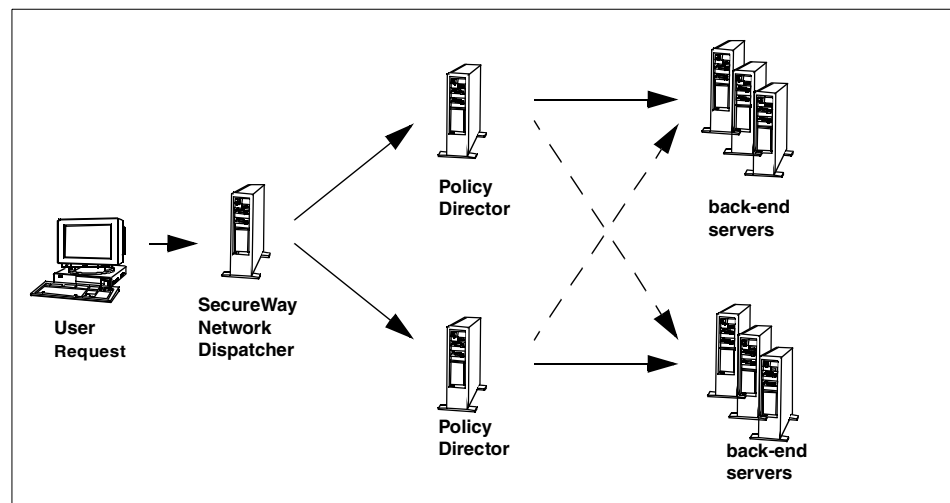


Figure 46. Load balancing Policy Director using SecureWay Network Dispatcher

More information about the IBM SecureWay Network Dispatcher can be found at:

<http://www.ibm.com/software/network/dispatcher>

4.11 Command line interface

Many administrative operations are performed with the Management Console graphical user interface of Policy Director. As mentioned previously, many of these commands, as well as many additional commands, are available from a command-line interface.

For instance, since junctioned servers are not created frequently, the `create` command is implemented only on the command line. In addition, many DCE commands are also available from a command line interface.

For a complete list of the commands, please refer to the *IBM SecureWay Policy Director Administration Guide* and the online DCE reference guides that ship with the product CD-ROMs.

4.12 Interoperability with other FirstSecure elements

Policy Director can interoperate with the other FirstSecure elements in a variety of ways. In FirstSecure Release 2, the interoperability between Policy Director and other FirstSecure elements focuses on mutually exclusive issues.

Specifically, Policy Director performs authorization functions using standard techniques (such as controlling access to HTTP requests over port 80) as well as providing customized authorization to applications via the Authorization API interface (Chapter 8, “The SecureWay Toolbox (TB)” on page 191).

Policy Director can accept public key certificates created by IBM SecureWay Trust Authority (see Chapter 7, “Public Key Infrastructure (PKI)” on page 167) for user authentication.

Another interoperability function is the exchange of user definitions between the Policy Director and the IBM SecureWay Boundary Server where the SecureWay Boundary Server is able to read user information from an LDAP directory that is managed by Policy Director. See the *IBM SecureWay Policy Director Administration Guide* for details about how to define IBM Firewall Proxy users through the Policy Director interface.

Chapter 5. The SecureWay Boundary Server (SBS)

Before starting to use and/or provide Internet services, or performing inter-enterprise networking, the connection of a company's network to an external site has to be established. This connection must not expose the internal network to the different security threats imposed by the foreign networks. Therefore, a complete part of the IBM SecureWay FirstSecure framework is dedicated to the task of protecting networks and their hosts from threats coming from other hosts or networks (Figure 47).

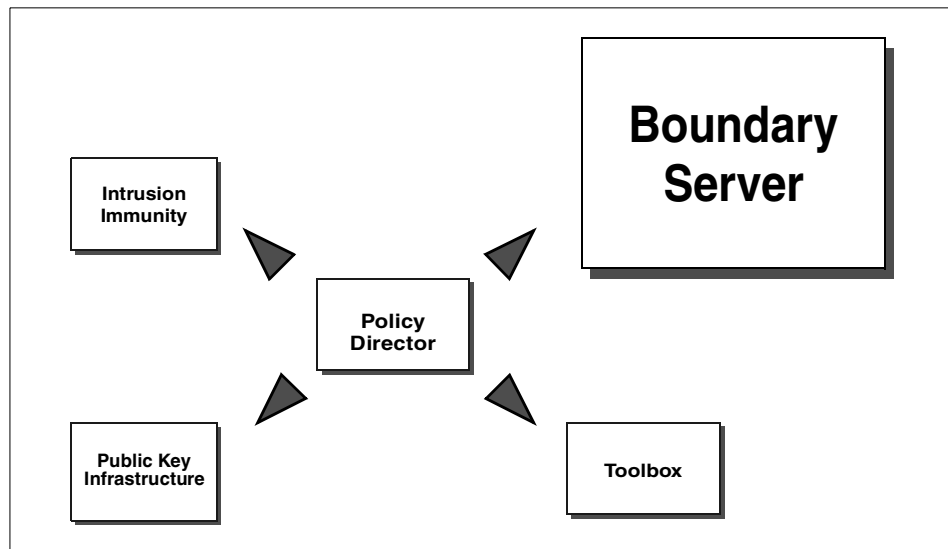


Figure 47. SecureWay Boundary Server

5.1 Introduction

There are some terms that you might not be familiar with, which are frequently used throughout the discussion of the SecureWay Boundary Server.

Most client/server applications that use the IP technology to send and receive data might use one or more of the following IP protocols: *TCP* or *UDP*. *TCP* stands for *Transmission Control Protocol*; *UDP* is an abbreviation for *User Datagram Protocol*.

TCP is known as a connection-oriented protocol, in which the protocol itself, not an application that uses it, will take care of the following common networking issues:

- Sequencing or ordering: Each packet has a sequence number to create an ordered flow of data. At the receiving end, an application receives the packets in the correct order, even if the network transmission caused the order to be out of sequence.
- Duplication: TCP will automatically discard repeated packets.
- Retransmission: Whenever a packet is lost on its path because of any event, TCP automatically requests that packet on the originating host.
- Checksums: TCP computes a checksum (16-bit number) of each packet and stores the computed value in the header when the packet is transmitted. The receiving end recomputes the checksum and verifies that it matches the value in the header. This guards against modification of the data in transit. (It does not guard against malicious tampering, however, when the checksum is altered along with the packet's contents.)
- Windowing: When sending packets, TCP can send a number of packets in a row without waiting for acknowledgment for each one before it sends the next. The window size defines how many packets may be in transit before it must wait for an acknowledgment. This prevents a faster sending TCP host from overrunning a slower receiving host.

These features make TCP a very reliable protocol. Applications that do not need these features, or that choose to implement some or all these features themselves, can use the UDP protocol.

When client/server applications use any of these protocols to exchange data, we can then define that particular communication between these two processes by the following tuple:

{Protocol, Source, Source Port, Destination, Destination Port}

Protocol indicates which protocol, TCP or UDP, is being used on that particular communication.

Source defines the IP address of the originating machine, while *Destination* indicates the IP address of the target machine.

Source port indicates the source port number assigned on the originating machine. Clients sometimes use dynamically assigned port numbers.

Destination port indicates the port being communicated to on the target machine. Server processes should conform to a certain and fixed standard when it comes to associate a certain service to a given port so that any client software knows at what port they should connect to receive an expected service. The association of ports and server services can be found in part in the services file on your machine (*/etc/services* on UNIX machines, for example) or comprehensively at:

<http://www.isi.edu/in-notes/iana/assignments/port-numbers>

For a given IP address of a server machine, it might have several services on officially or privately assigned port numbers, and each active service should have a server that would listen for incoming connections from clients. As an example, an *HTTP* server will normally listen for upcoming connections from clients at the *well-known* port 80 using the TCP protocol. With *well-known* port, we refer to the fact that this particular port is officially assigned to a particular application or service, in this case, the server handling the *HTTP* protocol. *Service* is an application protocol officially or privately assigned to a port number.

The tuple corresponding to an HTTP connection, for instance, would be as follows:

{TCP, <client IP address>, random port, <server IP address>, 80}

If a client establishes a second HTTP connection to that particular server at the very same time, the value for the *Source Port* will be different; so, there will be two different tuples for these two connections, different in at least one component at the very same time. Source port numbers will be reused on demand. Please note that, in addition to the tuple, the data that the applications are sending to each other and other control information is also sent across the network.

Computer networks that use the IP protocols are commonly referred to as *IP networks*. Organizations first build their own network across their organization with components, such as hosts, workstations, file servers, printers, routers, hubs, switches, and so on. This is what is called an *internal network* or *intranet*.

An IP network may also be called an *internet*. This should not be confused with the *Internet* (with the capitalized 'I'). The latter refers to the world wide network of interconnected networks and hosts that can be used for surfing the Web, inter-enterprise e-mail, conducting e-business, and the like.

Networks that are external to an organization, that is, that are under control of other companies, are commonly called *extranets*.

When networks are being connected to external entities, such as to other companies' networks or to the Internet itself, then security becomes an issue because private networks need to be protected from the threats that might come from any of those external entities. Also, some attacks may be attempted from internal sources as well; so, a network may need to be further protected, and additional means need to be in place on the internal network itself.

Among other means, such as secure authentication and authorization that are discussed and explained in the other chapters of this book, defining network boundaries and applying and enforcing security policies to such boundaries is important both inside an organization's private network and especially on the boundary to the external network.

A *firewall* is a generalized term for a special networking device that connects two networks in a secure way. Simply put, it filters traffic such that no attacks can pass through, but at the same time, allows traffic to pass through that is intended for the "other" side of the "wall" according to the business needs and security policies in place. A *boundary server* is basically a firewall, but it includes more functionality than what most firewalls provide.

5.2 Defining network boundaries

It is common sense that company networks have to be protected from the Internet by firewalls. This is a very simple statement, and it needs some additions to be applied when complex networks, sites providing complex Internet services, such as e-commerce, or sites that have several connections to different extranets need to be secured.

It is possible to classify the hosts of an organization as follows:

- Importance of the host's integrity:
 - Confidentiality of the host's data
 - Availability of a host's services
- Threat imposed by a host:
 - Degree of trusting the hosts' users
 - Risk that untrusted users or malicious program codes get access to the host

As a rule, the system administrator has to protect machines that are essential for the business and separate them from machines with considerable risk. The system administrator will try to minimize the risk for each of the hosts to become the target of attacks, but, in some cases, it might not be possible to get a reasonable small risk of getting compromised.

In a typical scenario, you might find hosts grouped as follows (Figure 48):

- External networks – These contain untrusted hosts that are not under the physical, computational, or legal control of the company. Examples are the Internet, inter-enterprise connection networks or extranets, and publicly accessible Internet cafes.
- Hosts that provide services to external networks – They have a high risk of being compromised due to possible misconfigurations or software bugs at operating system and application level.
- Workstations for employees – Trust to these machines can be raised to a fairly high level by removing floppy drives, educating the users, controlling their Internet activity, forbidding modems, and with anti-virus measures.
- Internal business servers – The confidentiality and availability of these machines are very important for conducting the business of your company. Examples are database servers, mail servers, and production systems.
- Dedicated administration and security services hosts – A very important group are the hosts that are required to enforce the security measures that we try to derive in this section.

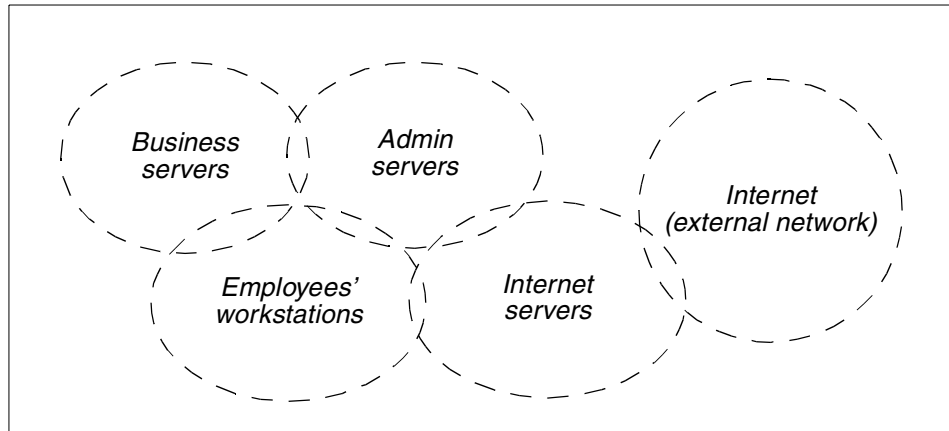


Figure 48. Differently rated networks

The following protections should be considered (Figure 49):

- Guard *all* hosts against the many extranets they might be connected to.
- Guard all workstations and internal servers against the Internet servers.
- Depending on specific security conceptions, essential servers might need to be guarded against all other groups.

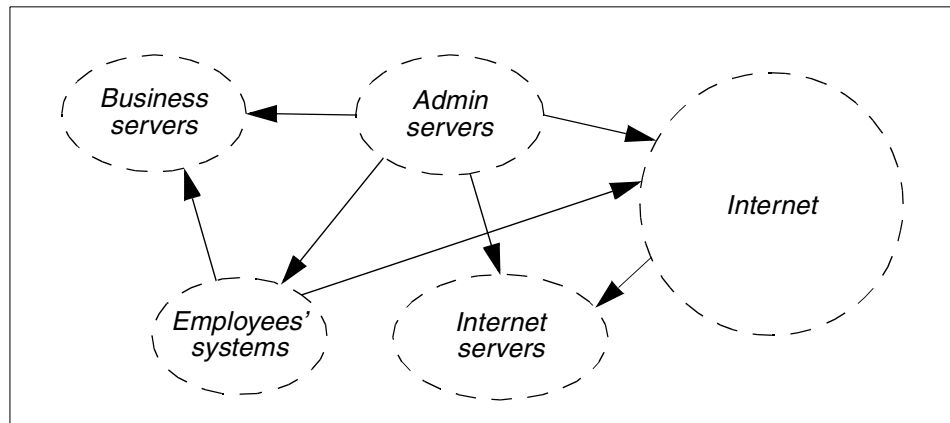


Figure 49. Required interconnections

Grouping the hosts is done by connecting hosts of the same group on one network. Where differently rated networks are connected, we speak of *network boundaries*. On network boundaries, we can usually define a *secure side* (secured side, internal network, clean network) with the network of higher security requirements, and a *non-secure side* (external side, dirty network) with the less trusted network.

On each network boundary, we need a secure boundary service that protects the hosts on the secure side against the threats from the non-secure side.

5.3 Secure boundary services

Secure boundary services are the measures on a boundary between differently rated networks to protect the secure network with all its hosts from threats from sources on the non-secure network (Figure 50). Traditionally, this is achieved by firewalls. Note that the firewall host plays a special role since, while it needs at least the same level of protection as the secured network, it is, at the same time, part of the non-secure network. Because of that, it needs special measures to protect itself. This section describes the two basic operation principles that are provided by the IBM SecureWay Boundary Server: Firewalls, content filters, and mobile security code.

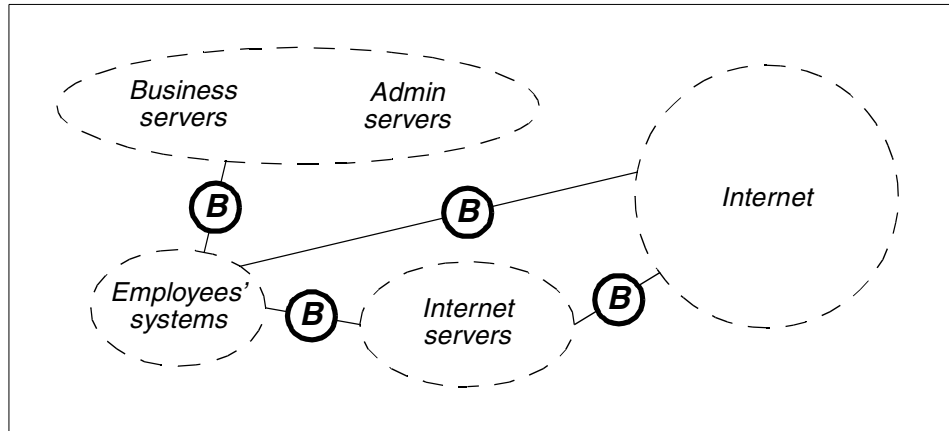


Figure 50. Network boundaries

5.3.1 Traditional firewalls

This subsection describes firewalls that protect against threats on the network and application level. Attack forms using a level that is viewed as data by the application protocol, for example FTP transferring program code containing viruses, are covered in the next subsection.

Firewall machines that protect networks against the Internet have a long history. There are very different concepts and implementations in use. The following parts introduce the most common mechanisms.

IP filter firewalls

These work just like routers, which may be configured to permit or deny IP packets depending on criteria, such as destination or source addresses and destination or source ports. Usually, this firewall type is combined with network address translation (NAT) to translate private address ranges to official Internet addresses. They provide the following advantages:

- High throughput
- No changes to client software
- No special means required for passing the target server address
- Firewall cannot be target of application level attacks

Stateful inspection

This is a relatively new firewall type. Firewalls based on this mechanism can be understood as IP filter firewalls that are conceptually extended. They not only statically filter IP packets as filtering firewalls do, but also remember

which connections are currently established and only allow packets to pass that open a new connection conforming to the rule set or belong to an existing connection. This mechanism is especially useful for UDP traffic and normal mode FTP data connections. Application level functionality is added by intercepting the protocol traffic and feeding it to special application gateways. They provide the same advantages as IP filter firewalls, plus the internal machines are partly protected from IP and application level attacks.

Circuit gateway firewalls (Proxies and SOCKS)

These kind of firewalls do not forward packets across its network interfaces but provide application independent programs to forward protocol data transparently. Clients connect to these programs on the firewall, where the connections are permitted or denied depending on criteria, such as addresses and ports. If appropriate, these gateway programs build connection to the destination servers. Then they pass the application data unfiltered. The advantages are as follows:

- Internal machines are protected from IP level attacks
- No changes to firewall IP stack

The simplest form of a circuit gateway firewall has *generic proxies*. The client connects to a given configured firewall port. The proxy program accepts the connection and then looks up a table using keys as the source address and port on the firewall. The proxy program then looks at the other values as permit/deny connection, destination address, and destination port. If the connection is permitted, the proxy opens a connection to the target address and port and then passes data in both directions transparently. The advantage of this approach is that no changes to client software are required.

The main problem of generic proxies, especially for outbound connections, is the absence of target server information in the clients access request. That information is derived from a limited table on the firewall. The *SOCKS* protocol overcomes this restriction. The client connects to a dedicated firewall port (usually 1080) and passes the information about the target server and target port, plus authentication data, to the socks server on the firewall. The socks server looks up a table with the following keys:

- Source address
- Destination address
- Destination port
- User name

If the connection is accepted, the socks server opens the connection to the target server and then passes data in both directions transparently. SOCKS Version 4 (SOCKS4) has only a very weak kind of authentication using an IP address and a user name that makes it suited only for outgoing connections. SOCKS Version 5 (SOCKS5) supports several advanced authentication methods, such as user ID/password, challenge/response, and SSL that enable it for inbound connections too. The advantages are as follows:

- All kinds of protection except application level.
- Client determines target host.

The client software must support the socks mechanism in order to take advantage of it. If you want to make clients work with the socks server included in the IBM Firewall, search the Web with your favorite search engine using the keywords *how to socksify*.

Application gateway firewalls

Application gateway firewalls utilize programs that are designed to understand the protocol that they transport. When the client connects to the firewall, there may be some kind of low-level authentication based on the client IP address. Then, some authentication within the protocol can be applied. If authentication succeeds, the gateway program opens a connection to the target server. It then, typically, provides some filtering of the application data. While this mechanism provides the highest level of security, it requires a specialized gateway program for each supported protocol and sometimes special measures on the client side. A common example of these kinds of applications are the HTTP proxy servers.

The following (Table 19) compares the different firewall mechanisms.

Table 19. Comparison of firewall mechanisms

	IP filter + NAT	Stateful inspection + NAT	generic proxy	SOCKS 4 (outbound only)	SOCKS 5	Applic. gateway
IP level protection of client	no	1)	yes	yes	yes	yes
Application level protection of client	no	1)	no	no	no	yes
Changes to client SW	no	no	no	yes	yes	yes, 2)
Client selects target server	yes	yes	no	yes	yes	3)

	IP filter + NAT	Stateful inspection + NAT	generic proxy	SOCKS 4 (outbound only)	SOCKS 5	Applic. gateway
Changes to FW IP stack	yes	yes	no	no	no	no
Exposure of FW	IP level	1)	IP level	IP level + socks	IP level + socks	IP level + appl. level
IP protocols, Services	TCP; UDP 5) all 4)	TCP, UDP; all 4)	TCP, UDP	TCP	TCP, UDP	HTTP, SMTP, ...
Authentication beyond IP address	no	1)	no	weak	strong	3)
External servers see client address	no	no	no	no	no	3)

Legend for Table 19:

- 1) Depends on implementation and protocol
- 2) Support for proxies often exists in standard client software
- 3) Depends on service and proxy
- 4) All only with NAT one-to-one mappings
- 5) Inbound/outbound cannot be determined for bi-directional protocols

These mechanisms alone do not necessarily make a firewall machine behave as intended. Appropriate configuration is required to control the direction of the traffic (inbound/outbound) and to restrict use of the services as defined by the security policy.

In particular, for all mechanisms that allow the client to select the target server, configuration measures are required to protect the internal network against external clients and to protect the firewall loopback interface against internal and external clients. Otherwise, they could bypass filters or IP address based authentication to connect to services on the internal network or to protected firewall IP services. For example, consider a firewall host with an HTTP proxy on port 8080 and an HTTP based management tool on port 800. The management tool runs on a port that can – due to some filter rules – only be reached from an administrator's PC. An unprivileged user that is allowed to use the proxy server can specify the URL `http://127.0.0.1:800/` and might then get connected with the management server.

5.3.2 Protocol content filters

As mentioned earlier, a secure boundary server does more than what traditional firewalls do. There are other threats to a network's integrity that require a much higher level of data traffic analysis: Trojan horses, viruses, malicious JavaScript, ActiveX, Java code, and incoming mails with bothering or offensive contents, to mention a few. These functions are usually not included in firewall programs, but can be interposed between firewall and internal hosts.

To find these threats in data streams requires rather complicated check algorithms being applied to large data blocks. Especially for HTTP traffic, they must be performed in real-time and, therefore, require high CPU performance.

Basically, two methods exist to detect malicious program code and other potential attacks:

Pattern search compares the suspicious program code with a set of signatures kept on a database. A *signature* is a piece of code that represents a known threat. For example, a certain virus can be detected because its code (signature) can be found within a program. Pattern search used a database of known signatures. To provide maximum protection, is important that this database is kept up-to-date with the latest signatures. This method is relatively simple and leads to high-protection against known virus and trojan horse threats but offers little or no protection for new or unknown viruses.

Code analysis tries to understand what the program code does. If it seems to be programmed to perform dangerous actions, such as formatting hard drives or establishing network connections, the program code fails the test. This method is rather complicated and computing intensive, especially with long programs. It, therefore, is not well suited for program downloads. Due to its technical reliability, it is not dependent on an actual database. Code analysis, on the other hand, is well suited for inspecting mobile code, such as Java applets and ActiveX controls.

5.4 Components of the IBM SecureWay Boundary Server

After introducing the threats a secure boundary server has to face and explaining the techniques that are available for protection, the next subsections show how the modules of the IBM SecureWay Boundary Server apply selected techniques to satisfy their tasks.

The IBM SecureWay Boundary Server consists of the following program components:

- The *IBM SecureWay Firewall* for the IP and application level access control on the dedicated firewall host.
- The *ACE/Server* from Security Dynamics for token based strong authentication.
- The *MIMESweeper* from Content Technology, consisting of *MAILsweeper* for SMTP traffic content screening and *WEBSweeper* for HTTP based traffic content screening.
- The *SurfinGate* from Finjan for blocking malicious mobile code.

These components are discussed in the sections that follow.

5.4.1 The IBM Firewall

This section introduces the features of the IBM Firewall 4.1 as part of the IBM SecureWay Boundary Server. It is beyond the scope of this book to explain the technology and detailed features of the IBM Firewall 4.1, as it is a fairly complex product. A good, and general, introduction to firewalls can be found, for example, in the following IBM Redbooks:

- *Protect and Survive Using IBM Firewall 3.1 for AIX*, SG24-2577
- *Guarding the Gates Using the IBM eNetwork Firewall V3.3 for Windows NT*, SG24-5209

Since the IBM Firewall 4.1 is a new version of the product, it is always wise to check the IBM Redbook Web site for the latest redbooks at:

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

Please also refer to the firewall product documentation that ships with the product.

The IBM Firewall 4.1 is available on AIX 4.X and Windows NT 4.0. With IBM Firewall Version 4.1, SMP (shared memory multiprocessor) machines are supported on both AIX and Windows NT platforms. An install wizard guides the administrator through the installation process. After performing some operating system dependent steps to harden the system platform, the firewall software is copied to the hard disk of the host. Then, the filter module is integrated into the operating system's IP stack, and some firewall components are hooked to the operating system as NT services or UNIX daemons. After rebooting, the firewall software is activated. More about

firewall installation considerations can be found in 9.3, “SecureWay Boundary Server” on page 238.

The IBM SecureWay Firewall implements IP filters, SOCKS 5, and application proxies. For many services, the best fitting mechanism can be selected depending on client issues, security, and performance considerations.

5.4.1.1 IP filters

The filter module provides static IP filters usually in combination with NAT. Filtering occurs by the following criteria:

- IP source address, IP destination address
- IP protocol (ICMP, UDP, TCP, and others)
- UDP or TCP source and destination port or ICMP type and code
- TCP flags (SYN, ACK)
- Interface of firewall (secure or non-secure interface or physical interface)
- Firewall local or routed traffic depending if the firewall machine is a connection endpoint
- Flow direction (inbound or outbound as seen by the firewall IP stack)
- IP fragments can be permitted or denied
- Time of day and date, for example, to ban traffic not important for business from working hours

5.4.1.2 SOCKS 5

The IBM Firewall implements circuit level gateway functionality with SOCKS 5. It provides the following features:

- Relaying of TCP and UDP connections
- Filtering by source and destination address, destination port, user, and authentication success
- Authentication with the firewall authentication mechanisms
- Support for SOCKS 4

The IBM Firewall provides application proxies for the most important Internet services. The following section describes the HTTP, Telnet, FTP, and SMTP proxies, as well as a DNS server on a firewall.

5.4.1.3 HTTP proxy

A specialized version of the IBM Web Traffic Express for Multiplatforms (WTE) serves as *secure HTTP proxy*. It has caching and some less important

features of WTE turned off. Protection and firewall authentication features are included. It is able to proxy not only HTTP connections, but also HTTPS and FTP via HTTP. If installed on a Windows NT host, it can invoke Finjan SurfinGate (see also 5.4.4, “SurfinGate” on page 139) for code inspection. User and group administration for proxy users can be done on the firewall or, better yet, through the Policy Director.

5.4.1.4 Telnet proxy

The IBM Firewall *Telnet proxy* allows you to relay Telnet for inbound and outbound connections. In *transparent Telnet proxy* mode, any user from the internal network can connect to Internet servers without authentication on the firewall. For example, if a user wants to connect as *anonymous* to server telnet.domain.com, he or she types, at the firewall Telnet prompt, `anonymous@telnet.domain.com` and then, at the password prompt, the password for the target server authentication. In *Telnet proxy* mode, the user has to authenticate on the firewall with the method that the administrator has configured for it. The user can then use only very few commands on the firewall, such as `telnet`, `ping`, `passwd`, and `exit`. This mode is useful not only for outbound Telnet session, but – in combination with strong authentication – especially for inbound connections. The third mode supported by the Telnet proxy is the *server* mode. This mode, however, is supported by the IBM Firewall natively but disabled when SBS is installed. The user and group administration for these proxy users can be done on the firewall or, better yet, through the Policy Director.

5.4.1.5 FTP proxy

The modes of the *FTP proxy* are very similar to the Telnet modes. In *transparent FTP proxy* mode, the user authenticates with `anonymous@ftp.domain.com` and then enters the password for the target server. In *FTP proxy* mode, the user authenticates to the firewall with user ID and password (or some other configured authentication method). Then the user can only use the FTP command `quote site ftp.domain.com`. After establishment of the connection, the user has to enter `user anonymous` and is then prompted for the password. From then on, all commands work just as if the client had a direct connection to the server. The third mode supported by the FTP proxy is the *server* mode. It allows access only to the firewall file systems. The user must be a firewall admin user. The user can connect to the firewall, just like to any other host, and has access to all files on the machine with the respective privileges. Just like with Telnet proxy users and groups, the user and group administration for these FTP proxy users can be done on the firewall or through the Policy Director.

5.4.1.6 Mail proxy

The secure mail proxy with IBM Firewall 4.1 runs in daemon mode and supports ESMTP (Extended Simple Mail Transfer Protocol). The mail proxy supports anti-spoofing and anti-spamming. It forks some child processes before the first mail connections occur. With the GUI, it is possible to specify triples consisting of an external mail domain name, the related internal mail domain name, and the internal mail server. The mail proxy relays mails to the target mail server. This might not be possible because the target mail server may be off-line. Because the mail proxy does not have a queuing mechanism, it needs an *overflow server* to handle the mail for the currently unreachable user mailboxes. This overflow server may be an ordinary SMTP mail program, such as sendmail, that can even be running on the mail proxy itself. If it is running on the firewall, it must not use port 25 because this is already occupied by the secure mail proxy. The overflow server queues the mails and tries to deliver them at a later time.

5.4.1.7 Virtual Private Network support

The IBM Firewall includes VPN support based on the CDMF (40-bit DES), DES, or triple DES IPsec standards for encrypted tunnels over non-secure networks and MD5 or SHA for authentication. The firewall can be configured to not use any encryption or authentication method. Application independent secure connections between different hosts can be established over non-secure networks using VPNs. Theoretically, all products supporting the IPsec standard can communicate with IBM Firewall hosts using IPsec VPNs.

5.4.1.8 Network Security Auditor

Part of the IBM SecureWay Firewall for AIX is the Network Security Auditor (NSA). It is a network security scanner that scans hosts over the network. NSA allows an administrator to perform a variety of checks, such as UDP and TCP port scans, examination of active services for known weaknesses, and auto detection of services on unregistered ports. It compares the result with a policy and generates reports of the scan results. It can not only be applied to firewalls, but also to virtually every host on the network.

5.4.1.9 Network Address Translation (NAT)

NAT is a mechanism that allows use of private (RFC 1597) addresses or “stolen” IP addresses without getting routing problems on the Internet. Packets that originate from hosts on the internal network get their sender address translated on the firewall to an address that is part of the firewall’s non-secure network and that is managed by the firewall. It then sends the packet on to its Internet router. If a packet from the Internet arrives at the firewall addressed to that external address, the firewall translates the target

address to that of the internal machine and sends the packet on to the internal network.

The firewall not only translates the IP addresses, but also recalculates the packet's checksum. To make FTP normal mode data connections function correctly, it also has to recognize the PORT command in outbound FTP data connections and change the embedded clear-text client IP address.

NAT provides various mechanisms to meet different needs. The *one-to-one* mapping or Map configuration reserves an external, "official" address for exclusive use by an internal host. This is useful for hosts that frequently pass the firewall with NAT, and it is a must for servers that should be reached from a non-secure network. The *many-to-one* mapping provides only one official address for several internal machines. The firewall can recognize to which client the answer packets belong.

5.4.1.10 Firewall administration

The IBM Firewall may be configured and administered using a Java based graphical user interface (GUI) from a Windows 95, Windows NT, or AIX workstation, or directly from the graphic display of the firewall. For remote connections, it is possible to select between SSL or unencrypted traffic.

The IBM Firewall comes with a comprehensive set of predefined firewall services and socks rules to simplify configuration of the connections. *IP filter connections* are generated by combining *network objects* with *firewall services*. Firewall services are generated from *firewall rules*. *Socks connections* are generated by combining *network objects* with *socks rules*.

During installation, the IBM Firewall disables unsafe applications to ensure a secure platform for the firewall to run on. This process is called *operating system hardening*. The steps performed on AIX briefly are:

- Disabling Common Desktop Environment CDE
- Removing unnecessary entries from /etc/inittab
- Removing unnecessary daemons from /etc/rc.tcpip
- Removing unnecessary entries from /etc/inetd.conf
- Removing all unnecessary operating system users
- Removing all unnecessary operating system groups
- Deleting all unowned files and directories
- Erasing all file system permissions for non-secure applications

The IBM Firewall provides functions for generating extensive activity and event logs. They can be monitored at real-time to generate alerts being transferred to other hosts or archived for later examination. Performance statistics can be generated. In general, it is good use to store the logs not (or not exclusively) on the firewall, but to forward them to a machine in the secure network, therefore having them in a safe place for analysis after any suspicious event. The log maintenance and archiving routines can be installed separately from the base firewall software.

The IBM Firewall supports several authentication methods. These can be selected per user, protocol, and secure/non-secure sides. Besides the trivial choices of permit all and deny all, an administrator can select a firewall password or SecurID card. On Windows NT, the IBM Firewall additionally provides for a user-supplied authentication and NT logon password authentication.

5.4.1.11 Enterprise Firewall Manager

The Enterprise Firewall Manager (EFM) is a program available with the IBM Firewall for AIX. It must be installed on a machine with the firewall software. It allows an administrator to remotely configure and administer other IBM Firewalls through the means of the common firewall administration GUI. The communication between the enterprise manager and the firewalls goes via a manual IPSec tunnel, which has to be set up between the enterprise manager host and the other firewalls. It brings a simplified and secure way to manage a group of firewalls.

5.4.2 MAILsweeper

With IBM SecureWay FirstSecure, the tool of choice for screening SMTP mail contents is the MAILsweeper from Content Technologies, Ltd. It consists of three parts: The MAILsweeper itself, a the MAILsweeper policy editor, and the MAILsweeper manager GUI. All these components are installed on one machine. The MAILsweeper manager can additionally be installed on a different machine for remote management. Both installation options require Microsoft Management Console to be previously installed, which, in turn, requires Microsoft Internet Explorer.

MAILsweeper runs on Windows NT where it establishes three Windows services (Figure 51):

- *MAILsweeper for SMTP Receiver* for listening on port 25 and accepting SMTP messages sent to the MAILsweeper host.
- *MAILsweeper for SMTP Security* for performing the actual checks regarding MAILsweeper's security policy.

- *MAILsweeper for SMTP Delivery* for delivering messages that passed the security checks to their destinations or the next SMTP chain hop.

MAILsweeper checks mails and their attachments for a configurable size limit, or it might delay transmission of mail with oversized contents. It also looks for types of attached files, for keywords, phrases, and combinations thereof. Accordingly, MAILsweeper can classify, for example, the content's mail as junk, offensive, or confidential. MAILsweeper can check compressed files in several popular formats, such as ARJ, UNIX compressed, ZIP, GZIP, UNIX Tar, or MIME. You might choose to quarantine, or send to other user-defined areas, the files that are of an unrecognized data format. Graphics format files, such as GIF, JPEG, BMP, or TIF, are recognized, too. Depending on the policy criteria, the mails' direction, and the type of files attached, the mail can be delivered, quarantined, or dropped. The appropriate settings can be user dependent. If the mail has recipients with different security policy settings, delivery is split to allow different handling.

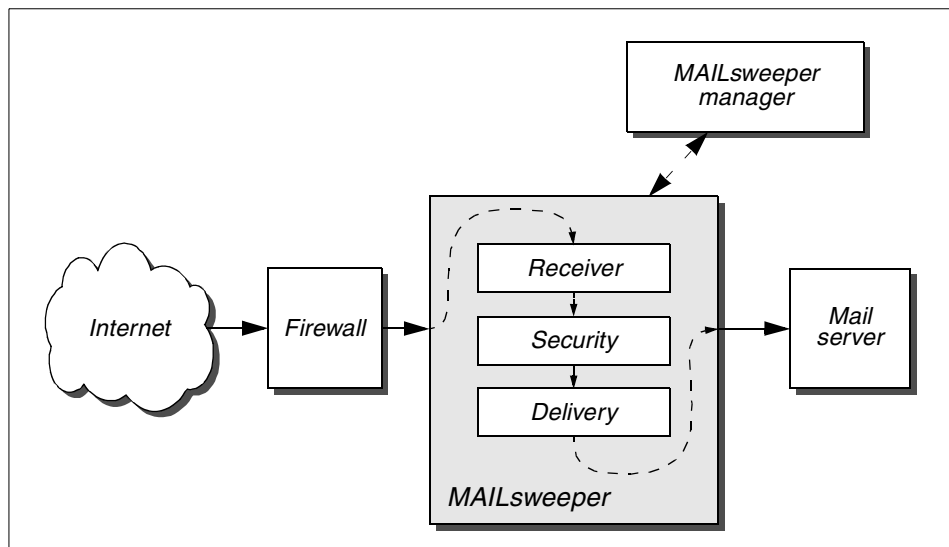


Figure 51. Logical mail flow

If mail messages exceed a customizable size limit, their further delivery can be delayed to times of day with low network traffic.

An anti-virus check program can be integrated with MAILsweeper that can detect infected attachments and eventually clean them. Norton AntiVirus command line version, as packaged and supported with FirstSecure, is one possibility. Other supported anti-virus products include: *Dr. Solomon's*

Anti-Virus Toolkit, McAfee VirusScan, Sophos Anti-Virus, F-Prot, Thunderbyte Anti-Virus, VET Anti-Virus, and H+B EDV. MAILsweeper can quarantine infected messages and inform the sender, recipient, or administrators about the incident or send the cleaned message to its recipients.

All actions performed by MAILsweeper can be collected in a log file. If security relevant events occur, notification mails can be sent to the mail sender, the mail's recipients, or the local site's administrators. The notification behavior can be fine tuned depending on mail direction and the kind of security event.

Installation of MAILsweeper is simplified by an installation wizard that checks the software prerequisites and prompts for essential information, such as company name, DNS mail domain, address of mail server, and mail gateway.

Further configuration and administration is done with the MAILsweeper policy editor. The policy editor allows selection and tuning of all the MAILsweeper features, such as anti-virus tool invocation, blocking due to content ratings, and so on.

5.4.3 WEBSweeper

WEBSweeper, from Content Technologies Ltd., is a tool for filtering HTTP and FTP over HTTP traffic. It provides various filter mechanisms that can independently be enabled and configured including anti-virus check, PICS rating, keyword blocking, content screening, and code blocking. In addition, WEBSweeper also acts as an HTTP proxy. These features and a brief description of administration and dependencies are discussed in the sections that follow.

WEBSweeper checks the data portion of FTP and HTTP downloads for suspected file name extensions and file content for known extensions. These extension typically indicate file types that may contain viruses, especially executable code or application macros, or they indicate file archives or compressed files. Compressed files are deflated, and archives are taken apart recursively until only plain files remain. The files that are subject to viruses are fed one-by-one to an anti-virus checking program. Depending on the result (clean, infected, or unknown), the original data is quarantined or delivered. Similar to MAILsweeper (see last section), WEBSweeper supports various third-party, anti-virus products. The anti-virus checking of WEBSweeper is optional if other means for anti-virus are in place.

PICS (Platform for Internet Content Selection) is a system for rating the contents of Web pages. A *label* describes the contents of a Web page with

some describing attributes, such as, for example, the level of violence. Some more information, such as date of label creation, MD5 hash, etc., may be included as well. Labels can be provided by a Web server for its own pages (in HTML `<META>` tags or HTTP headers) or by third parties.

The WEBSweeper configuration allows blocking Web pages that exceed some limits or that contain no PICS rating at all.

WEBSweeper has the capability to block the transfer of program code types, such as executables or Java, or if the code is not certified. Within IBM SecureWay FirstSecure, SurfinGate, from Finjan Software Ltd. (5.4.4, "SurfinGate" on page 139), specializes exclusively on code scanning and might, therefore, be a valuable alternative.

WEBSweeper has a built-in caching proxy function for HTTP and FTP traffic. This may be a valuable feature for sites with a relatively low amount of Web traffic. Typically, WEBSweeper is used primarily for screening the traffic, which, by the nature of the protocols, must be done online. Processing power should, therefore, be dedicated to the screening process, and caching may be disabled if a separate caching server is available, and caching is a requirement. You can also choose to install WEBSweeper HTTPS Proxy (WSWHTTPS), which is an NT service that adds HTTPS support to WEBSweeper.

WEBSweeper runs as a Windows NT service that is automatically started after the machine's bootstrap.

Despite some crucial configurations with the graphical install wizard, WEBSweeper is configured by editing configuration files.

WEBSweeper features enhanced log and display functions. All actions of WEBSweeper may be logged to files. If an HTTP page is blocked due to violation of the security policy, the client user gets a page with an appropriate message.

WEBSweeper can be used in chain with other HTTP proxies or even in chain with other WEBSweeper installations. It does not support SOCKS.

If performance is of concern, WEBSweeper should run on a dedicated machine with no other applications to reduce latency especially for HTTP traffic. It listens on port 80 of its host. It can be configured to forward all requests to another proxy (proxy chain).

5.4.4 SurfinGate

SurfinGate, from Finjan Software Ltd., is a content screening tool for checking mobile code in HTTP and FTP traffic for malicious actions. SurfinGate allows HTTPS traffic to pass through. It can be used as a stand-alone tool within an HTTP/HTTPS proxy chain or as plug-in for the IBM Firewall secure HTTP proxy using the WTE API (Windows NT only).

5.4.4.1 Servers, consoles, and databases

SurfinGate consists of three parts that all run under Windows NT: The server that actually performs the inspections, a database for storage and retrieval of user settings and previous inspection results, and the console for supervising the server locally or remotely.

During installation, the system administrator selects which of the components should be installed. The SurfinGate server runs as a Windows NT service and behaves like a HTTP proxy server, listening on TCP port 8080 by default.

While the principal checking mechanism of SurfinGate is code inspection, due to performance reasons, it works with a database of applet IDs. For each inspected applet, a unique ID is generated and stored in the database together with the applet's derived security characteristics. Therefore, the same applet need not be checked another time if it is downloaded again. Two different database programs can be used, either a built in Microsoft Access module or an Oracle database, which must be purchased and installed separately from SurfinGate.

It is possible to have more than one SurfinGate server on different hosts working together. If they use the database, one of these servers must be designated as a primary server that hosts the database, while the others access this database.

5.4.4.2 Content screening

In particular, SurfinGate can be configured to perform *content inspection* of Java, ActiveX, and JavaScript code. SurfinGate can also *block* applets, controls, and JavaScripts regarding inspection result and a per user, per department or company, security policy basis. Furthermore, SurfinGate can generally block or allow Java, ActiveX, JavaScript, VBScript, plug-ins, and cookies.

SurfinGate supports *Access Control Lists (ACLs)* on applet level, regarding file system access, network access, and invocation of application files with given extensions. The security characteristics of an applet or ActiveX control can be inspected and compared with the security policy.

SurfinGate supports JDK 1.1 *digital signatures*.

5.4.4.3 Administration

Finjan SurfinGate is administered from a management console, the *SurfinConsole*. The SurfinConsole can be used to supervise several SurfinGate servers. All security events are logged to a native log mechanism. Activity reports can be generated to gain information about specific event types. To simplify administration, SurfinGate can auto detect users that access the server but are not yet known to the system.

SurfinGate can also notify administrators by email upon important events.

Applets that did not pass the screening check and security policy comparison may be substituted by a customized applet notifying the user of the problem.

5.4.4.4 Dependencies

If the MS Access database is used, all the SurfinGate servers and the SurfinGate consoles must be able to reach the primary SurfinGate server, where the database files are stored, via the SMB protocol.

5.4.5 Security Dynamics SecurID authentication system

The Security Dynamics ACE/Server security solution is included in the IBM FirstSecure framework as an alternative authentication system for customers who need this kind of strong authentication. It typically can be used to give single users anywhere on the Internet access to secured resources, for example, to let a travelling employee access to their company e-mail box.

SecurID authentication provides a two step logon procedure to protected systems. The user has to provide his or her user name as with normal password authentication. Then, the input of a PIN (personal identification number), just like a password, is expected. The last step is providing a code that is generated by an electronic device, the SecurID token. Each code is valid only for 60 seconds, then another, unpredictable code is generated. The ACE/Server determines the actual code for that user and token by using the same algorithm as the electronic device to ensure correct code verification. Of course, the server and the device must be synchronized.

This kind of authentication is much safer than simple password authentication. Guessing a token code from a dictionary, as is a popular approach for cracking password authentication, is impossible. Eavesdropping of the code is nearly useless because it is valid, typically, for only 60 seconds and can be used only one time for authentication. Stealing the device or card is of little use as long as the PIN remains secret.

Keep in mind that only the authentication is secured. No encryption of the following data traffic is provided; so, it is still open to eavesdropping and session stealing.

A system for SecurID authentication consists of the following components:

ACE/Server – Administers the user database and answers questions from the ACE/Clients about authentication success. A *primary (master)* server may be substituted with *secondary (slave)* servers for backup purposes. The ACE/Server software is supported on AIX and Windows NT.

ACE/Agent – Is a program that protects access to the computer where it is installed by providing SecurID authentication.

ACE/Client – Is the client that makes use of the ACE/Agent. It directs the protocol specific authentication dialog with the user.

SecurID tokens – The electronic devices that provides the user with the actual token.

Remote administration is provided for the ACE/Server with a GUI. It is only available for Windows platforms.

In a typical authentication scheme, a user connects to the firewall where the ACE/Client controls access. The user has to enter his or her PIN code and the number generated by the token for that very minute. The ACE/Client communicates over an encrypted connection with the ACE/Server to check if the given numbers are correct. Access is then granted accordingly.

For more information about the Security Dynamics ACE/Server, please visit:

<http://www.securitydynamics.com>

5.5 Building secure boundaries

Now that the components that constitute the SecureWay Boundary Server and what they do have been introduced, let us have a look at how they work together to build secure boundaries. The most common boundaries are discussed in the sections that follow.

5.5.1 Secured SMTP gateway

Combining the different e-mail processing components, it is possible to build a reasonably secure gateway for inbound SMTP traffic. The components that can be used for building SMTP chains are: IBM Firewall secure mail proxy,

MAILsweeper, and a third party anti-virus product (not part of IBM SecureWay FirstSecure).

Note

SMTP traffic is the only service that almost every Internet gateway has to provide for the non-secure network. Therefore, it requires special attention because intruders that aim at your company can easily confront you with arbitrary SMTP based-threats.

The following features are available to process SMTP mail:

Antirelaying – Met by IBM Firewall secure SMTP proxy or MAILsweeper.

Checking for keywords and text patterns – To detect junk mail, offending material, or confidential data. It is provided by MAILsweeper.

Exceeding size limits – Checking for exceeding the size limit. It is provided by secure SMTP proxy or MAILsweeper.

Note

Mail relaying is an attack form classifiable as resource theft. An attacker may want to deliver spam mail to a large number of recipients. The attacker establishes a connection to any relay enabled SMTP server on the network. A long list of recipients is then transferred, followed by the mail body itself. The SMTP server tries to deliver the mail to all of the recipients as requested anywhere on the network. Besides the fact that these recipients usually are not glad about receiving junk mail, the resources of the victim's SMTP server site, especially network bandwidth, are extremely loaded by transferring hundreds or even thousands of mail messages. (For more information about such spam mail, please check <http://www.orbs.org> or search the Web for *spam*.)

There are several possibilities for constructing the SMTP chain by selecting and combining these measures. There is no single and perfect way for all situations; therefore, we want to give a few hints and a few recommended scenarios:

- Use the IBM Firewall secure mail proxy as the first gate for incoming SMTP traffic. It is designed for the purpose of eliminating low-level threats, such as mail relaying, mail spoofing, bad syntax of sender and recipient addresses, and so on.

- Perform all checks that may reduce mail bandwidth (size check, junk mail check) before performance intensive checks (anti-virus).
- For outgoing mail, it might be necessary to prevent client hosts from going round the SMTP chain.

A simple, but comprehensive, SMTP proxy chain for incoming SMTP mails might be constituted as shown in Figure 52:

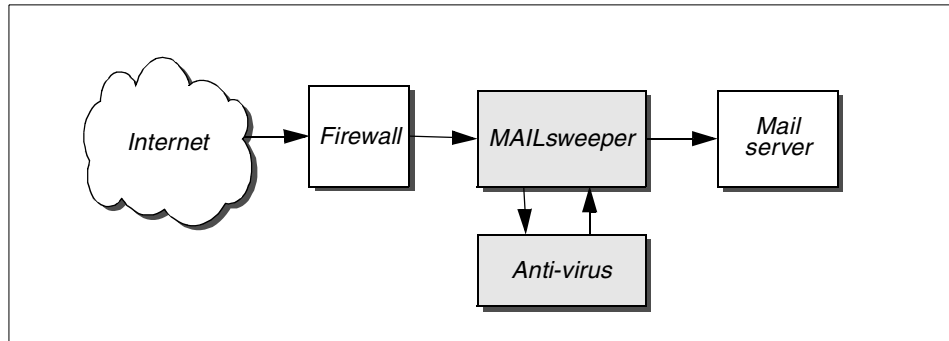


Figure 52. MAILsweeper with anti-virus (inbound mail)

An IBM Firewall with secure mail proxy works as front end. It rejects relay and spoofing attempts and refuses to accept oversized mails. Its overflow server is sendmail listening on, for example, port 26 of the loopback interface. Both programs are configured to forward inbound mail to the MAILsweeper host.

MAILsweeper then performs checks for junk contents and eventually blocks the mail. If there is data attached, MIMESweeper decompresses and splits archives and invokes the anti-virus tool, such as Norton AntiVirus command line version, running on the same machine. Finally, clean mails are delivered to the mail server.

Here is a possible solution for an outbound SMTP chain (Figure 53):

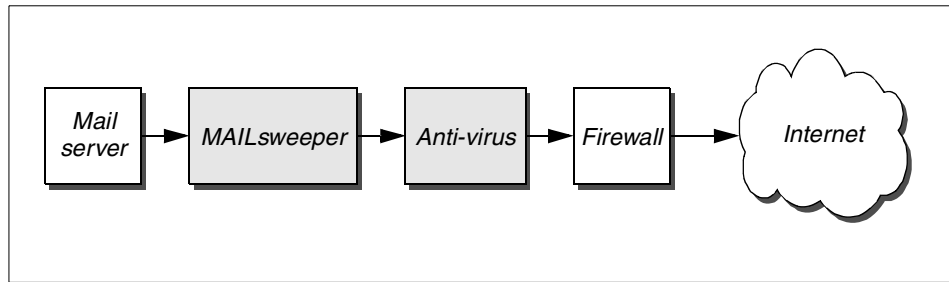


Figure 53. Mail outbound chain

The mail server sends outbound message to the MAILsweeper host. MAILsweeper checks for confidential or offending contents, then eventually delays further processing of the message due to size and time of day. The next action is transmitting the mail to the anti-virus host. The anti-virus software checks the mail for viruses and eventually passes it to the firewall for delivery. The secure SMTP proxy of the firewall delivers the mail to the appropriate Internet destination.

In this case, measures are required to prevent clients from bypassing the content screening hosts. One possibility is to put the content screening hosts into a DMZ (that must not contain Internet servers). The firewall allows outbound mail per NAT from the secure network only to the MAILsweeper host and mail from anti-virus to the secure SMTP proxy.

5.5.2 HTTP traffic

The HTTP traffic path describes the push from the clients (users) in the secure network through the SecureWay Boundary Server to the non-secure network. We only consider a “full” path consisting of WEBSweeper, SurfinGate, anti-virus, caching proxy, and firewall. HTTP and FTP, via HTTP, are handled together.

The following features are available for HTTP and FTP, via HTTP traffic handling:

- Anti-virus checking (anti-virus scanner with WEBSweeper and/or IBM Firewall secure HTTP proxy).
- Inspection of mobile code (Finjan SurfinGate stand-alone, Finjan SurginGate with IBM Firewall secure HTTP proxy).
- Inspection of mobile code (Finjan SurfinGate stand-alone, Finjan SurfinGate for Firewalls).
- URL filtering, content rating (WEBSweeper).

- Cache (WEBSweeper, Web Traffic Express as an additional product).

Note: Finjan recommends to deploy SurfinGate in a dedicated network on a third firewall adapter (DMZ) to prevent users from accessing the SurfinGate data space. This applies only if there are no Internet servers in the same DMZ.

We introduce two scenarios, one with little HTTP traffic and another with much HTTP traffic. These scenarios are not recommendations; they point out possible solutions.

5.5.2.1 Low-volume HTTP traffic

We assume that the traffic does not bring WEBSweeper with an anti-virus check or SurfinGate on their particular machines to full CPU utilization during peak hours. The firewall platform is Windows NT; so, we can use SurfinGate and NAV for Firewalls as an WTE (Web Traffic Express) plug-in.

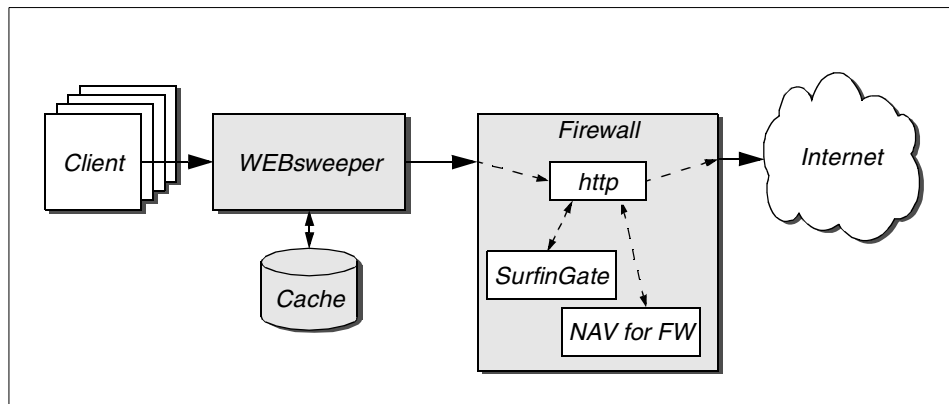


Figure 54. WEBSweeper with low-volume HTTP traffic

As shown in Figure 54, the clients direct their requests to the WEBSweeper host (measures may be required to prevent users from connecting directly to the firewall as their proxy). WEBSweeper can have caching enabled (if SurfinGate does not have different security profiles for different users) so that it can already fulfill some requests. If the cache does not contain the requested data, WEBSweeper connects to the firewall proxy. The firewall proxy connects to the target server. The firewall proxy passes the reply data to the SurfinGate plug-in. A SurfinGate server is required in conjunction with the SurfinGate plug-in for Java applet and ActiveX screening.

5.5.2.2 High-volume HTTP traffic

With many users accessing the external Web simultaneously, both SurfinGate and WEBSweeper, with its anti-virus program, may create a bottleneck with high CPU utilization figures. HTTP is, in most cases, used interactively; so, users will soon experience significant delays during Web surfing.

A possible solution for this situation (besides maximizing CPU and disk performance on the servers) is discussed in the following.

Simplifying the tasks of the content screening servers include:

- Reducing the logging levels
- Relieving the WEBSweeper host from caching
- Removing other applications running on critical machines

Another means to improve overall throughput is to scale up with multiple systems as shown in Figure 55. An efficient arrangement requires a caching proxy nearest to the client hosts. Because SurfinGate's per user or per department security policy cannot be enforced once an item is in the cache, clients with equal security policies should use the same caching proxy. A solution is to use different caching proxies for different groups of clients. It might even be possible to run all these caching proxies on only one machine, for example, an IBM Web Traffic Express (WTE), listening on different ports or addresses. With appropriate configuration of the proxies, the clients can be prevented from accessing an incorrect daemon.

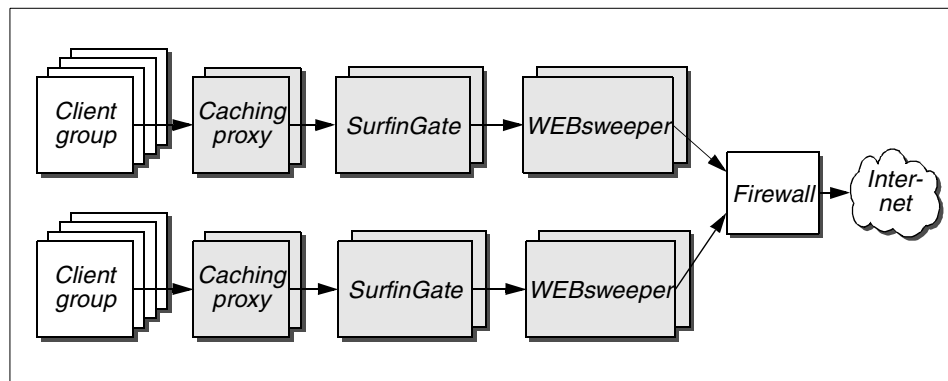


Figure 55. High-volume HTTP traffic configuration

The caching proxies are arranged to access different SurfinGate servers for distributing the load. The SurfinGate servers, in their turn, can access different WEBSweeper servers. The WEBSweeper servers should pass the firewall with static NAT to maximize throughput.

Note

When using NAT to pass FTP through the firewall, special care must be taken to face attacks over the FTP data opening in the IP filters. See the next section 5.5.3, "FTP traffic" on page 147.

5.5.3 FTP traffic

In this section, we handle usage of the real FTP protocol (for FTP via HTTP, see section 5.5.2, "HTTP traffic" on page 144). The following discussion is for *outbound* FTP sessions, that is, FTP sessions initiated on clients that are on the secured network connecting to servers on the non-secure network. This has nothing to do with the direction of the actual data transfers.

For FTP, it is not possible to use WEBSweeper's features for URL rating or content screening. Therefore, if you deploy WEBSweeper *and* enable outbound FTP traffic, you should keep in mind that WEBSweeper's screening features can easily be circumvented by the users. In other words, the security of WEBSweeper is questionable if FTP is allowed at the same time. Therefore, as a rule of thumb, FTP should not be allowed when WEBSweeper is being deployed to protect the boundary, or it should be allowed only for selected servers and trusted users.

When using FTP, the main security concern is protection from misuse of the FTP data connection by intruders. FTP has two different modes of operations, which are explained briefly before moving on.

Normal mode FTP works with a control connection from a port ≥ 1024 on the client to port 21 on the FTP server for authentication and all FTP commands. A second data connection is established from port 20 on the server machine back to a port ≥ 1024 on the client host.

Passive mode FTP has the same control connection; the data connection is built from a port ≥ 1024 on the client host to a port ≥ 1024 on the FTP server.

If IP filters are used on the firewall for protection, and normal mode FTP is allowed from the secure intranet to the external Internet, then everybody on the Internet can request connections from port 20 on their hosts to any port ≥ 1024 on machines on the intranet. This is a potential working point for attackers.

With the means of the IBM Firewall, there are the following ways to handle this problem:

- Use passive mode FTP and do not open the filters for normal mode data connections. This requires FTP clients capable of passive mode. There exist a couple of them, but the `ftp` program delivered with most operating systems usually does not belong to them.
- Dedicated client machines with no services running might pass the firewall with *dynamic* (many-to-one mapping) NAT using normal mode FTP.
- Build the final stage of the FTP connection to the Internet from the firewall host and avoid or protect all services on the firewall on ports ≥ 1024 . A couple of mechanisms exist on the firewall, each of which has some disadvantage:
 - SOCKS requires a socks-enabled client program.
 - FTP proxy requires an intermediate authentication on the firewall.
 - Transparent FTP proxy requires a modified authentication on the firewall, thus, replacing the authentication on the FTP server.

Which mechanism should be selected depends on factors, such as available client software and possible education of the users.

5.5.4 HTTPS traffic

The content screening features available for HTTP traffic analysis cannot work with HTTPS because it provides a client to server encryption. Nevertheless, SurfinGate and secure HTTP proxy can forward HTTPS traffic. For performance reasons, however, it might be preferable to pass HTTPS traffic directly from the clients to the firewall secure HTTP proxy.

5.5.5 Domain Name Service

The SecureWay Boundary Server has to provide access to the Internet domain name space for internal hosts or at least an internal name server. DNS uses UDP, but because it is unsafe to pass UDP traffic through the firewall, the use of an application gateway is required.

The DNS server of AIX and Windows NT can be configured as a caching-only name server that forwards internal requests to the appropriate Internet name servers or to the ISP's name server. If an attacker uses the related IP filter opening to access the firewall name server, he or she can only inquire what this name server knows about, and this is only public Internet DNS records.

For more details on DNS and its setup, please refer to Appendix C, "Basic firewall and name service design" on page 327.

5.6 Scalability

With many users, and a fast Internet link, the hosts of the SecureWay Boundary Server might get overloaded. Many of its components can be installed on multiple hosts.

For HTTP traffic, we have already provided a static solution for all components in the previous sections except for the firewall. The multi-path approach for SurfinGate and WEBSweeper can be extended to use multiple firewalls statically. The firewalls could use HACMP for high availability.

To utilize load balancing, the SecureWay Network Dispatcher can be used. It is supported on AIX and Windows NT. The particular architecture will depend on where you identify your bandwidth critical pathes.

For a complete discussion and instructions how to installation firewalls using network dispatcher or HACMP, see the IBM Redbook *Highly Available IBM eNetwork Firewall Using HACMP or eNetwork Dispatcher*, SG24-5136.

5.7 Interoperability with other FirstSecure components

The SecureWay Boundary Server is a core element likely to be deployed in most SecureWay FirstSecure installations. It is responsible for permitting or denying any traffic between secure and non-secure networks. So, in addition to the “ordinary” data traffic and the connections required for the SBS’s administration, it must be configured to allow all required traffic between the diverse FirstSecure components on the different networks.

The IBM Firewall supports authentication as directed by the FirstSecure Policy Director. Authentication information for firewall proxy users and groups stored in an LDAP directory by the IBM Policy Director can be accessed and used by the IBM Firewall by installing the SecureWay Boundary Server on the firewall machine. Group information for SurfinGate are stored in LDAP as well.

As mentioned earlier, Telnet, FTP, SOCKS, and HTTP proxy users can be administered either locally or by means of the Policy Director Management Console. In the latter case, the IBM Firewall extracts the relevant user information from the Policy Director’s LDAP User Registry. See 10.1.4, “Managing IBM Firewall proxy users” on page 283 for more details on this.

Chapter 6. Intrusion Immunity (II)

In Chapter 2, “The systematic approach to managing security” on page 11, and Chapter 3, “The IBM SecureWay FirstSecure building blocks” on page 41, we have seen the principles of intrusion immunity, how it can be applied to help protect your computing environment, and which products come with the IBM SecureWay FirstSecure framework. This chapter takes a closer look at the concepts and technics that *Tivoli Cross-Site for Security* and *Norton AntiVirus Suite* use.

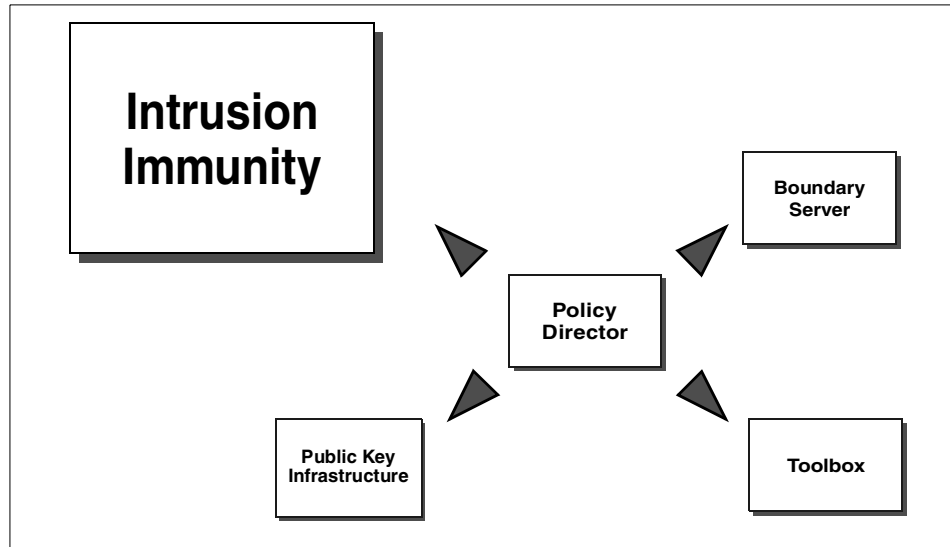


Figure 56. *Intrusion Immunity*

As depicted in Figure 56, Intrusion Immunity (II) is one of the five major building blocks that constitute the foundation for the IBM SecureWay FirstSecure framework.

6.1 More about intrusion

There are different ways an organization’s security might be compromised. ID covers (detects) security events directly affecting the local computing environment, which might be an organization internal (secure) intranet and the DMZ.

Let us have a look at what attackers could do to compromise security. The following list represents some typical scenarios; it is far from being comprehensive or even a survey:

- They might gather information about your networks and computers:
 - Who's database (publicly available) contains information about network addresses, administrators, and so on.
 - DNS (domain name service) and reverse DNS (in parts publicly available) give information about machine names and their addresses.
- They might try to find more information through:
 - Network scans which result in a list of directly available machines.
 - Port scans, which result in lists of open TCP and UDP ports on the particular machines.
 - Operating system detection, which helps to guess the operating system that is running on the particular machines.
 - Protocol detection of found ports, which helps to guess what kind of service is available and, sometimes, what software program in which version is running.
- They might try to (mis)use available services to get more information. Some simple examples include:
 - Invoke internal DNS servers
 - Download password files
 - Guess user IDs with SMTP (Simple Mail Transfer Protocol) features
- They might try to bypass your security measures:
 - Spoof their IP addresses to trusted ones
 - Use source ports that may open reverse connections (FTP data port)
- They might try to get unauthorized access to systems:
 - By using available services, for example, Telnet, with a guessed password
 - By misusing misconfigured servers, for example, by packing a shell command into CGI (Common Gateway Interface) parameters
 - By exploiting bugs, typically feeding executable code into buffer overflows
- They might also want to bother you with denial of service attacks:
 - Overload your servers with valid requests (mail bombing)
 - Overload your servers with invalid requests (SYN attack)
 - Try to crash your machines with malformed IP packets (fragmentation attacks)

Obviously, a real intrusion attempt may, depending on the hackers' motivation, differ from the scenarios listed above. Therefore, the intruders might use denial of service attacks to disable security measures. After partial successes (if they are not content with exchanging the Web server's home page), they might misuse the confidence that other machines give in the "conquered" host, or they might repeat parts of the above procedures. Or, attackers might launch some attack forms that are bound to the local network, such as ARP based attacks, eavesdropping, or session stealing.

There are many other forms of attacks to be taken into consideration, such as routing attacks, DNS spoofing and malicious mobile code, such as trojan horses and viruses, as well as technics to crack more machines on the local network after getting control over one.

An interesting source for information on security is the CERT Coordination Center collection of security advisories, vulnerabilities, and technical tips available on their Web site at:

<http://www.cert.org>

ID constitutes the methods to detect attacks on the network. In the next sections, the two components of the IBM FirstSecure Intrusion Immunity building block, the network intrusion detection system *Tivoli Cross-Site for Security* and the *Norton AntiVirus Suite*, are described.

6.2 Tivoli Cross-Site for Security

Tivoli Cross-Site (XSite) for Security is a network intrusion detection system that can detect attacks by monitoring network traffic. Due to its three-stage, scalable architecture consisting of agents, server, and consoles, it can be applied to small, as well as large and complex, network environments.

Before the concept and operation of Tivoli Cross-Site is explained, some terms used in the context of Tivoli Cross-Site need to be defined and explained.

In this context, *user* means administrative users that have access to the intrusion detection system.

A *resource* is an item that can be managed including Cross-Site agents, users, and collections.

A *collection* is a group of resources that common policies may be applied to.

A *policy* specifies how a resource behaves by setting up restrictions or guidelines.

A *domain* is a complete Cross Site for Security system, comprised of consoles, agents, one server and its repository, working together.

Events are alerts, error messages, or status messages.

Tasks are actions run on resources with parameters set by policy.

The following sections take a closer look at the different components of Cross-Site for Security, and how they work together.

6.2.1 Security agent

Security agents are the functional parts of the Tivoli Cross-Site network intrusion system. They are installed on strategic points on the network and monitor the traffic. Security agents act like smart IP packet sniffers that catch all packets on the network, check them for abnormalities on different network layers, and keep status of established connections and statistics. They watch the data flowing in established connections and can detect if suspicious information inquiries are sent or some text patterns indicating typical login or buffer overflow attacks are passed.

Security agents communicate only with the Cross-Site server, not directly with the consoles. All administrative connections are initiated by the agent using HTTP or HTTPS protocols.

The agents connect to the Cross-Site server under two conditions:

- After regular time intervals, the agents connect to the server by issuing a *heartbeat*. The server expects this heartbeat; if the server does not receive an agent's connection in time, it releases an "Agent not responding" event. If the connection succeeds, the client uploads the actual log entries and eventually updates its configuration from the server.
- When an agent detects an incident that is rated *critical*, it immediately connects to the server to notify it.

The Cross-Site security agent program is available for AIX, Sun Solaris, and Windows NT. Under Windows NT, it runs as a service; under AIX and Solaris, it runs as a daemon. Currently supported networks are Ethernet and Token-Ring.

The security agent can detect the following types of attacks:

Denial of service attacks on the IP level – Detecting fragmentation attacks, ping of death, and flood attacks is hard coded in the agent. Configuration can enable it and set parameters, such as time intervals.

ICMP redirect - Support is coded in the agent.

Port scans - TCP and UDP port scans, Network scans.

Specific Services – Support for detection of some kinds of suspicious activity is prepared in the agent code for DNS, portmapper, NFS, rstatd, NIS, and SMB.

Default Port Services – Configuration for detection of suspicious text patterns is already contained in the default policy file for the following services: telnet, r-services (rlogin, rsh, rexec), FTP, finger, TFTP, WWW (HTTP), SMTP, X-Windows, gopher, IMAP, POP, ident, NNTP, IRC, LPR, talk, UUCP, Kerberos, and writesrv.

Generic Port Services – Flow of user defined suspicious text strings on specified ports.

The trigger of alerts can be configured depending on IP packet source, destination, time of day, or the number of occurrences per time. Detection of one of the configured intrusion patterns on the network is an *incident*. The security policy defines priorities for the different possible incidents. The highest priority, denoted as “1”, means that the agent contacts the Management Server immediately. Lower priority incidents are logged to a file at the agent and uploaded to the server during scheduled heartbeat uploads.

6.2.2 Cross-Site server (Management Server)

Most information on a Cross-Site for Security system, including configuration and logs, is stored in the *management repository*, based on a DB2 or Oracle database. An IBM DB2 or Oracle RDBMS is a prerequisite for the Cross-Site server.

Communication between server and agents, and between console and server, is never initiated by the server. A *Netscape Enterprise Server*, that must be installed on the Cross-Site server machine, listens for HTTP or HTTPS client connections from agents and consoles. This HTTP server communicates with the Cross-Site server with the help of a servlet. Agents must be registered on the server before their requests are handled accordingly.

The server is the management and configuration center. All administration tasks and more common configurations are controlled via Cross-Site Management Consoles. Only the generation or modification of Cross-Site security policies requires text file editing and text commands.

Events can be escalated through e-mail, SNMP traps, or forwarded to a Tivoli Enterprise Console (TEC).

The Cross-Site server is available for AIX, Sun Solaris, and Windows NT. Under Windows NT, it runs as a service; under AIX and Solaris, it runs as a daemon.

6.2.3 Management console

The Tivoli Cross-Site Management Console is a graphical user interface for administration and control of the Cross-Site server and, indirectly, of the agents. Typically, it will be installed on the server machine and on additional administrators' workstations.

For communication with the server, the consoles connect via HTTPS. Authentication requires an administrative Cross-Site user name and a password. The Cross-Site console is available for AIX, Solaris, and Windows 95/98/NT.

6.2.4 Configuration considerations

How can Cross-Site for Security be applied to a networking environment to get the best protection? This section looks at how the components are best positioned and how to combine them. Two practical scenarios are then discussed in subsequent sections.

6.2.4.1 Positioning security agents

The ability of the Tivoli Cross-Site for Security agents to monitor all critical network traffic is crucial for the efficiency of this intrusion detection system. Another important factor is the availability and security of the server and the interconnections of the agents to the server.

The following conditions influence the decision as to where to put security agents.

The agent can reside on a *dedicated host* in the network segment if this machine can see all the IP traffic on the network (unswitched network connection or host on mirroring switch port and adapter in promiscuous mode) and if it is a UNIX machine. The host where the security agent is

installed may be dedicated to the purpose of monitoring traffic but might as well perform additional functions.

The agent can run on any *single host* to protect this very machine if the agent is supported on this platform. In this case, the network can be switched or unswitched, and the network adapter does not need to support a promiscuous mode. No additional hardware is required if this method is deployed. A disadvantage of this solution is that a successful intruder can change the policy of the agent to hide further activities.

The agent must be able to connect to the Cross-Site server with HTTP or HTTPS.

When traffic between agent and server crosses a non-secure network, HTTPS should be used instead of HTTP.

6.2.4.2 Positioning the Cross-Site server

When arranging the position of the Cross-Site server within a network environment, the following conditions should be met:

The Cross-Site server must be in a secure location. The whole intrusion detection system breaks if the security of the server is compromised.

The agents must be able to connect to the server via HTTP or HTTPS, as well as the consoles must be able to connect to the server via HTTP or HTTPS.

Due to basic security considerations, it is wise not to put the Cross-Site server on a network together with Internet servers. A possible solution is to put the Cross-Site server into the internal network and have the agent(s) in the DMZ connect to the server with HTTPS over a generic proxy on the firewall.

6.2.5 Deploying Cross-Site for Security - Scenario 1

This first example scenario represents a simple installation.

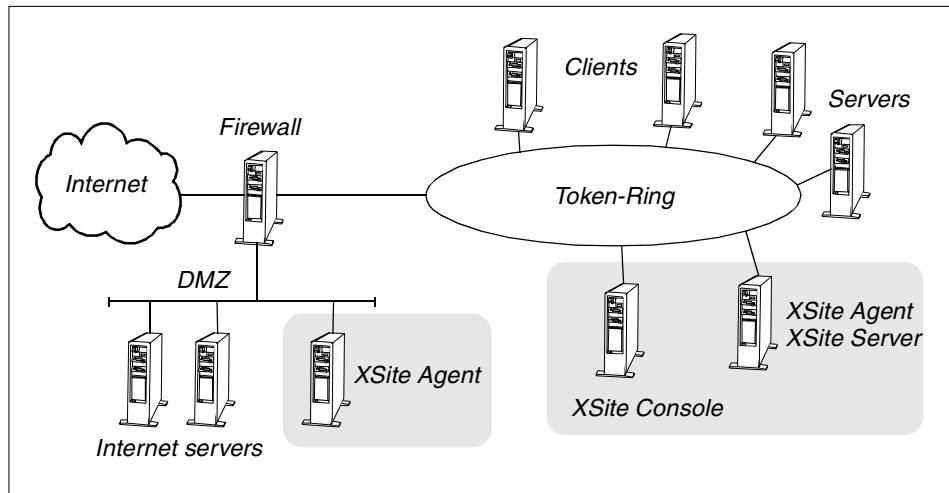


Figure 57. Cross-Site for Security - Scenario 1

The DMZ in this scenario (Figure 57) is a network that connects to a third network interface on the firewall machine. It is implemented as a simple Ethernet segment; so, one agent is sufficient to protect all Internet servers there. It connects to the Cross-Site server with HTTPS through the firewall.

The intranet in this example is an unswitched Token-Ring with no bridges. It has some servers and clients. Only one Cross-Site agent is required to monitor this network. It resides on a machine used at the same time as the Cross-Site server. The agents on the DMZ connect through the firewall to the server.

The administrator PCs are equipped with Cross-Site consoles.

6.2.6 Deploying Cross-Site for Security - Scenario 2

This second sample scenario represents a more complex environment that shows the principles for even more complex networks.

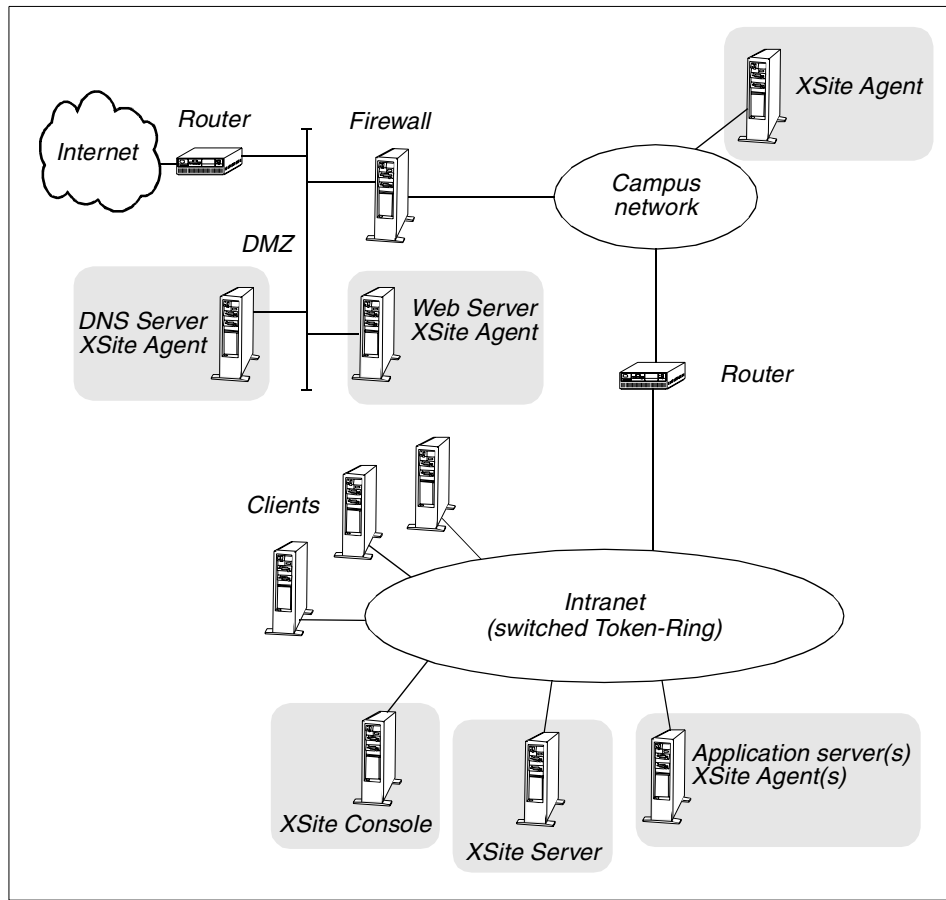


Figure 58. Cross-Site for Security - Scenario 2

The DMZ in this second sample scenario (Figure 58) is implemented between a filtering router and the firewall with a switched Ethernet to prevent eavesdropping. The Web server and the DNS server both have a Cross-Site agent installed for individual detection. This is necessary because a separate Cross-Site agent would not work due to the inherent characteristics of a switched network.

The intranet in this example is a switched Token-Ring network. For the same reason as mentioned above, a comprehensive network traffic monitoring is not possible. Because of this, critical servers need to have a Cross-Site agent installed in order to be monitored. To monitor at least the traffic between the Internet and the secured intranet, an additional network segment between the firewall and the internal switched network can be inserted, or an existing

campus network can be used for this purpose. In either case, that network must not be switched, or special routing provisions must be taken. A Cross-Site agent monitors all traffic there on that network. The Cross-Site server and consoles are on the internal network.

Because of the nature of a switched network (Token-Ring or Ethernet), a comprehensive monitoring that involves all servers and clients is difficult to implement and requires either special network equipment set up or a reasonable number of Cross-Site agents.

6.2.7 Configuring Tivoli Cross-Site for Security

Security agents get their configuration from the Management Server. The source of the agents' policy configurations are the files `ids.cfg`, `ids.msg`, and `ids.rules` that are maintained on the Management Server. The Tivoli Cross-Site for Security product comes with a rich set of default configurations based on research and learned from actual attacks. These defaults can be used and serve most environments to a large extent.

The `ids.rules` file on the Cross-Site server is a plain text file for configuring what IP addresses and ports should be monitored, what text or byte patterns should be detected, and what specific services, such as DNS, NFS, or SMB should be observed. For each such setting, a priority and message ID may be specified.

The `ids.rules` file consists of sections. The beginning of a section is marked by a `PORTS` line, the end of a section is marked by an `END` line. There exist statements for *generic port services*, where the intrusion signatures may be customized, and statements for *default port services*, where at least some of the intrusion signatures are hard-coded in the agent.

The following is an excerpt from an `ids.rules` file, shown here for illustration purposes:

```
#
##### generic port services, customized to watch FTP #####
# this section applies to the FTP port
PORTS 21
# monitor TCP with authentication features, in both directions
PTYPE TCP AUTH SRC DST
# Service ACL
PROMPT USER      "USER"
PROMPT PASSWD    "PASS"
PROMPT AUTHFAIL  "ogin incorrect"

# Authentication Section
```

```

AUTH USER "lp"      NOTIFY    ANY          3044    1
AUTH USER "sync"    NOTIFY    ANY          3045    1
AUTH USER "demos"   NOTIFY    ANY          3046    1
AUTH USER "ftp"     NOTIFY    ANY          3043    1
AUTH USER "anonymous" NOTIFY    ANY          3043    1
AUTH USER "root"    NOTIFY    OFFPEAK      3042    2
AUTH USER "guest"   NOTIFY    ANY          3043    2
#
# Signature List
#
SIG DST "[Ee][Xx][Ee][Cc]" 3047    2
SIG DST "passwd"         3052    2
SIG DST "MKD"           3053    2
SIG DST "[Pp][Aa][Ss][Vv]" 3054    2
SIG DST ".rhosts"       3055    2
SIG DST "hosts.equiv"   3056    2
SIG DST "../.."         3057    2
SIG SRC "uest login"    3048    3
SIG DST "CWD ~root"     3049    3
SIG DST "SYST"          3050    3
SIG DST "SITE"          3051    3
END
#
##### DNS default port service section #####
DNS
# issue message 766 with priority 2 if someone makes zone transfer
REQ ZONEXFER NOTIFY ANY          766    2
# alert if someone sends "/bin/sh" or similar
REQ SIG "in/" NOTIFY ANY          769    1
END

```

The ids.cfg file on the Management Server contains some parameters specifying times and number of packets possibly indicating scan and flood attacks.

The following excerpt from a default ids.cfg file illustrates its purpose:

```

# peak times from 9am to 5pm (seconds from midnight), Mo..Fr
PEAK          25200    64800    1 2 3 4 5
# LOWEST port to check for scans
lowscanport 1
# HIGHEST port to check for scans
highscanport 65530
# Sample Time for Fast window in seconds
epochtime    10
# Single hosts send # SYNS to port in 'epochtime' seconds - issue
# SYN flood alert

```

A pair of related `ids.rules` and `ids.cfg` files comprise a *policy*.

Another important configuration file is `ids.msg`. It contains the full-text messages for all event types of all policies.

Each security agent is configured with one policy. Different security agents may be assigned the same or different security policies. Whenever the files of a policy have been modified, the Management Server must be notified. The server reads in the new policy files, stores their contents in the database, and waits for the heartbeat of the appropriate agents. The server then informs the agents about the policy change, and the agents download and activate their new configuration.

A comprehensive and meaningful default policy already exists in the Tivoli Cross-Site package. New policies can be created on the Management Server with a command starting from the default or another existing policy. A new folder is created, and the files of the existing policy are copied there.

Most configuration and administration tasks, except creating or updating policies, can be controlled with the Tivoli Cross-Site console. Using the console's GUI, an administrator can view event entries, generate reports about security related events, assign policies to agents or collections, manage tasks, and perform some purely administrative actions, such as managing administrative Cross-Site users.

For more information about Tivoli Cross-Site for Security installation and configuration, see 9.4, "Intrusion Immunity" on page 252.

6.3 Norton AntiVirus Suite

The Norton AntiVirus Suite (NAV) is a comprehensive collection of anti-virus tools. It provides programs for checking Internet data traffic, for protecting servers and workstations, for central management of the anti-virus programs on many hosts, and for regularly updating the virus signature database.

The programs comprised in the FirstSecure framework are:

Desktop solutions:

- Norton AntiVirus for Windows 95/98
- Norton AntiVirus for Windows NT 4.0

Server solutions:

- Norton AntiVirus for Windows NT 4.0
- Norton AntiVirus for Lotus Notes

Gateway solutions:

- Norton AntiVirus for Internet e-mail Gateways
- Norton AntiVirus for Firewalls

Administration:

- Norton System Center
- Other administration tools, including Norton AntiVirus Network Manager

6.3.1 Norton AntiVirus for Windows 95/98 and Windows NT

These programs are designed for use on workstations. They protect computer memory and disks from computer viruses.

The Norton AntiVirus Auto-Protect component is residually installed in the computer's memory after booting up that computer. It monitors system activity and alerts the user if suspicious action are observed.

When files are downloaded from the Internet, or when packed zip files are decompressed, the resulting program files are automatically checked for viruses. And, the user can directly invoke NAV at any time to check computer memory, files, or disks for viruses. When a virus is detected, the infected program may be quarantined or, if possible, repaired.

On Windows 95/98, after NAV installation, change of hardware, disk partitions, and update of virus protection or operating system, a set of rescue floppy disks should be generated to help repair the computer after serious kinds of virus infections.

To keep up-to-date with the signature database, the user can download an actual version with the LiveUpdate feature. This update process can also be scheduled to run at given intervals. LiveUpdate connects via FTP to a host at Symantec Corp. to download.

If the user suspects a file to be infected with a virus that NAV does not know, he or she can send it to the SARC (Symantec AntiVirus Research Center) for analysis. If the file contains a yet unknown virus, Symantec creates related entries to the virus protection database and makes an updated version available.

6.3.2 Norton AntiVirus for Windows NT Server

This product is the NAV version for Windows NT servers. It provides some features of the workstation products, such as Auto-Protect, manual and scheduled scans, and LiveUpdate.

Norton AntiVirus for Windows NT Server uses two sophisticated methods for detecting unknown viruses and polymorphic viruses:

Bloodhound heuristic technology performs code analysis to find out if a program contains sequences that give it viral behavior. A modified algorithm is used to detect unknown macro viruses.

Strike technology is applied to clean program files infected by polymorphic viruses. The decryption part of virus spreading is performed in a virtual machine that would not allow the macro to spread or do something harmful. After self-decryption of the virus, it can be removed from the infected file.

Note

Polymorphic viruses can have different appearance in each infected file by encrypting itself with random keys. Therefore, they bypass classical byte pattern based anti-virus checks. When getting activated, they decrypt themselves and get back their executable form.

The full power of NAV for Windows NT servers is achieved when using it together with *Norton System Center* or *Network Manager for Windows*. It helps to manage network-wide anti-virus operations. Especially roll-out to remote servers, central LiveUpdate control, and quarantine management are helpful while Norton Event Manager enables a central alerting system.

6.4 Interoperability with other FirstSecure components

Intrusion Immunity (II) integrates with other components of the IBM SecureWay FirstSecure framework in the following ways:

- II provides an additional level of security for all kinds of security-critical services and networks.
- Norton AntiVirus checks not only critical FirstSecure servers, but also administrators' workstations and even other, non-related workstations.
- II, through the use of the HTTP/HTTPS protocol, can easily be configured to operate across boundary servers (firewalls).

- Trust Authority (the PKI building block of IBM SecureWay FirstSecure) can generate the digital certificates used for mutual authentication amongst the Cross-Site for Security systems.

Chapter 7. Public Key Infrastructure (PKI)

IBM SecureWay FirstSecure's firewall and Web applications security technologies can take advantage of the strong authentication offered by digital certificates based on public key technology. In addition, many applications today involve the use of smart cards, SSL authentication, virtual private networking (VPN), and other implementations. These require the issuing and management of digital certificates and the keys they are based on.

Many organizations are deploying or piloting a Public Key Infrastructure (PKI) to help provide the trust they need. But, organizations are concerned about the lack of standards and interoperability. They are also concerned about the difficulty of managing a PKI in-house, leading many to choose to outsource instead. Many deployers are waiting for trusted third parties to build the necessary trust infrastructures to spur trusted business-critical transactions.

FirstSecure provides an answer. It provides these functions through its fourth building block: The IBM SecureWay FirstSecure Public Key Infrastructure for X.509 V3 (Figure 59).

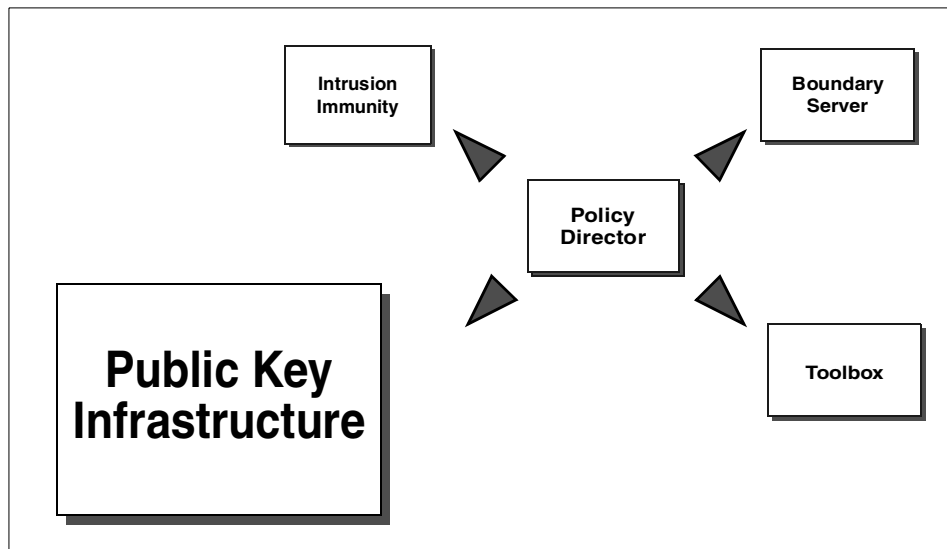


Figure 59. Public Key Infrastructure

The FirstSecure PKI offers a high trust, flexible registration and certification solution that adheres to the Common Data Security Architecture (CDSA) and

Public Key Infrastructure for X.509 certificates (PKIX) standards and support Web applications, VPN, secure e-mail and custom applications. It supports PKIX as well as other prevalent certificate request and distribution methods, such as PKCS, native browser, and e-mail.

PKI is built on IBM SecureWay Trust Authority 3.1, thus, helping organizations to quickly get started implementing and using a PKI.

The Trust Authority's main function is to provide the use of certificates for authentication, secured communications, and validation of signed policy (Figure 60).

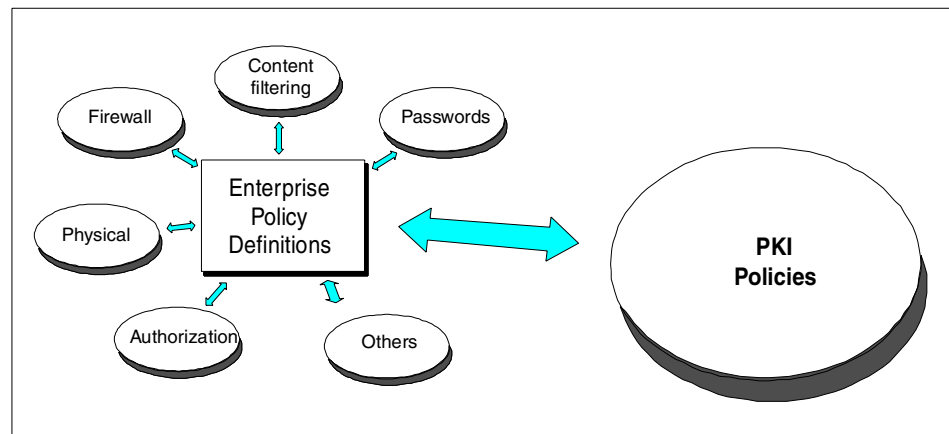


Figure 60. PKI policies

Trust Authority focuses on providing a simple way to deliver certificates to end users, thus, encouraging widespread use of certificates. This includes operating systems, Web servers on various platforms, key middleware products that form the basis for many e-business, end-to-end flows (for example, CICS, MQSeries, IMS, DB2) and other FirstSecure components, such as Policy Director, Service Boundary Server, and Intrusion Immunity components (see 7.5, "Interoperability with other FirstSecure components" on page 188 for more information about the interoperability among FirstSecure components).

FirstSecure PKI solution also provides a PKI application programming interface (API) as part of the Toolbox component (see 8.3.1, "PKIX API" on page 203 for more information). This API is a collection of certificate services (generate, revoke, sign, and verify) that provides, to developers, basic PKI capabilities. These capabilities include Certificate Authority (CA) and

Registration Authority (RA) functions, certificate life cycle management, and enablement for interface with PKI in FirstSecure.

7.1 PKI for X.509 V3 architecture

The FirstSecure PKI for X.509 V3 architecture is based on the Internet Engineering Task Force's (IETF) Public Key Infrastructure (PKI) working group specifications to ensure interoperability with the PKI offerings from other vendors. The implementation of the IETF specifications is an X.509 based Public Key Infrastructure (PKIX) architecture, also known as *Jonah implementation standard*. This standard was jointly developed by IBM, Lotus, and Iris, and is publicly available on the WEB at:

<http://www.mit.edu/pf1>

The Jonah implementation is a set of proposals that provide a well-rounded definition of an interoperable PKI.

The task of the working group was to develop Internet standards needed to support an X.509-based PKI. The goal of this PKI was to facilitate the use of X.509 certificates in multiple applications, which make use of the Internet and to promote interoperability between different implementations choosing to make use of X.509 certificates. The resulting PKI is intended to provide a framework that supports a range of trust/hierarchy environments and a range of usage environments.

The difference between PKIX and existing PKI implementations is that PKIX is a comprehensive, open solution that encompasses all aspects of a Public Key Infrastructure. PKIX constrains the broad X.509 standard, therefore, giving up some flexibility in order to gain interoperability. PKIX includes digital signature standards, certificate and CRL formats, certificate and CRL retrieval protocols, certificate management protocols, directory services, and key management. The advantages of PKIX are its openness and interoperability.

The management of certificates (containing public keys) for widely-distributed users or systems requires a public-key *infrastructure*. The X.509 standard constitutes a widely-accepted basis for such an infrastructure, defining data formats and procedures related to distribution of public keys via certificates digitally signed by Certificate Authorities (CA). Before the Jonah implementation, RFC 1422 (available, for example, at <http://www.ietf.org>) specified the basis of an X.509-based PKI targeted primarily at satisfying the needs of Internet Privacy Enhanced Mail (PEM). Since RFC 1422 was issued, application requirements for an Internet PKI have broadened

tremendously, and the capabilities of X.509 have advanced with the development of standards defining the X.509 Version 3 certificate and Version 2 Certificate Revocation Lists (CRL).

The implementation focuses on tailoring and profiling the features available in the X.509v3 certificate to best match the requirements and characteristics of the Internet environment.

The implementation also specifies options for CA-to-CA certification links and structures, revocation alternatives, certificate and CRL distribution options, policy definition and registration, generation of key pairs, administrative protocols and procedures, including certificate generation, revocation notification, cross-certification, and key-pair updating.

Figure 61 diagrams the Jonah function.

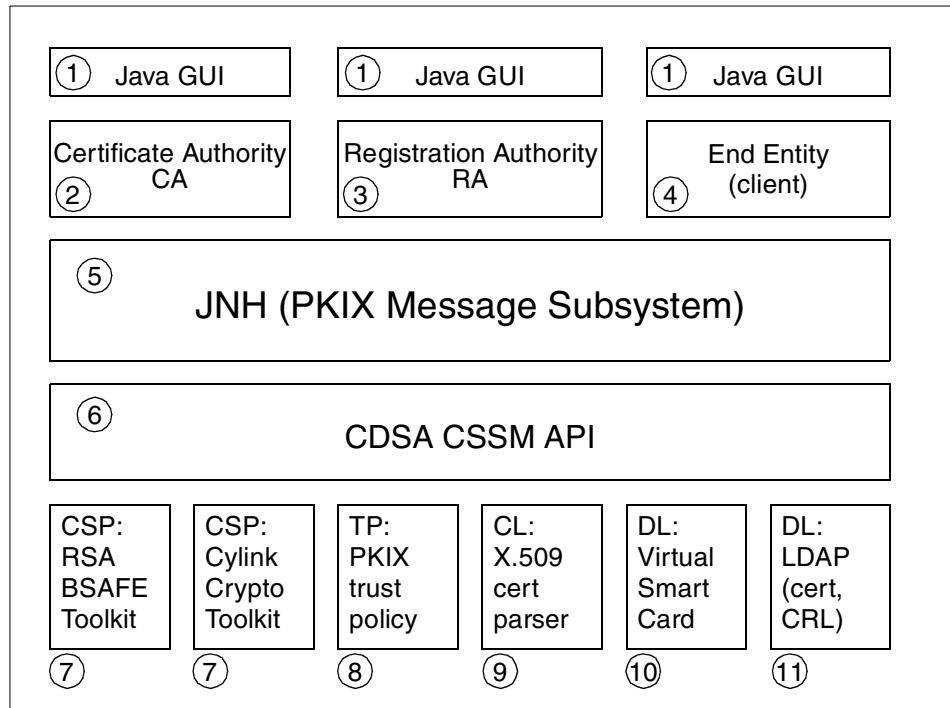


Figure 61. The overall Jonah architecture

The following are descriptions and explanations of the numbered items shown in Figure 61:

1. *Java GUI* – Java graphical user interface to provide portability across platforms.
2. *Certificate Authority (CA)* – The CA creates and signs certificates, maintains Certificate Revocation Lists (CRL), and provides an administrator interface for managing certificates and CRLs (authorize or refuse certificate and CRL operation). The CA can accommodate hierarchical models.
3. *Registration Authority (RA)* – The RA is an optional component that allows the CA to be operated as an off-line process, typically used by a person who is present to authorize operations. It acts as an interface between the clients and the CA. The RA acts as an agent for the CA by buffering management requests and responses, performing cryptographic operations, such as key generation or key archival on behalf of the client, authentication and issuing names to prospective PKI users, token distribution to client's, and archiving keys as required by CA policy or law. Splitting the CA and RA capability may be considered an additional security feature and a performance enhancement. The RA is designed to run either co-resident with the CA or on a separate machine. The RA has a user interface for monitoring CA operations and for controlling audit and archive settings. The RA accepts the PKCS #10 or PKIX certificate request format. Mail or TCP/IP protocols and diskettes may be used to transmit requests to the CA.
4. *Client* – The client interface is used to enroll with the PKI or whenever a change must be made to the user's personal security profile. Through this client interface, the user issues requests for certificate generation, revocation, certificate lookup, certificate posting, and maintains the user's personal security profile. For existing PKI-aware applications, the client offers an interface for selected applications (for example, Netscape Navigator and Internet Explorer) to load PKIX certificates into these applications. The certificate export functionality is extensible, thus, allowing support for additional applications to easily be added.
5. *JNH (PKIX Message Subsystem)* – The high-level API that interfaces with the certificate management.
6. *CDSA CSSM API* – CDSA is the Common Data Security Architecture that provides a software security framework consisting of APIs designed to enhance security for applications. The Common Security Services Manager (CSSM) is a layer under CDSA that defines a certificate-based API for security services and manages add-in security modules that provide services to applications.

7. CSP – *Crypto Service Providers* are add-in modules that perform cryptographic operations, such as encryption, decryption, digital signing, and so on.
8. TP – *PKIX Trust Policy* module that implements policies (rules or business practices). Examples of policies may be: Who can issue a certificate? If a certificate is issued by another CA, will another CA trust it?
9. CL – *Client PKI* access library is a directory and other repository for certificates and CRLs. The PKI library offers application developers a way to access the PKI. There is no user interface component. The current Jonah implementation uses an ASN.1 class library.
10. DL – *Data storage library for virtual smart card* (key storage) PKCS #11 interface.
11. DL – *Data storage library for LDAP* (Certificate and CRL storage).

PKIX does not displace Public Key Cryptography Standards (PKCS #1-11) (refer to 7.3, “Supported standards” on page 182); the two standards coexist and interoperate. The set of PKCS (see the document at: <http://www.rsa.com/rsalabs/pubs/PKCS/>) defined by RSA Data Security continue to have a significant influence on the evolution of the PKI, as they define many of the basic cryptographic services and primitives necessary to enable a working PKI.

Because PKIX is an open standard, it can use any underlying security protocols. IBM has selected the *Common Data Security Architecture* (CDSA) from Intel Corp., which has been accepted as a standard by The Open Group (<http://www.opengroup.org>) for its PKIX implementation although other protocols, such as Microsoft’s proprietary CryptoAPI, can be used. CDSA is an open, modular, and flexible standard that easily allows for different levels of security (see also 7.3.1, “PKI basic standards” on page 182).

7.2 Trust Authority 3.1

IBM SecureWay Trust Authority, as part of the IBM SecureWay FirstSecure offering, is an offering for the certification authority application markets. It is a cross platform, easy-to-use offering with simple registration and installation capabilities targeted at medium and large enterprises and ISVs with small, sophisticated operational requirements.

Trust Authority provides organizations with the ability to enroll for PKIX certificates using an *end entity client* and also handles requests for other certificate types, such as S/MIME and VPNs, through browser-based enrollments. Trust Authority supports the complete certificate life-cycle

including enrollment and initial certification, key-pair update, certificate renewal, certificate and CRL publication, and certificate revocation. It provides graphical user interface (GUI) support for product installation and configuration, Registration Authority (RA) administration, and client (end-entity) certificate administration. It also provides GUI support for changing values in configuration files and utilities for securely starting the system, changing passwords, cross-certifying CAs, checking audit log integrity, and sealing audit archive integrity. The product includes an API library for organizations who want to integrate PKI applications with their products.

The Trust Authority application incorporates a PKIX-based Certificate Authority (CA) based on the IBM Jonah PKIX reference implementation (refer to 7.1, “PKI for X.509 V3 architecture” on page 169) to ensure interoperability of PKI offerings with other vendor products. The CA includes features, such as cross-certification, certificate hierarchies, and user-defined certificate extensions. The CA leverages IBM KeyWorks for cryptographic and key store functions.

The Trust Authority offering includes software for IBM AIX and Windows NT platforms, a registration application, a CA, support for an LDAP directory, an audit subsystem, and optional support for the IBM SecureWay 4758 PCI Cryptographic Coprocessor (see also 7.2.7, “Crypto hardware support” on page 180).

7.2.1 General capabilities

The Trust Authority provides Internet applications with the means to authenticate users and ensure trusted communications. A Trust Authority system provides an organization with all the tools necessary to issue, publish, and administer digital certificates.

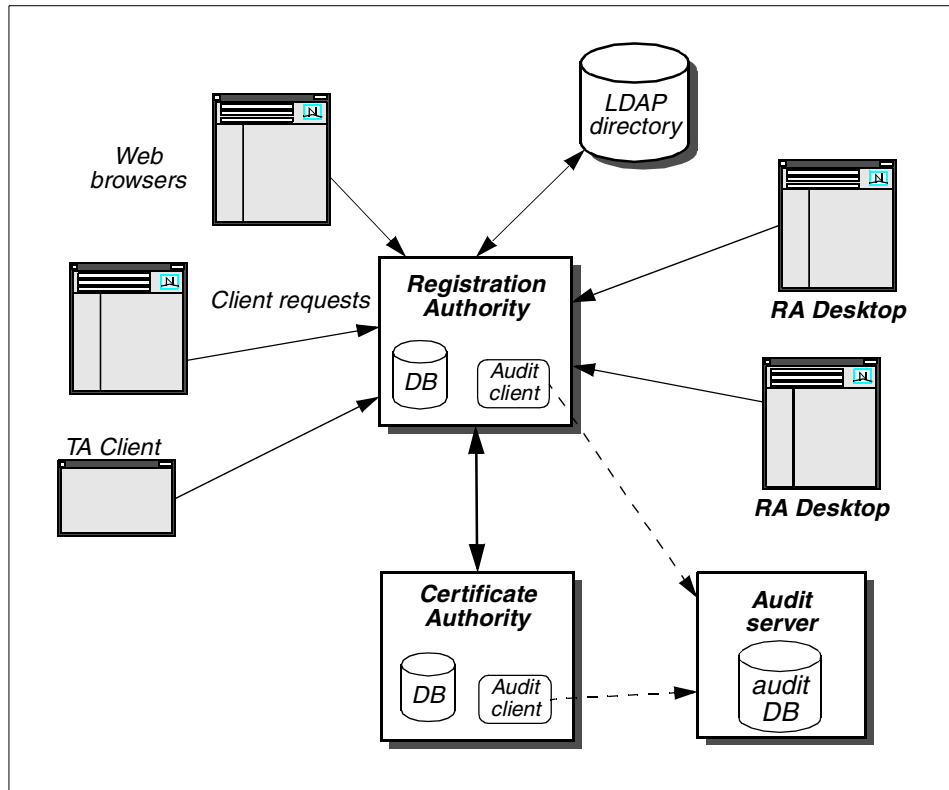


Figure 62. The Trust Authority components

The major components of Trust Authority are (Figure 62):

Registration Authority (RA) – Supports the complete life-cycle of certification, from initial enrollment (or registration) through certificate renewal and revocation. The RA allows an organization to implement and control a customized security policy that can be handled either by personnel or via automated responses. For example, an organization’s policy may require a user who needs access to sensitive servers to apply in person before the RA administrator and show two forms of identification. Only then can the administrator approve access via the RA function. In contrast to access non-sensitive servers, such as common printers, a valid e-mail address may be adequate for gaining access and might be approved automatically. This administration, which can be implemented through automated processes or human decision-making, may include the following types of tasks:

- Confirming a user's identity

- Approving or rejecting requests to obtain, renew, or revoke certificates
- Generate a valid key pair
- Validating that the user has possession of the private key associated with the public key in a certificate
- Following the rules in a given business process or certificate profile to issue particular types of certificates to particular types of users

The RA publishes information about certificates in a public key directory, the IBM SecureWay LDAP Directory (see 7.2.6, “Directory support” on page 180).

Trusted Certificate Authority (CA) – Creates and issues digital certificates. The CA publishes information about revoked certificates, allowing applications to readily ascertain the authenticity of requests. The CA can optionally use cryptographic hardware, such as the IBM SecureWay 4758 PCI Cryptographic Coprocessor (see 7.2.7, “Crypto hardware support” on page 180) or Smart Cards (see 7.2.5, “Smart Card support” on page 179), to extend its ability to protect keys. If the cryptographic hardware support is not used, the CA’s keys are encrypted and stored in the KeyWorks data storage library (DL).

The RA Desktop – A Web-based administrative interface that enables an administrator to easily approve or reject requests to obtain, renew, or revoke certificates, and to perform other administrative duties. Figure 63 shows a sample of how an RA administrator selects, lists, and works on certificates.

Web browser enrollment interface – Makes it easy to request certificates for browsers, servers, and certain devices, such as Smart Cards. Users can also use these enrollment forms to pre-register for a PKIX certificate. This form can be customized to meet the needs of an organization.

The Trusted Authority client – A stand-alone Windows interface that allows users to obtain, renew, and revoke certificates without using a Web browser.

Audit subsystem – Uses message authentication codes (MACs) to ensure that events it receives from Trust Authority CA and RA can be authenticated. A configurable option allows audit records to also be integrity-protected when they are logged.

A collection of certificate profiles – Enables an organization to issue particular types of certificates to particular types of users.

Administrative interfaces – Configures the system, approving and rejecting requests for certificates, or revoking previously approved credentials, cross-certifying CAs, integrity checking audit logs, changing secure

passwords, securely starting and stopping the system components, and modifying configuration files.

Application Programming Interface (API) – Enables application developers to write custom PKI applications (see 8.3.1, “PKIX API” on page 203 for more information).

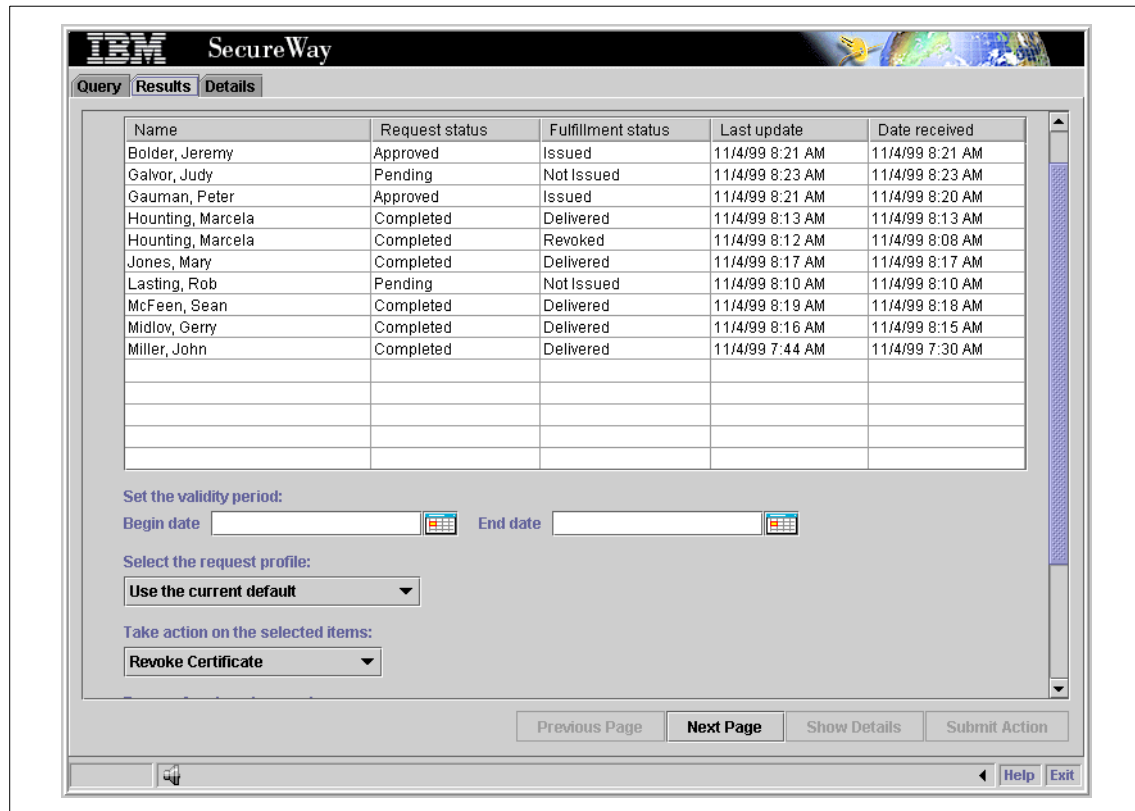


Figure 63. Sample RA Desktop view

Trust Authority also includes and supports a SecureWay LDAP Directory for the publishing of certificates and Certificate Revocation Lists (CRLs). The directory can be local or remote on another system.

The capabilities introduced above are described in more detail in the sections that follow.

7.2.2 System configurations

A Trust Authority installation manages a *security domain*. A security domain is a logical collection of users, systems, and other resources that are administered through one single CA. Among other features, Trust Authority functions as a CA.

A Trust Authority installation, managing such a single security domain, can exist on one single system, or it can be spread among multiple systems for scalability reasons. With reference to Figure 62 on page 174, the following general system configuration guidelines apply:

- The Trust Authority main server (not depicted separately in Figure 62 on page 174) and the RA server must reside on the same system.
- The CA server and the Audit server must reside on the same system.
- The SecureWay Directory server can be installed on a separate system or local on any of the Trust Authority servers.

These basic configuration principles allow the installation of Trust Authority on either one, two, or three server systems. The Trust Authority server functions are supported on Windows NT and IBM AIX.

The RA desktop (running in a Web browser on Windows), the Trust Authority client, and the client Web interface run on users' or administrators' workstations.

For maximum security, Trust Authority supports a comprehensive *internal trust model*. This includes code signing for TA code, message signing for inter-component communications, data encryption for all stored keys and other stored data, and secure KeyStores that store security-related objects.

7.2.3 Principles of operations

Trust Authority serves as, and is intended to function as, an organization's own (private) Certificate Authority. In fact, Trust Authority includes more functionality than what a CA or a certificate server normally do. The following is a description of the general operations flow in a Trust Authority environment to help you understand the functions provided by Trust Authority (TA).

When installing and configuring TA, some decisions have to be made upon the system environment (see previous section) and the policies for the security domain being created. Such policies include:

- A list of certificate types that are being issued and managed.

- Certificate lifetimes.
- How the enrollment for certificates is being conducted.
- Personnel responsibilities, such as the designation of RA administrators.
- How this security domain collaborates with other domains.

Once these decisions have been made and incorporated in the installation and configuration of TA, the typical operations flow is:

1. A user requesting a certificate fills in a form, either through the TA client application or through a Web interface provided by the RA server. The user will be given a request ID that will be required, along with an optional challenge question/response, to obtain request status information or download the certificate. When requesting a certificate, a private/public key-pair is generated on the client. The public key is then included with the request sent to the RA, while the private key remains local on the requestor's workstation. Generating a key-pair and creating the certificate request is done by either a Web browser (in conjunction with a request applet) or by the TA client application.

Note that the certificate request can be for any type of certificate, and that the form can be customized to fit the specific requirements of an organization.

2. If auto-approve is configured for the requested type of certificate, and if all the required information is correct, the request is approved automatically. If auto-approve is not configured, an RA administrator must approve (or reject) the request through the RA desktop interface.
3. Upon approval, the user can download the certificate through a Web browser or the TA client application. The user is required to enter the request ID and, optionally, a challenge response for proper authentication and authorization to download the certificate. If the requested certificate was a Web browser or an e-mail certificate, the Web browsers will automatically include them in their certificate databases (this is because TA uses standardized protocols and procedures that browsers can recognize and act upon).

In addition to issuing certificates, TA keeps local records of all activities. An RA administrator can work with these records through the RA desktop interface.

If, later on, a user's certificate needs to be revoked or put on hold, the user can simply point his or her browser to a URL on the RA server and request this certificate to be revoked. Optionally, the RA administrator can revoke certificates, too. If configured, revoked certificates can be published to

Certificate Revocation Lists (CRLs) that are stored in the LDAP directory. This is highly recommended such that the list of revoked certificates can be checked easily by applications by means of a directory lookup.

The above describes the most common operation being done with TA. Additionally, administrative operations include:

- Checking logs and audit logs
- Establishing cross-relationships with other security domains
- Maintaining the servers

7.2.4 Default certificate types

By default, Trust Authority supports the following certificate types:

- Web browser certificates, valid either one or two years
- E-mail certificates, valid either one or two years
- Signing-only certificate, valid either one or two years
- Encryption-only (key or data) certificates, valid either one or two years
- Server or device certificates, valid either one or two years, including:
 - Web server certificates
 - IPSec certificates
- CA cross-certificates

When requesting a certificate, the requestor is presented a list of available certificate types, depending on method of request (Web browser or TA client) and customization of TA.

7.2.5 Smart Card support

A *Smart Card* (or *security token*) is a device, normally of the physical dimensions of a common credit card, that houses an electronic chip for specific security operations. Such operations may include the safe storage of private keys and public certificates as well as the capability to generate private/public key pairs. Although different vendor's Smart Card products may support a different subset of possible functions, the interface to communicate with Smart Cards is well defined in the PKCS #11 standard (see 7.3.1, "PKI basic standards" on page 182).

Trust Authority supports Smart Cards. This means that certificates can be stored on smart cards that support the PKCS #11 interface on Netscape browsers.

It also allows the use of keys already resident on a Smart Card (virtual or real) when generating a new certificate request on the EE.

7.2.6 Directory support

For the storage of certificates and CRLs, Trust Authority supports the IBM SecureWay Directory. The IBM SecureWay Directory implements the LDAP directory standards, and it uses IBM DB2 Universal Database for data storage.

This directory can be pre-existing or can be configured from scratch. This means that the Directory can be an existing server, local or on a remote system, or one that is installed and configured specifically for Trust Authority. The LDAP protocol is used to access CRLs and the directory containing the digital certificates it authenticates.

The directory is used for the management, storage, retrieval, and distribution of the following types of data or objects:

- X.509v3 certificates
- CRLs
- Cross-certificates
- Other custom data types and values
- Information about registered users and servers

In addition to the LDAP directory, the IBM DB2 Universal Database (UDB) is used by the CA, RA, and audit components of Trust Authority to store data and maintain transaction logs for auditing, reporting, and recovery purposes.

7.2.7 Crypto hardware support

The IBM 4758 PCI Cryptographic Coprocessor is the Trust Authority hardware solution to protect and securely back up the CA signing key and to minimize the risk associated with administrator misuse of the system. It is an optional component of the Trust Authority product supported on the Certificate Authority server on AIX.

What is the IBM 4758 PCI Cryptographic Coprocessor?

The 4758 PCI Cryptographic Coprocessor is a programmable, tamper-responding cryptographic PCI-bus card offering high performance DES and RSA cryptographic processing. The cryptographic processes are performed within a secure enclosure on the card. The card is designed to meet the stringent requirements of the FIPS PUB 140-1 level 4 standard. Software can run within the secure enclosure. For example, credit card transactions can be processed via the SET standard.

In addition to its cryptographic intelligence, the 4758 coprocessor is able to detect attempts of physical tampering, voltage and temperature irregularities, and excess radiation. Upon detection, the keys required to access data secured in the module are erased.

Although this hardware is optional, the user is encouraged to use the IBM 4758 PCI Cryptographic Coprocessor to maximize the security of the CA's signing key. Doing so can minimize the exposure of harm from abusive system administrators or system infiltrators.

If the user decides to use the 4758 coprocessor, he or she must install it on the machine where the Trust Authority CA and Audit server software are installed. As part of configuring the CA, the user must specify whether to use the coprocessor to protect its signing key.

In most Trust Authority systems, the CA's key is not physically stored on the 4758 hardware card. Instead, the hardware's master key is used to encrypt the CA's key. This step provides an extra layer of security against attempts to compromise, or otherwise decipher, the CA's signature.

A configuration option allows the storage of the CA's key in hardware, an action that IBM discourages. The following are the risks and corrective actions associated with that decision:

- When the 4758 coprocessor is backed up, only the master key is backed up, not the CA's key or any other keys stored on the hardware card. Therefore, if an unrecoverable hardware failure occurs, you risk losing the CA's signing key.
- If the CA's key is lost or compromised, the CA must be stopped and then restarted with a new key. While the CA is unavailable, users who possess certificates signed by the CA cannot use the certificates because there is no means to validate them.

- After the CA has been re-established, certificates signed with the new key must be reissued to all users since the current certificates are signed with an invalid (lost) key.

Storing the signing key in hardware presents a risk to the security of the key and poses a potential administrative disaster. If this choice is selected, a disaster recovery plan should be prepared. If the 4758 coprocessor hardware fails, immediate corrective action must be taken.

7.3 Supported standards

The standards related to the public key infrastructure can be split into two groups: Those that specifically define the PKI and user-level standards that rely on the PKI but do not define it.

As described earlier (refer to 2.4.2, “PKI components and functions” on page 27), PKI is composed of five components: The three major components (Certificate Authority, Registration Authority, and Certificate Repository) are part of the PKI model; the other two components (end entities and relying entities) are not part of the PKI model; they only use it.

PKI standards permit multiple PKIs to interoperate and multiple applications to interface with a single consolidated PKI. In particular, standards are necessary for:

- Enrollment procedures
- Certificate formats
- CRL formats
- Formats for certificate enrollment messages (client requests certificate, server issues certificate)
- Digital signature formats
- Challenge/response protocols

7.3.1 PKI basic standards

The primary focus for interoperable PKI standards is the PKI working group of the Internet Engineering Task Force (IETF), known as the PKIX group (for *PKI for X.509 certificates*).

The PKIX is an open, platform-independent standard. The PKIX specifications include guidelines for CA implementation, certificate revocation, certificate and CRL distribution options, and administrative protocol and procedures.

These specifications are based on two other standards: X.509 from the International Telecommunication Union (ITU-T, formerly known as CCITT) and the Public Key Cryptography Standards (PKCS) from RSA Data Security.

X.509 was intended to specify authentication services for X.500 directory services. The certificate syntax of X.509 has been widely adopted outside X.500 environments.

X.500 Directory

An X.500 Directory is a hierarchical collection of object information. An object can be, for example, a user, a terminal, a distribution list, or OSI application elements. Each directory entry contains a collection of information about a specific object, and the *Directory Information Base* (DIB) contains information about the types of objects.

By definition, distinguished names are the standard form of naming. All information present in the DIB is identified by a *distinguished name*, which is comprised of one or more *relative distinguished names*. The purpose of the distinguished name is to uniquely identify the entities as being part of the DIB or X.500 Directory tree. Each relative distinguished name is comprised of one or more attribute values, which consists of an attribute identifier and its corresponding value information.

Because X.500 is rather complex, the Lightweight Directory Access Protocol (LDAP) has been developed that has become a widespread standard for directories. LDAP is very important for a PKI for storage of certificates and CRLs. See Appendix B, "Introduction to LDAP" on page 321.

The most universally supported PKI standard is the ITU's X.509. The primary purpose of X.509 is to define a standard digital certificate format. The standard format currently required for certificates is Version 3, commonly written as *X.509v3*.

However, X.509 was not intended to define a complete, interoperable PKI. To supplement X.509, vendors, users, and standards committees have turned primarily to de facto PKI standards defined in PKCS.

PKCS are a set of standards for public key cryptography, covering PKI in areas of certificate enrollment and renewal, and CRL distribution. For PKI interoperability, the three most important PKCS standards are:

- PKCS #7 – “Cryptographic Message Syntax Standard,” defined as a general syntax for messages that include cryptographic enhancements, such as digital signatures and encryption.
- PKCS #10 – “Certificate Request Syntax Standard,” the syntax for certification requests.
- PKCS #11 – “Cryptographic Token Interface (Cryptoki),” the interface for cryptographic token, such as Smart Cards.
- PKCS #12 – “Personal Information Exchange Syntax Standard,” specifying a portable format for storing or transporting user’s private keys, certificates, miscellaneous secrets, and so on.

A smaller IETF work has also defined the *Simple Public Key Infrastructure* (SPKI) whose proposal is to eliminate the idea of hierarchy, therefore, proposing a very flat architecture in which all certification, certification retrieval, and verification occur in a simpler way. According to the SPKI, anyone should be capable of performing the CA function, and PKI should operate on a much flatter model.

PKI can also adhere to another standard that address interoperability: it is the *Common Data Security Architecture* (CDSA), a set of specifications adopted by The Open Group (<http://www.opengroup.org>) that outlines pluggability of multiple technologies for ciphering and deciphering data, signing and verifying messages, and implementation of trust models.

The *Common Data Security Architecture* (CDSA) is a cryptographic framework consisting of APIs designed to make computer platforms more secure for applications dealing with electronic commerce, communications, and digital contents.

CDSA supplies a set of security building blocks for Independent Software Vendors (ISVs) and is designed as an overall infrastructure for data security on PCs, workstations, and servers. It is founded on two fundamental data security premises: Digital certificates and portable digital tokens, which store cryptographic keys and perform cryptographic operations.

CDSA defines four different layers, each building on the more fundamental services of the layer below it. The bottom layer is made up of service provider modules that start with basic components (cryptographic algorithms, base certificate manipulation facilities, and storage) and builds up to secure, digital certificate-based transaction protocols in the uppermost layer (System Security Services). CDSA also supports many different programming environments, ranging from ANSI C to Java.

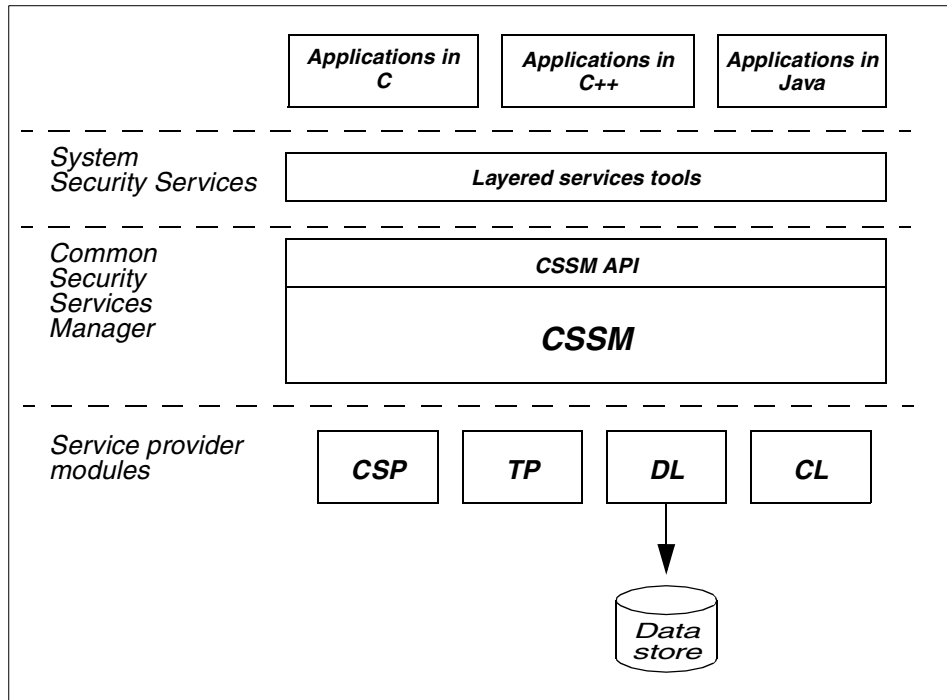


Figure 64. The CDSA architecture

As illustrated in Figure 64, at the core of CDSA is the *Common Security Services Manager* (CSSM). CSSM is an extensible infrastructure that defines a certificate-based API for security services and manages add-in security modules that provide services to applications in accordance with the CSSM-defined API. Four basic service provider module types integrate into the CDSA environment. These are:

- *Cryptographic Service Provider* (CSP) module provides encryption capabilities for the system. In addition, it stores private keys.
- *Trust Policy* (TP) module is a set of rules used to determine if a requester is trusted or authorized to perform an action on a data object.
- *Data Service Library* (DL) module provides persistent storage for security-related data objects used throughout the CSDA product. These objects can be certificates, CRLs, public keys, or trust information. DLs are analogous to a "file cabinet" for security data.
- *Certificate Library* (CL) module performs operations on digital certificates. Each certificate library knows one or more specific certificate structure. It

also provides access to remote signing capabilities, such as Certificate Authorities.

Any number of independent software or hardware vendors can create service provider modules, highlighting the CDSA emphasis on openness and interoperability. Through the CSSM APIs, an application can directly or indirectly select the service provider modules needed for specific security services. The architecture is designed to be both modular and extensible. Extensibility is important because it encourages ISVs to develop incremental functionality. For more information about the CDSA architecture, see:

<http://www.opengroup.org>
<http://www.cdsasecurity.com>

7.3.2 Standards that rely on a PKI

Most major security standards are designed to work with a PKI. For instance, Secure Sockets Layer (SSL), Transport Layer Security (TLS), Secure Multipurpose Internet Mail Extensions (S/MIME), Secure Electronic Transactions (SET), and IP Security (IPSec), all assume, require, or allow the use of a PKI.

S/MIME is the IETF standard for secure messaging. S/MIME assumes a PKI for digitally signing messages and to support encryption of messages and attachments without requiring prior shared secrets. Because e-mail is the most mature of the popular Internet applications, the S/MIME committee has led the way in implementing and extending PKI standards, taking advantage of the PKIX standards when possible, and filling in where additional standards were necessary. S/MIME follows the syntax given in PKCS #7 for digital signatures and encryption. For more information, check the following Web site:

<http://www.rsa.com/rsa/S-MIME>

SSL and the emerging IETF standard, TLS, which is based on SSL, are the most important standards for providing secure access to Web servers. SSL and TLS are also being used for general client/server security in a variety of non-Web applications. Both rely on a PKI for certificate issuance for clients and servers. The SSL handshake protocol supports server and client authentication and is application independent (allowing protocols, such as HTTP, FTP, and Telnet to be layered on top of it transparently).

The SSL protocol consists of two phases: Server authentication and an optional client authentication. In the first phase, a client sends a request to the server, and then the server sends the certificate back to the client along with the appropriate cipher preferences. The client then generates a master

key, which it encrypts with the server's public key, and transmits the encrypted master key to the server. The server recovers the master key and authenticates itself to the client by returning a message authenticated with the master key. Subsequent data is encrypted and authenticated with keys derived from this master key. In the optional second phase, the server sends a challenge to the client. The client authenticates itself to the server by returning the client's digital signature on the challenge as well as its public key certificate.

SET facilitates secure electronic bank card payments. SET uses keys for authentication, confidentiality, and data integrity. PKI is a critical support for authentication of the parties involved in a payment transaction.

The Internet Engineering Task Force's (IETF) Internet Security Protocol (IPSec) standard defines protocols for IP encryption and is one of the primary protocols used for deploying Virtual Private Networks (VPNs). IPSec requires keys for encryption and authentication. Complete PKI standards for IPSec are emerging, and a PKI is the most scalable way of managing IPSec keys. Use of IPSec is still fairly limited; however, the need for PKI grows with IPSec deployment.

Some protocol is required to allow open access to public keys and certificates within the secure domain. One convention stands out as the most effective in this situation: The Lightweight Directory Access Protocol (LDAP). LDAP represents an established, general-purpose Internet Engineering Task Force (IETF) standard. LDAP is a specification for a client-server protocol for accessing a directory service and retrieving and managing directory information. It was initially used as a front-end to X.500 but can also be used with stand-alone and other kinds of directory servers that follow the X.500 data models.

The LDAP information model is based on the entry, which contains information about some object, for example, a person. Entries are composed of attributes, which have a type and one or more values. Each attribute has a syntax that determines what kind of values are allowed in the attribute and how those values behave during directory operations. Refer to Appendix B, "Introduction to LDAP" on page 321 for more information.

7.4 Scalability

Scalability of a PKI solution must be considered from two different angles. First, the *infrastructure* itself must be scalable, and second, the *systems* that provide the functions must be scalable.

7.4.1 Infrastructure

A *scalable infrastructure* includes policies and rules that govern the issuance, lifecycle, and revocation of certificates. If the policies are such that certificate management becomes difficult, then the whole PKI solution is likely not to be scalable. Key and certificate lifecycles must be chosen such that a viable compromise between security requirements and manageability is achieved. Shorter lifecycles increase the safety but add to the burden of administration. Another fact to consider is the number of certificates that are issued per End Entity (for example, per user). While some PKI deployment only issue one certificate per user, others choose two per user: One for signing and another for encryption and decryption. There could be more certificates per user, one for each possible purpose, but such a solution only adds unnecessary administration overhead. The storage and maintenance of certificates should be chosen (or even mandated) such that only a relatively small number of certificates need to be revoked and reissued. Finally, certificate issuance to multiple End Entities should be scattered over time to prevent reissuance of a large number of certificates within a short period of time.

The features of cross-certification and hierarchical trust relationships supported by Trust Authority allow very large environments to be split up into smaller security domains that are easier to manage and maintain.

7.4.2 Systems

A *scalable system* platform is a platform that can grow with a higher load. Trust Authority supports Windows NT up to powerful AIX systems. Furthermore, the individual Trust Authority functions can be given out to different systems to spread the load. The high-performance and highly scalable IBM DB2, used by Trust Authority components and the IBM SecureWay Directory Server (LDAP server), provides for a solid base on which the load can grow.

The optional IBM SecureWay 4758 PCI Cryptographic Coprocessor can offload some of the computing-intensive work from the CPU where it is installed. The IBM SecureWay 4758 PCI Cryptographic Coprocessor is supported on the CA server on AIX only.

7.5 Interoperability with other FirstSecure components

Although not required and not directly related with other building blocks of IBM SecureWay FirstSecure, Trust Authority can interoperate with them in various ways:

- Trust Authority can be the issuing authority for the various certificates that can or must be used in other building blocks, such as:
 - Certificate authentication with Policy Director
 - Certificates for the many possibilities of SSL communications
 - Certificates for secure Web servers and browsers (where required)
- Trust Authority systems can benefit from the additional security provided by the SecureWay Boundary Server. To have maximum security, Trust Authority systems should be isolated such that only required access is possible from the network.
- Intrusion Immunity helps detect potential attacks against Trust Authority servers.
- Trust Authority can share an LDAP directory with Policy Director to ease administration overhead.
- Policy Director can be used to control access to Trust Authority servers if this is desired.

Chapter 8. The SecureWay Toolbox (TB)

The last FirstSecure building block to be described is the SecureWay Toolbox, which is a set of tools and APIs available for developing secure middleware and applications. The toolbox is only available with FirstSecure and is provided via Web download from a secured Web site. Web access instructions are provided with the FirstSecure media pack.

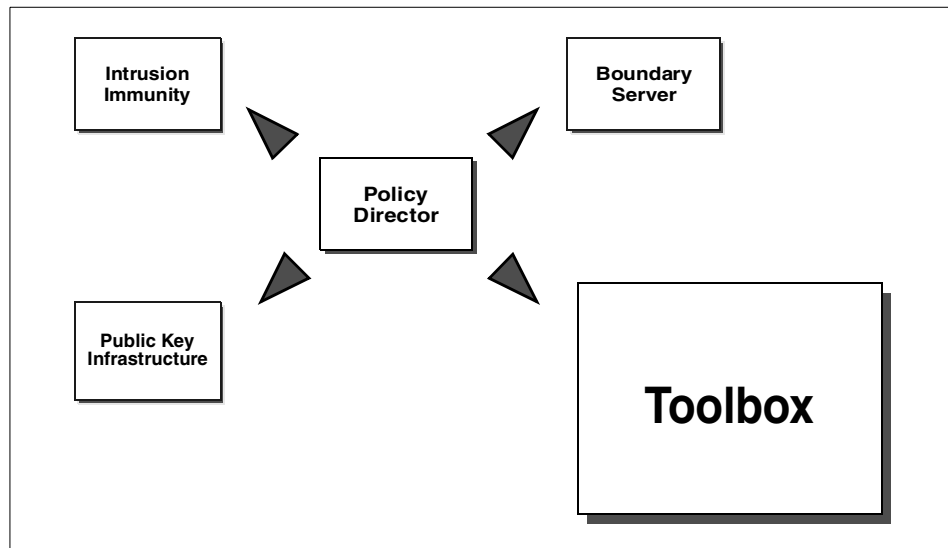


Figure 65. Toolbox

The toolbox offers system integrators, solution providers, and developers a way to exploit cryptographic and other security services in their applications using standard APIs and related cryptographic, security, and directory services with simple installation (for a description of the supported APIs, refer to 8.1, "Toolbox APIs" on page 192). This set of APIs allows the users to utilize and modify the security features of FirstSecure based on their needs and security experience, therefore, focusing on the security of their e-business rather than the programming details.

Trust management services are provided by the KeyWorks CDSA Toolkit (see 8.2.1, "IBM KeyWorks Toolkit" on page 194 for more information). It is a cryptographic middleware framework that provides cryptographic services and a trust environment to application and middleware in heterogeneous operating environments.

The toolbox also includes the IBM Key Recovery Service Provider 1.1 toolkit (refer to 8.2.1.1, “IBM Key Recovery Service Provider” on page 196) that is an application plug-in used to recover encrypted information. It uses the functions provided by the IBM KeyWorks Toolkit and Key Recovery Server.

8.1 Toolbox APIs

The Toolbox provides the tools for programmers to exploit cryptographic and other security services in applications using standard API interfaces.

The Toolbox contains access to Certification Authority services (PKI), Authorization services (Policy Director), LDAP, SSL protocol, and cryptographic services (Keyworks).

The SSL API is implemented only in Java language, while the LDAP API is implemented in both Java and C languages.

8.1.1 LDAP

The LDAP API provides Lightweight Directory Access Protocol client services. It provides access to directory servers through the use of LDAP V2 and V3 protocols.

The LDAP API defines the following functions:

- Initialize and open an LDAP connection
- Initialize and open a secure (SSL) LDAP connection
- Bind (authenticate) to an LDAP server
- Search the directory and retrieve the results
- Compare directory entries
- Modify (add, change, rename, delete) directory entries
- Control functions to specify and change operations parameters

The LDAP API is supported on the Windows NT 4.0, AIX 4.2.x, and Sun Solaris 2.6 systems and is provided in both the C and Java languages.

8.1.2 PKIX

The PKIX API is a collection of certificate services used to generate, revoke, sign, and verify PKIX certificates. Based on the IBM KeyWorks toolkit, it provides basic PKI capabilities including CA (Certificate Authority), RA

(Registration Authority), EE (End Entity), certificate life cycle management, and interfacing with the PKI functions in FirstSecure.

The PKIX API provides three classes on the server side:

- CA class for managing the creation, storage, and revocation of certificates.
- RA class, which manages user registration of certificates.
- EE class, where most certificates are created. Also, the End-Entity is where certificates and their keys are stored.

PKIX provides the following methods for each class:

- *Generate method* for certificate generation
- *Revoke method* for certificate revocation
- *Sign method* for certificate sign
- *Verify method* for certificate verification

For more information, please see Chapter 8.3, “Toolbox component APIs” on page 203. The PKIX API is supported on the AIX 4.2.x and Sun Solaris 2.6 systems and is provided in the C language environment.

8.1.3 Authorization API

The Authorization API provides an integration interface to the Policy Director authorization functions. It can be used by software resource managers and clients to enforce access control policy. It enables applications to utilize FirstSecure Policy Director access control functions.

The Authorization API is fully documented in the *IBM SecureWay Policy Director Programming Guide and Reference*. It is available for download from the IBM Security Web site located at:

<http://www.ibm.com/software/security/library>

The Authorization API is supported on the Windows NT 4.0, AIX 4.2.x, and Sun Solaris 2.6 systems and is provided in the C language.

8.2 Toolbox security services

The SecureWay Toolbox Security Services include:

- IBM KeyWorks 1.1.3 Toolkit, a cryptographic and trust services toolkit, which enables applications to provide these functions. It works with the PKIX high-level API.
- The SSLite Java based toolkit, which provides basic SSL protocol and utilities. It uses the SSL API.
- LDAP services, which provide access to LDAP V2 and V3 servers and Policy Director. This service works with the LDAP API.

8.2.1 IBM KeyWorks Toolkit

The IBM KeyWorks Toolkit provides a set of layered security services and associated programming interfaces designed to furnish an integrated set of information and common security capabilities. It helps accelerate the development of cryptographic and security programs by providing an insulating layer between application programs and cryptographic and trust functions.

This insulating layer provides standard interfaces that applications can use to invoke critical cryptographic and security services as well as standard interfaces that Service Providers (see 8.2.1.1, “IBM Key Recovery Service Provider” on page 196) can use to plug into the toolkit.

These standard interfaces are based on CDSA, the Common Data Security Architecture Version 2.0, which was recently adopted by The Open Group as an industry-accepted specification for the development of secure applications that are interoperable, extensible, and offer cross-platform support (see 7.1, “PKI for X.509 V3 architecture” on page 169 for more information about CDSA architecture).

The KeyWorks architecture consists of a set of layered security services and associated programming interfaces designed to furnish an integrated set of security capabilities for security applications. Each layer builds on the more fundamental services of the layer directly below it.

These layers start with fundamental components, such as cryptographic algorithms and random numbers, and builds up to digital certificates, key management and recovery mechanisms, and secure transaction protocols in higher layers.

Figure 66 shows a simplified view of the layered architecture of a system built on top of KeyWorks.

There are four major layers in the architecture: Application Domains, System Security Services, KeyWorks framework, and Service Providers.

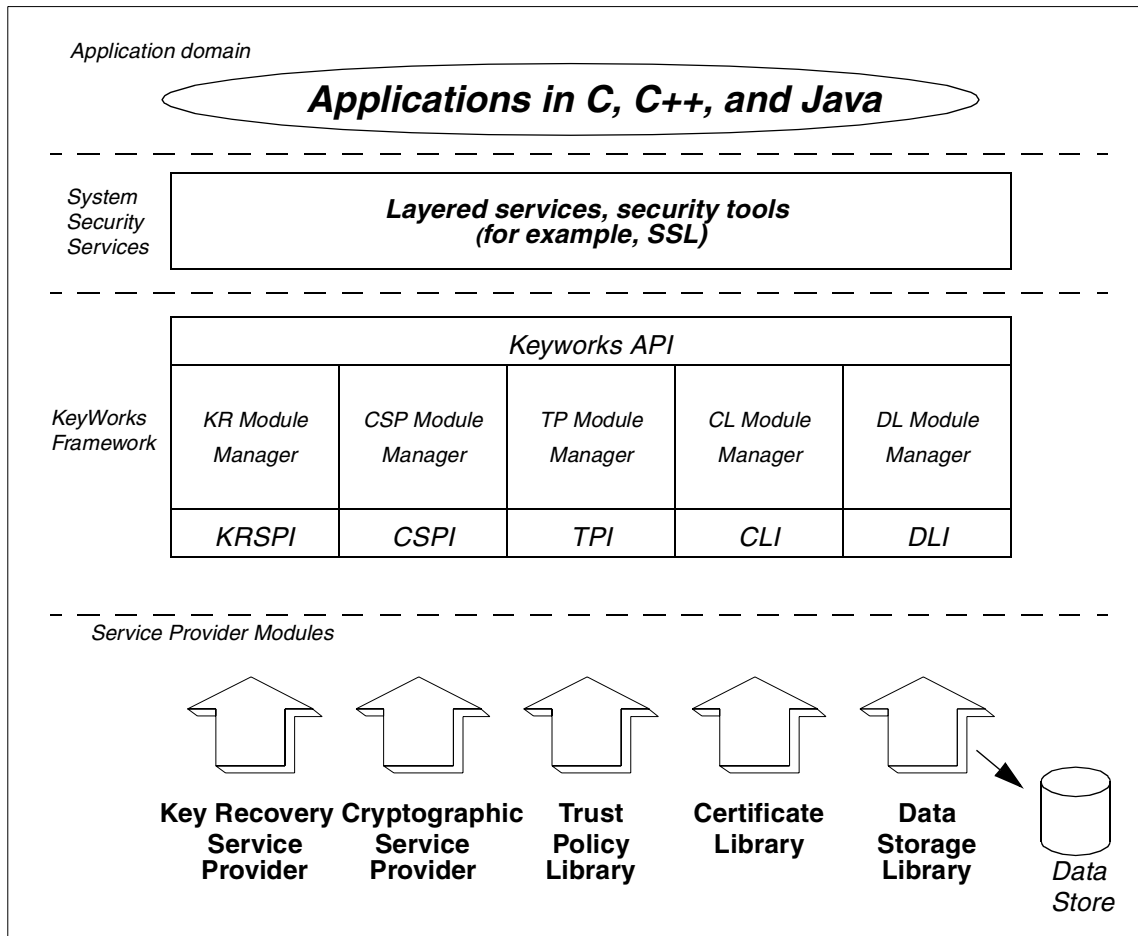


Figure 66. KeyWorks architecture

The KeyWorks framework defines a common security API that must be used by the applications to access the services of service provider modules.

The KeyWorks framework is also extendible; as additional functions are defined, the framework can be extended to include them.

The toolkit also includes service provider plug-in modules for the following functions: Cryptography, certificate recognition, trust policy verification, and data (certificate and key) storage. These modules support open standards including PKCS#11 tokens, RSA cryptographic functions, X.509V3 certificates, and LDAP. The toolkit does not require the use of any specific certificate authority for the creation of certificates.

The APIs provided by the KeyWorks toolkit are as follows:

- Core services API
- Privilege mechanism API
- Cryptographic services API
- Key recovery services API
- Trust policy services API
- Certificate library services API
- Data storage library services API
- Error handling API

A key recovery system that enables the recovery of encrypted information without distributing the key is incorporated in the KeyWorks toolkit. This key recovery system is provided by the Key Recovery Service Provider toolkit.

8.2.1.1 IBM Key Recovery Service Provider

The IBM Key Recovery Service Provider is a service provider module that uses the standard functions provided by the IBM KeyWorks Toolkit. It is a plug-in module for KeyWorks that creates key recovery blocks.

This service provider enables the recovery of stored and transmitted encrypted information without collecting and escrowing private keys and creating single points of cryptographic vulnerability. The key recovery information is based on the session key associated with the communication between correspondents.

The key recovery process is an add-on to existing encryption schemes and could be optionally invoked (through the framework) by cryptographic services in support of an application. The framework enables the key recovery service to be used with any cryptographic service provider that uses the SPI.

The framework also protects against the bypassing of key recovery (if that is the organization's or jurisdiction's policy) and sets up a complete key recovery environment before any encrypted communication takes place.

8.2.1.2 IBM Key Recovery Server

The IBM Key Recovery Server is an NT-based application that is required for developing and testing application runtime environments using the KeyWorks framework and Key Recovery Service provider. It is based on the KeyWorks

framework with altered user and communications interfaces to allow recovery of corporate data by using the IBM Key Recovery process.

8.2.2 SSL APIs

Secure Socket Lite uses SSL for controlling data access. It encrypts data using public and private keys for several purposes including user authentication, prevention of access by unauthorized clients, and prevention of data tampering.

Secure Socket Lite does not function alone, but SSL technology is incorporated in several other APIs for encrypting data and creating passwords.

The toolbox provides the *SSL API* and the *SSLight Socket API* for Java environments.

8.2.2.1 The SSLight Socket API

The *SSLight Socket API* is based on the regular Java Socket API. It provides `SSLSocket` and `SSLServerSocket` classes that can be used similar to `java.net.Socket` and `java.net.ServerSocket` classes.

These SSL classes are subclasses of the corresponding Java Socket classes.

When creating a SSL socket, an already connected socket can be supplied to be used for the connection. It can also be specified that closing the SSL socket should not close the underlying socket. By these means, a number of SSL connections can be created and closed on the same socket.

The operation of the SSL V3.0 protocol is controlled via the `SSLContext` objects. SSL context defines a list of cipher suites and compression methods that can be used by a SSL connection and specifies the key ring used by the SSL protocol. The key ring contains the X.509v3 certificates of signers and sites trusted by the application as well as the private certificates and private keys of the application itself. The content of the key ring can be stored in a byte array or in a character string embedded in a Java class. SSL context provides methods for import and export of such key ring repositories. A `SSLight` key ring repository can be signed, and the sensitive private key information contained in the repository can be encrypted, which is always highly recommended. The key ring repository signature end encryption is password based.

The `SSLight` key ring repository can be down-loaded as a part of an applet; for security reasons, either https has to be used to down-load the applet or

the applet has to be signed. It is important to verify that the security of the applet code and the key ring repository of the applet are not compromised.

SSL Context provides a number of methods that can be redefined by sub-classing to control the operation of the SSL protocol dynamically. For instance, the method *handleCertificateChain* is called by SSL if a certificate chain has to be validated by the SSL protocol but that cannot be done based on the information stored in the key ring repository associated with the context or there is no key ring defined at all. By overriding that method, the application can implement its own certificate verification policy. If an application wants to use the built-in certificate verification, but perform some additional checks afterwards, it might redefine the *confirmCertificateChain* method of the SSLContext.

The functionality to manage the SSL session cache is provided by the SSLSession class. It provides a method to specify the session cache size. The cache size defines the maximal number of passive sessions (sessions without connections) maintained at the same time. If the cache size is set to zero, no passive sessions will be resumed. The session cache size has no implications on resuming active sessions in which parameters are used by ongoing connections.

8.2.3 LDAP API

The IBM SecureWay Directory uses LDAP technology to allow you to organize, control, and access your directories. It is based on a client/server model that provides client access to an LDAP server.

SecureWay Directory provides a means of maintaining directory information in a central location for storage, updating, retrieval, and exchange. It uses SSL support to create passwords and encrypt information so that you can control who has access to your directories.

The toolbox provides access to all LDAP API function calls for LDAP V3 provided by the IBM SecureWay Directory Client SDK. The function calls are listed in the tables below.

Table 20 lists the functions used to initialize a connection and authenticate a client to an LDAP server.

Table 20. Functions that initialize and terminate a connection

Function	Description
<code>ldap_init()</code>	Initialize a session with an LDAP server

Function	Description
ldap_simple_bind() ldap_simple_bind_s()	Initiate a simple bind to an LDAP server
ldap_sasl_bind() ldap_sasl_bind_s()	Authenticate the client to an LDAP server using the Simple Authentication Security Layer
ldap_ssl_init() ldap_ssl_client_init() ldap_ssl_start	Initialize a secure session using the Secure Sockets Layer (SSL) protocol
ldap_set_rebind_proc()	Reauthenticate, for example, when another server through a referral result message is involved
ldap_unbind() ldap_unbind_s()	Close an LDAP session, dispose the session handle

Table 21 lists the session-handling functions that are used to influence the session handle options once a connection is initialized.

Table 21. Session-handling functions

Function	Description
ldap_set_option()	Set the value of a specified option
ldap_get_option()	Get the value of a specified option

The interactions with the server are shown in Table 22. They are used to send and receive data through the network to/from an LDAP server.

Table 22. Functions that send or receive data

Function	Description
ldap_search() ldap_search_s()	Initiates a synchronous or asynchronous search of an LDAP directory
ldap_search_ext() ldap_search_ext_s()	Like ldap_search(), but server and client controls can get specified
ldap_search_st()	Like ldap_search_s(), but a time value for the API to wait until the results are received can get specified
ldap_compare() ldap_compare_s()	Compares a given attribute value against the actual one stored within the LDAP server
ldap_compare_ext() ldap_compare_ext_s()	Like ldap_compare(), but the comparison of binary values is possible
ldap_modify() ldap_modify_s()	Adds, deletes or replaces values of an attribute
ldap_modify_ext() ldap_modify_ext_s()	Like ldap_modify(), but LDAP V3 server and client controls are supported

Function	Description
ldap_modrdn() ldap_modrdn_s()	Changes the name (RDN) of an entry
ldap_rename() ldap_rename_s()	Modifies the distinguished name of an entry
ldap_add() ldap_add_s()	Add an entry to an LDAP directory
ldap_add_ext() ldap_add_ext_s()	Like ldap_add(), LDAP V3 client and server controls are supported
ldap_delete() ldap_delete_s()	Deletion of leaf entries
ldap_delete_ext() ldap_delete_ext_s()	Like ldap_delete(), LDAP V3 client and server controls are supported
ldap_abandon() ldap_abandon_ext()	Abandon operation in progress. LDAP V3 client and server control supported by ldap_abandon_ext()
ldap_result()	Obtaining the result of a previously issued asynchronous operation

The functions for error handling, those used to retrieve the errors of previous LDAP function calls, are listed in Table 23.

Table 23. Functions for error handling

Function	Description
ldap_parse_result()	Returns error code and other information of previous API function call
ldap_parse_sasl_bind_result()	Returns error code of a SASL bind call
ldap_get_errno()	Returns error code of most recent operation
ldap_err2string()	Converts numeric error code into an error string

The following functions (Table 24) are used to step through the results obtained by synchronous or asynchronous search functions.

Table 24. Parsing the results

Function	Description
ldap_count_messages()	Counts number of messages in the LDAP message structure
ldap_first_message()	Returns first message in a chain of results obtained by ldap_result()

Function	Description
ldap_next_message()	Returns entry in a chain of results obtained by ldap_result()
ldap_first_entry() ldap_next_entry()	Stepping through a chain of search results
ldap_count_entries()	Counts the number of entries returned by a search operation
ldap_first_reference() ldap_next_reference()	Retrieving and stepping through a list of continuation references obtained by a search result
ldap_count_references()	Returns the number of references
ldap_parse_reference_np()	Extracts referrals and controls from a search result
ldap_first_attribute() ldap_next_attribute()	Stepping through a list of attributes returned with an entry
ldap_count_attributes()	Returns the number of attributes
ldap_get_values()	Get values of a given non-binary attribute
ldap_get_values_len()	Get values of a given attribute
ldap_count_values() ldap_count_values_len()	Count returned values of a attribute
ldap_get_dn()	Retrieve the name of an entry
ldap_explode_dn() ldap_explode_rdn()	Breaks up a DN or RDN into its components
ldap_get_entry_controls_np()	Extracts LDAP controls from an entry

The functions listed in Table 25 are used to free memory occupied by search results, attribute values, and so on.

Table 25. Memory-freeing functions

Function	Description
ldap_ber_free()	Frees a buffer used by ldap_first_attribute() and ldap_next_attribute() to keep track of the current position in an entry
ldap_msg_free()	Frees results of previous calls to ldap_result or an synchronous search routine
ldap_memfree()	Frees memory occupied by LDAP library functions, such as ldap_next_attributes() or ldap_get_dn()
ldap_mods_free()	Frees memory occupied by previous modify functions

Function	Description
ldap_value_free() ldap_value_free_len()	Frees memory occupied by values returned through ldap_get_values() or ldap_get_values_len()
ldap_free_url_desc()	Frees memory occupied by a URL description obtained from ldap_url_parse()

The functions listed in Table 27 are used to perform operations with LDAP URLs.

Table 26. LDAP URL functions

Function	Description
ldap_is_ldap_url()	Checks the validity of an LDAP URL
ldap_url_parse()	Parses and separates parts of an LDAP URL
ldap_url_search() ldap_url_search_s() ldap_url_search_st()	Performs a search based on the contents of an LDAP URL

The functions listed in Table 27 provide character encoding and translation services.

Table 27. Character encoding functions

Function	Description
ldap_xlate_local_to_utf8() ldap_xlate_utf8_to_local() ldap_xlate_local_to_unicode() ldap_xlate_unicode_to_local() ldap_set_locale() ldap_get_locale() ldap_set_iconv_local_codepage() ldap_get_iconv_locale_codepage() ldap_set_iconv_local_charset()	Functions for managing the conversion of strings between UTF-8 and a local code page

All the functions used to carry out other various operations that are not covered by any other category above are listed in Table 28 below.

Table 28. Other functions

Function	Description
ldap_msgtype()	Returns the type of an LDAP message
ldap_msgid()	Returns the ID of an LDAP message passed as a parameter to an asynchronous call

Function	Description
ldap_version()	Retrieves basic information about the API implementation, such as SDK version or protocol version
ldap_control_free() ldap_controls_free()	Disposes a single client control or an array of client controls allocated by LDAP API calls
ldap_password_set() ldap_password_get()	Sets or gets the password from local configuration using a supplied encryption key
ldap_default_dn_set() ldap_default_dn_get()	Sets or gets the default DN in local configuration
ldap_enetwork_domain_set() ldap_enetwork_domain_get()	Sets or gets the users default DNS domain name
ldap_server_locate() ldap_server_conf_save() ldap_server_free_list()	Manages information about LDAP servers retrieved from a DNS server

For more information about the LDAP APIs, see the *IBM SecureWay Directory Client SDK Programming Reference* that is included as on-line HTML documentation with the LDAP Client SDK.

8.3 Toolbox component APIs

The toolbox component APIs include the *PKIX* and the *Authorization APIs*. The PKIX API provides basic PKI capability including CA, RA, certificate life cycle management, and enablement for interface with the Trust Authority application. The Authorization APIs can be used by software resource managers and clients to enforce access control policy. They provide enablement of applications so that they have access control enforced by the FirstSecure Policy Director.

8.3.1 PKIX API

The PKIX API enables you to create, manage, store, distribute, and revoke certificates based on public-key cryptography. Specifically, the API provides the following components:

- A Certificate Authority (CA)
- A Registration Authority (RA)
- Support for a client system, also referred to as an end entity (EE)
- Control objects
- An object store

The Certificate Authority (CA) is a server application trusted by one or more users to create, manage, and revoke certificates. The CA is responsible for the following:

- Enforcing security policies (such as a certificate validity period or key revocation policy)
- Creating certificates
- Revoking certificates
- Maintaining lists of trusted RAs and cross-certified CAs

The Registration Authority (RA) is a server application entity given the responsibility for performing some of the administrative tasks necessary in the registration of subjects, such as:

- Confirming the subject's identity
- Validating that the subject is entitled to have the attributes requested in a certificate
- Verifying that the subject has possession of the private key associated with the public key requested for a certificate

The end entity (EE) initiates certificate and revocation requests and maintains the private keys and certificates.

The API uses two objects to control the processes of creating and revoking certificates: The certificate request and the revocation request. Each of these objects acts as a unique container for information needed by the RA or CA to create or revoke certificates. The objects are primarily created at the EE (although the RA and CA can also create them) and sent from EE to RA to CA and back again. The RA and CA can modify these requests to add their own constraints (such as a specific length of time for certificate validity). The CA converts a certificate request into a certificate and one or more revocation requests into a certificate revocation list (CRL).

Once a certificate request becomes a certificate, a copy of that certificate is placed in the certificate store where the CA can retrieve information about the certificate during its lifetime. The certificate store contains copies of each certificate created by the CA, which are indexed by serial number.

The tables that follow list, and briefly describe, some of the IBM PKIX interfaces used to create and manage certificates. For a full list and detailed description, please refer to the *IBM SecureWay X.509 Public Key Infrastructure for Multiplatforms Programming Guide and Reference* that can be downloaded from:

<http://www.ibm.com/software/security/firstsecure/library>

Table 29 shows the routines used to register users and create certificates.

Table 29. Users registration and certificates creation

API Name	Function
JNH_authorize_registration	Authorizes a user registration request
JNH_create_certificate	Creates a certificate for a user
JNH_preregister_user	Prepares for a user registration
JNH_publish_certificate	Publishes a certificate to an LDAP directory
JNH_RA_preregister_user	Creates a preregistration record for an end-user registration
JNH_register_user	Submits a user registration request
JNH_reject_registration	Rejects a user registration request
JNH_validate_registration	Validates that the version and signing algorithm are present in a certificate request

The routines described in Table 30 are used for revoking certificates and creating Certificate Revocation Lists.

Table 30. Revoking certificates and creating CRLs

API Name	Function
JNH_authorize_revocation	Approves a revocation request for a certificate
JNH_cancel_revreq	Cancel a revocation request
JNH_create_CRL	Creates a certificate revocation list
JNH_create_revreq	Creates a revocation request
JNH_create_revreq_from_certificate	Creates a revocation request for a certificate on a smart card
JNH_new_revreq	Creates a new revocation request
JNH_publish_CRL	Publishes a certificate revocation list to the RA
JNH_reject_revocation	Rejects a revocation request
JNH_request_CRL	Requests a CRL
JNH_request_revocation	Creates a revocation request
JNH_revoke_certificate	Revokes a certificate

Table 31 shows routines used to obtain information about certificate requests.

Table 31. Information about certificate requests

API Name	Function
JNH_inquire_certreq_basicConstraints	Retrieves the basic constraints
JNH_inquire_certreq_enddate	Retrieves the expiration date
JNH_inquire_certreq_issuer	Retrieves the issuer name
JNH_inquire_certreq_KeyUsage	Retrieves the key-usage information
JNH_inquire_certreq_privkey_EE	Returns the key length and algorithm of the EE-generated private key
JNH_inquire_certreq_serialnumber	Retrieves the serial number
JNH_inquire_certreq_startDate	Retrieves the starting date
JNH_inquire_certreq_status	Retrieves the status
JNH_inquire_certreq_subject	Retrieves the subject name
JNH_inquire_certreq_subjectkey_algorithm	Retrieves the subject key algorithm

Table 32 shows the routines that provide information about certificates in the EE's key store.

Table 32. Certificate requests modification

API Name	Function
JNH_cert_inquire_enddate	Retrieves the expiration date
JNH_cert_inquire_issuer	Retrieves the issuer name
JNH_cert_inquire_keyUsage	Retrieves the key usage information
JNH_cert_inquire_serialNumber	Retrieves the certificate serial number
JNH_cert_inquire_startDate	Retrieves the starting date
JNH_cert_inquire_subject	Retrieves the subject name
JNH_Keypair_Selected	Selects a key pair from a list of key pairs
JNH_keystore_inquire_cert	Retrieves a certificate from the key store
JNH_List_Existing_Keypairs	Lists any existing key pairs
JNH_list_SC_certs	Returns the subject names and key identifiers for all the certificates in the key store

Table 33 shows the routines used in order to modify the certificate requests.

Table 33. Certificate requests modification

API Name	Function
JNH_add_certreq_extension	Adds an unsupported extension to a certificate request
JNH_modify_certreq_extension	Modifies certificate extensions
JNH_remove_certreq_extension	Removes an unsupported extension
JNH_set_certreq_basicConstraints	Sets the basic constraints
JNH_set_certreq_endDate	Sets the expiration date
JNH_set_certreq_issuer	Sets the issuer name
JNH_set_certreq_keyUsage	Sets the key-usage
JNH_set_certreq_privkey_EE	Returns the private key algorithm and key length of an EE
JNH_set_certreq_startDate	Sets the starting date
JNH_set_certreq_subject	Sets the subject name

Table 34 shows the routines that are available to retrieve information about revocation requests.

Table 34. Retrieval of information about revocation requests

API Name	Function
JNH_inquire_revreq_certissuer	Returns the name of the certificate issuer
JNH_inquire_revreq_certserialnumber	Returns one of the certificate serial numbers from the revocation request
JNH_inquire_revreq_certserialnumbers	Returns a list of the certificate serial numbers for the certificates in a revocation request
JNH_inquire_revreq_hold_instruction_code	Returns the hold_instruction_code from one of the certificates being revoked
JNH_inquire_revreq_invalidDate	Returns the invalidity date from one of the certificates being revoked
JNH_inquire_revreq_reason	Returns the reason a certificate is being revoked
JNH_inquire_revreq_requests	Determines how many certificates are being revoked in a revocation request

Table 35 shows the routines used if the revocation requests need to be modified.

Table 35. Revocation requests modification

API Name	Function
JNH_set_revreq_certissuer	Sets the name of the certificate issuer in the CertDetails section of the revocation request
JNH_set_revreq_certserialnumber	Sets the certificate serial number of a certificate being revoked
JNH_set_revreq_hold_instruction_code	Sets the hold_instruction_code for a certificate being revoked
JNH_set_revreq_invalidDate	Sets the invalidity date for a certificate being revoked
JNH_set_revreq_reason	Sets the reason a certificate is being revoked

Table 36 shows the routines that are used to create and manage browser-based certificate requests.

Table 36. Creation and management of browser-based certificate requests

API Name	Function
JNH_CA_nonpkix_user_check	Checks the status of a certificate request submitted by JNH_CA_register_nonpkix_user
JNH_CA_nonpkix_user_done	Removes from the object store the certificate request created by JNH_CA_preregister_nonpkix_user
JNH_CA_nonpkix_user_get_cert	Retrieves the certificate for a non-PKIX entity
JNH_CA_preregister_nonpkix_user	Creates an initialization request for anon-PKIX entity
JNH_CA_register_nonpkix_user	Submits a certificate request for a non-PKIX entity
JNH_RA_nonpkix_create_revreq	Creates a revocation request object for a non-PKIX end entity
JNH_RA_nonpkix_request_revocation	Requests a revocation for a certificate for a non-PKIX end entity
JNH_RA_nonpkix_user_check	Checks the status of a certificate request submitted by a non-PKIX entity
JNH_RA_nonpkix_user_done	Removes from the object store the certificate request created by JNH_RA_preregister_nonpkix_user
JNH_RA_nonpkix_user_get_cert	Retrieves the certificate for a non-PKIX entity

API Name	Function
JNH_RA_preregister_nonpkix_user	Creates an initialization request for a non-PKIX entity
JNH_RA_register_nonpkix_user	Submits a certificate request for a non-PKIX entity

Table 37 shows the routines that are used to perform various other tasks related to objects and certificates (the list is not complete; only selected routines are listed):

Table 37. Performance of various other tasks related to objects and certificates

API Name	Function
JNH_CA_list_RAs	Lists the RAs trusted by a CA
JNH_create_enrollment_request	Creates an enrollment request
JNH_delete_object	Deletes an object
JNH_enroll	Sends an RA enrollment request to the CA
JNH_export_credential	Exports a certificate or key to a PKCS12 file, a Smart Card, or a real Smart Card
JNH_get_CA_info	Finds a CA's URL, certificate subject name, and serial number
JNH_get_error	Retrieves text associated with a message code
JNH_get_object_state	Retrieves the status for an object
JNH_INI_readString	Reads a string from the Jonah.ini file
JNH_INI_writeString	Writes a string into the Jonah.ini file
JNH_list_objects	Causes all active objects to announce themselves
JNH_list_surrogates	Causes all surrogate objects to announce themselves
JNH_register_callbacks	Registers the callbacks for JNH_Notify and JNH_Display
JNH_release_object	Unlocks an object in the object store without first saving it
JNH_reserve_object	Locks an object in the object store
JNH_save_object	Saves and unlocks an object in the object store

API Name	Function
JNH_start_server	Initializes a server and opens or creates an object store
JNH_stop_server	Shuts down a server in an orderly fashion

These APIs along with the header files, code samples, and libraries, comprise the PKIX Software Developer's Kit (SDK).

8.3.2 Policy Director Authorization API

The *Policy Director Authorization API* is an application programming interface that allows Policy Director components and third-party applications to query the Policy Director Authorization Service to make authorization decisions and get Yes/No responses. It also can be used by resource managers and clients to enforce the access control policy.

One of the primary values and benefits of the Authorization API is its ability to shield the user from the complexities of the Authorization Service mechanism itself. The Authorization API makes it possible to request an authorization check and get a simple yes or no answer in return. The details of the authorization check mechanism are invisible to the user.

Standard programming interfaces are utilized and are independent of the authentication service. The high level of abstraction of the API encourages the use of centrally defined policy.

The Authorization API allows you to make standardized calls to the centrally managed Authorization Service from any legacy or newly developed application. It works independently from the underlying security infrastructure, the credential and entitlement format, and the evaluating mechanism. Issues of management, storage, caching, replication, EPAC formats, and authentication methods are all hidden behind the Authorization API.

Although the API incorporates generic credential manipulation functions, it does not address the credential format used. Finally, the API does not incorporate an audit interface.

The Authorization API components are illustrated in Figure 67 where the layer of services provided is outlined.

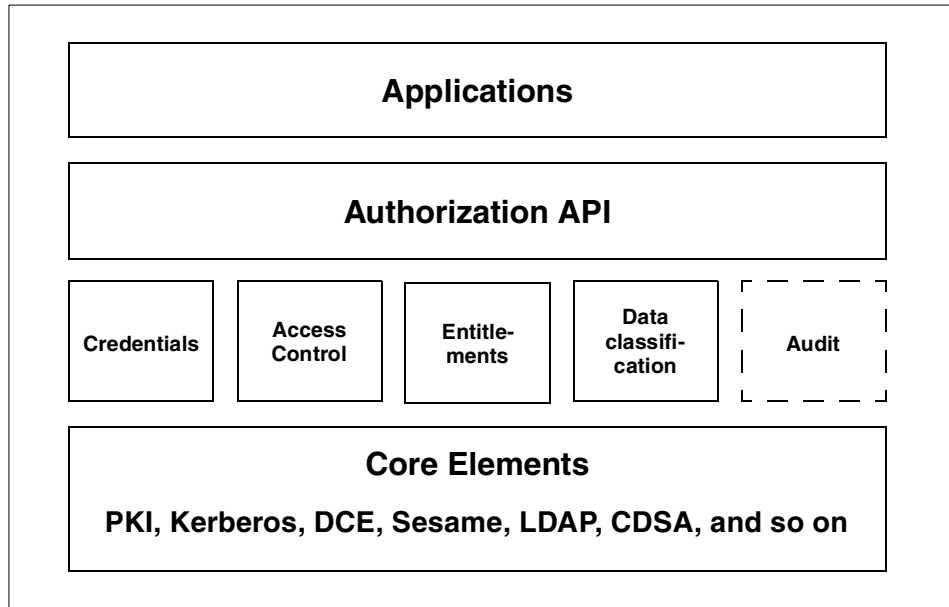


Figure 67. Authorization API layers

The above diagram is composed of a number of program design layers. The Application layer represents the user or client of the Authorization API. The Authorization API provides a standardized interface to the authorization services of credential acquisition, access control, entitlements, and data classification. The credential service API provides applications with an interface through which existing authentication system credentials can be mapped to authorization credentials. The task of access control are encapsulated in interfaces designed to allow different authorization models beneath them. The Entitlement interface provides a mechanism by which an API user can retrieve the entitlements that are granted in the system in a manner that is independent of the underlying API implementation. Finally, the data classification API implements the classification and labeling abstractions in the authorization framework. As it has been explained before, the auditing services are not included. The Core Elements layer represents the authentication mechanism, directory services, and database application that will be used by the authorization service mechanism to implement the authorization model underneath the authorization services.

The Authorization API can be used in one of two ways:

- Policy Director components, such as WebSEAL, NetSEAL, and ObjectSEAL, use an integrated implementation of the API that works

seamlessly with the Authorization Service to process authorization requests.

- Third-party applications can use the API to make the function calls to a special component of the Authorization Service that handles authorization requests from such applications.

Policy Director WebSEAL and NetSEAL dynamically use a similar Authorization API to inform the Policy Director Authorization Service of their applicable object space and to request authorization services.

The Authorization API may also be used in either of two modes, *replica mode* or *remote mode*, as described in the following.

If the API is used in *replica mode*, it downloads and maintains local replica of the authorization database on a local file system. Policy Director components WebSEAL and NetSEAL use this integrated implementation (secmgrd). The Authorization Service, Authorization API, and WebSEAL/NetSEAL are coded as a seamless process known as the Security Manager (secmgrd). The Security Manager is the combined network guard and authorization decision-making process for WebSEAL and NetSEAL communication. The Authorization API is built into the process. The WebSEAL component processes all HTTP communication. The NetSEAL component processes most TCP/IP legacy communication requests for ports, hosts, and networks.

If the API is used in *remote mode*, third-party applications can forward requests to IV Authorization Server (ivacl). ivacl can reside locally or remotely. The structure of the Authorization Service for third-party applications is different from the WebSEAL/NetSEAL architecture. Administrators program their applications to use the Authorization API to make requests of the Authorization Service for authorization decisions. Third-party applications use the function calls provided by the Authorization API to communicate to a special component called the Authorization Server (ivacl). The Authorization Server functions as the authorization decision-making evaluator for third-party applications. In this scenario, the Authorization API is not used seamless as in the case of WebSEAL and NetSEAL. Third-party applications only use the Authorization API calls and libraries to pass authorization requests to the Authorization Server for processing.

The API includes the functions listed in the following tables, Table 38 through Table 42. They are provided for your convenience and overview. For a complete list and detailed information, refer to the *IBM SecureWay Policy Director Programming Guide and Reference* that can be downloaded from:

The following functions are used to manipulate attribute lists:

Table 38. Attribute list functions

Function	Description
azn_attrlist_add_entry() azn_attrlist_add_entry_buffer() azn_attrlist_create() azn_attrlist_delete() azn_attrlist_entry_get_num() azn_attrlist_get_entry_buffer_value() azn_attrlist_get_entry_string_value() azn_attrlist_get_names()	Managing attribute lists

The following functions deal with credentials:

Table 39. Credentials functions

Function	Description
azn_creds_combine()	Creating a chain of credentials
azn_creds_create()	Obtaining user authorization credentials
azn_creds_delete()	Releasing allocated memory
azn_creds_for_subject()	Obtaining a credential from a chain of credentials
azn_creds_get_attrlist_for_subject()	Obtaining an attribute list from a credential
azn_creds_get_pac()	Converting credentials to a transportable format
azn_creds_modify()	Modifying the contents of a credential
azn_creds_num_of_subjects()	Determining the number of credentials in a credentials chain
azn_id_get_creds()	Obtaining user authorization credentials
azn_pac_get_creds()	Converting credentials to the native format

The following functions are used for authorization decisions:

Table 40. Functions for authorization decisions

Function	Description
azn_decision_access_allowed()	Obtaining an authorization decision
azn_decision_access_allowed_ext()	

The following functions are for initialization, shutdown, and error handling:

Table 41. Functions for initialization, shutdown, and error handling

Function	Description
azn_error_major()	Status codes and error handling
azn_error_minor()	
azn_initialize()	Initializing the authorization service
azn_release_buffer()	Releasing allocated memory
azn_release_string()	
azn_release_strings()	
azn_shutdown()	Shutting down the Authorization API

The following functions are API extensions:

Table 42. API extensions

Function	Description
azn_util_client_authenticate()	Logging in using a password
azn_util_password_authenticate()	Obtaining an identity for a user
azn_util_server_authenticate()	Logging in using a DCE keytab file

8.4 Documentation

Toolbox documentation is available via the Web from:

<http://www.ibm.com/software/security/firstsecure/library>

Part 3. Deploying IBM SecureWay FirstSecure

Chapter 9. Installation planning and considerations

Installation and configuration of the IBM SecureWay FirstSecure building blocks normally starts in a test environment where the detailed features are being evaluated, skills are being acquired, and some training takes place. Such a test environment is typically a relatively small installation that does not reflect a real scenario. Installation instructions for the building blocks can be found in the *Up and Running* guides that come with the product or can be downloaded from the corresponding library sections of each building block at:

<http://www.ibm.com/software/security/firstsecure>

This chapter does not repeat the information printed in the *Up and Running* guides, though brief prerequisite lists and installation steps are described for your convenience. This chapter does, however, give some additional information that you might take into account when planning for IBM SecureWay FirstSecure installation. Some considerations are also discussed that apply to certain environments. Some issues cannot be clearly answered, and guidelines cannot be given as they largely depend on individual organizations' policies, network layout, budget restrictions, expected load, availability requirements, and other hardware and software involved.

It is very important that you check the URL given above and any README documentation that comes with the product before any actual installation takes place for any important, last-minute, information and for exact levels of prerequisite products.

It is also a good idea to check the ITSO Web site for latest redbooks and redpieces about general concepts and individual components relating to IBM SecureWay FirstSecure. The ITSO Web site can be accessed at:

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

9.1 Pre-installation considerations

A complex computing framework, such as IBM SecureWay FirstSecure, makes demands on its platforms and environment. While the underlying operating systems are not part of the FirstSecure framework, their optimal configuration should, nevertheless, be considered to be very important for the quality and security of the final installation.

In the following sections, a few general hints are given for securing the platform on which IBM SecureWay FirstSecure is to be installed.

9.1.1 Hardening the operating systems

As mentioned earlier, off-the-shelf operating systems are configured for providing high and robust functionality. This is, in some cases, in contradiction to the requirements of a secure approach that should only provide the minimum of services that exactly match the functional demands. Therefore, prior to installation and configuration of the security applications, all involved servers should be *hardened*. This means de-installing or disabling all operating system features that could increase the chances of an intruder to break into the system.

We can only give some hints as to what services to disable because it depends on the network the machine is connected to (Internet or secured net), on your security policy (do you use unencrypted authentication, such as Telnet, within your secured network?), and last, but not least, on the requirements of the applications that run on the given machine (it might introduce new users or require RPC (remote procedure call) services).

For an overview what the IBM eNetwork Firewall 3.1 installation script does, please refer to the IBM Redbook *Protect and Survive Using IBM Firewall 3.1 for AIX*, SG24-2577, section 4.3.2.

For instructions on how to harden Microsoft Windows NT 4.0, you might lookup the IBM Redbook *Guarding the Gates Using IBM Firewall for Windows NT 3.2*, SG24-5209, sections 5.5.2 to 5.5.4.

For Sun Solaris, the procedure would be similar to AIX.

In either case, you should stay actual with all security related fixes.

9.1.1.1 Checking network services

For a quick check of the network services that a computer provides, you should, on AIX, invoke:

```
netstat -a -f inet
```

On Windows NT, on a command line, invoke:

```
netstat -a
```

This gives you a list of active services on a particular machine. Some of them might not actually be necessary (or, in fact, might pose a security hole). In both cases, you should ignore the lines that start with `tcp` (or `tcp4` or `tcp6`) but do not show `LISTEN` or `LISTENING` as their state.

You could use the Network Security Auditor (NSA), which is part of the IBM SecureWay Firewall distribution, to scan server systems regularly.

9.1.1.2 Checking local integrity

With machines where untrusted users have access to, or might have control over, program execution (for example, Web servers), you should also check for services that are only locally available. In principal, the number of processes should be low; the users should have as few privileges as possible.

On AIX, use the command:

```
ps -fade
```

to see a list of all active processes. To look for programs that are installed with the set user or group ID bit, type the following:

```
find / -type f \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \;
```

To display all world-writable files and directories, type:

```
find / ! -type l -perm -o+w -exec ls -ld {} \; | egrep -v '^c'
```

During the AIX installation process, you should consider using the trusted computing base (TCB). Then, you should set up auditing to get all interesting operating system information logged.

For Windows NT, check the Services window from the Control Panel. Only a few services should startup Automatic, and those services should be well-known services.

9.1.2 Security basics

Before implementing advanced security techniques, such as IBM SecureWay FirstSecure, you should make sure that basic security rules are in place, especially on servers that are located in the DMZs:

- Passwords – Make sure that passwords are enforced to be of a minimum length (typically six to eight characters) to contain at least one numeric character, to be different from the user ID to which they belong, and to be changed at specified intervals.
- User IDs – Make sure that every user (human and system users) has a password and that users are locked out after several logon attempts with wrong passwords (typically, three to five attempts). Keep the passwords to superuser accounts (root, supervisor, administrator, maint, etc.) among a very limited circle of trusted system, network, and security administrators.
- System defaults – Make sure that default user IDs are either disabled or have passwords that adhere to the minimum requirements stated above. Likewise, make sure that only those services are enabled that are required for a system to fulfill its designated role.

- Physical security – Make sure that access to the locations where critical systems and users physically reside is controlled appropriately. A receptionist might be as important as the corporate firewall. This includes access to backup media and networks. Data that might be well protected on a system may easily be intercepted on the network during a remote system backup operation.

9.1.3 Time synchronization

In an installation consisting of two or more computers, it is good use to have them all use the same time information. This is especially important for heterogeneous environments where log data from different sources are gathered. This applies also to the IBM SecureWay FirstSecure framework. Consider that you have to analyze some suspicious event where a couple of machines using different log mechanism are involved. If the timestamps of the logs are not consistent, it might be very difficult to reconstruct the exact sequence of the traces, thus, making it hard to understand what really happened.

A proven time protocol for complex network structures, as they are expected with FirstSecure, is the Network Time Protocol (NTP) because it allows an administrator to configure machines as time servers, time clients, and even time gateways.

Obviously, it would be advantageous not only to have all machines synchronized, but also have them use the correct (absolute) time. This can be achieved with either coupling the NTP servers to some external time server (that could be a time server at your Internet provider) or to equip a machine in the secured network with a radio clock device. Be alert, however, that the latter might be another source of troubles. It could be easy for an attacker to build a small radio transmitter that changes the time on the systems. A radio-controlled clock should always incorporate some feasibility tests and should not allow sudden changes to the time.

The Distributed Computing Environment (DCE) that is used as a security foundation for the IBM Policy Director also provides a time synchronization service, the Distributed Time Service (DTS). It can be run for time synchronization on any machine in a DCE cell. Its advantage over other protocols is that it can adjust clocks gradually, rather than abruptly. Please refer to the online documentation that comes with DCE for further information on DTS and how to configure it.

9.1.4 Low-level logging

Parts of the IBM SecureWay FirstSecure framework generate log data for some low-level events, such as IP packets not belonging to any permitted data connection. Sources for this kind of log data are firewalls and intrusion detection.

If you use filtering routers, they will generate log data for the same level of events, and these messages should also be gathered and archived so that you can get a comprehensive picture of what is going on in a network. Such events are of special interest in any DMZ, as they may indicate a potential attack.

Filtering routers should be configured (especially on the Internet) to forward their log data to a machine that accepts *syslog* datagrams for logging and, if necessary, later analysis.

9.2 Policy Director

Policy Director is made up of several components including:

- The Management Server
- The Security Manager with WebSEAL and NetSEAL
- The Authorization Manager
- The User Registry
- The Management Console
- The Directory Services Broker (part of the Management Server)
- The Credentials Acquisition Server, CAS (automatically installed)
- NetSEAT

In addition:

- A DCE cell, including a DCE Security Server and a Cell Directory Server (CDS), is required.
- DCE (client access) is required as a supporting security framework on each machine running one or more Policy Director servers.
- The User Registry may be held in an LDAP directory or in the DCE registry. If held in an LDAP directory, an LDAP server is needed.

As can be seen from this list, there are a number of possible components to consider before beginning installation. It is possible to put an entire Policy Director setup on one machine (including all DCE and LDAP services), but this is not recommended in a production environment. It would be more usual to spread components over several machines. We will discuss this further later in this section.

9.2.1 First steps

The first task to carry out when considering the production environment is to decide what should be installed where. This will depend on the overall environment, the business needs, what way other components of FirstSecure and any other security products are set up, and so on. This planning stage will take some time owing to all the other factors to be considered. For example, will the Security Manager (WebSEAL/NetSEAL) reside in a DMZ? If so, will there be just one, or will there be several? If more than one is being installed, what mechanism will be used to load balance between them?

If you have just received Policy Director and have never installed it before, you may want to set up a test/demonstration system to try out the installation of each component to help you understand better what is involved in the process.

We will first look at some general considerations when installing Policy Director. We will then look at a few specific points associated with installing Policy Director in different environments.

The first thing to consider is the products that are needed to support Policy Director, that is, DCE and, optionally, LDAP.

9.2.2 Supporting software: DCE and LDAP

As well as having appropriate operating systems, Policy Director requires some other products to be installed (depending on overall configuration decisions). This section will outline two of the products required. It is not a comprehensive or exhaustive description. For complete information, refer to the product documentation named here and in Appendix E, “Related publications” on page 353.

Both DCE and LDAP (if you selected LDAP as the User Registry) must be installed and running prior to attempting to install Policy Director.

9.2.2.1 Distributed Computing Environment (DCE)

Policy Director requires a working DCE environment in order to function. DCE provides secure communications between all the components of Policy Director. A group of DCE machines working together and administered as a unit is called a *cell*. A DCE cell is a logical group of networked computers that are managed as one unit. There are two kinds of nodes in a DCE cell:

- DCE client, or user machines, are general-purpose DCE machines. They contain software that enables them to act as clients to the DCE services.

- DCE server machines are equipped with special software enabling them to provide one or more of the DCE services. Every cell must have at least one of each of the following servers in order to function:
 - Security Server
 - Cell Directory Server

The former is the same Security Server referred to throughout Chapter 4, “The Policy Director (PD)” on page 53. The Security Server can be replicated within the cell to provide fault tolerance and scalability.

Each Policy Director server needs a DCE client (or NetSEAT on Windows NT) installed to interact with the DCE cell. Note that this is true even where all of DCE and Policy Director are being installed on one system. There will need to be at least one DCE server to provide the services to these clients.

There are different operating system and DCE requirements for the Policy Director servers (Management Server, Security Manager, and Authorization Server), the Management Console, and the NetSEAT client.

DCE is provided with Policy Director on the IBM Policy Director Security Services CD-ROM. The CD also contains *Quick Beginnings* guides for DCE on AIX and NT, an *Installation and Configuration Guide* for DCE on Sun Solaris, and several README files. You should refer to these documents for installation instructions and further information.

A later section in this chapter, 9.2.3.1, “Operating system and DCE versions” on page 225, lists the latest known combinations of Operating System and DCE levels to support Policy Director.

9.2.2.2 Lightweight Directory Access Protocol (LDAP)

If LDAP is to be used for Policy Director, then LDAP 3.1.1 must be installed. It is provided with the IBM Policy Director installation media. A short general description of LDAP is included in Appendix B, “Introduction to LDAP” on page 321. This section of this chapter is intended only to point out some of the LDAP dependencies and introduce two tools that will be used for administering Policy Director.

If you intend to use LDAP as your user registry, you will first need to consider where it will be installed and what it will be used for. The same LDAP installation can be used to store information for Policy Director, PKI (Trust Authority), the IBM Firewall, and many other applications that are LDAP-enabled. LDAP is recommended for scalability and flexibility.

LDAP is, as its name suggests, a protocol for accessing a directory. In fact, the IBM SecureWay LDAP directory uses a DB2 database underneath to provide performance and scalability. Therefore, it will require DB2 to run, which is included in the product.

In addition, the IBM SecureWay Directory provides a Web-based Administrator Graphical User Interface (administrator GUI) that includes online help for the administrator. From the administrator GUI, the administrator can:

- Set up the directory
- Manage day-to-day operations of the server:
 - Start up and shut down the directory server
 - Create, back up, and restore databases
- Manage access control lists
- Manage group membership
- Manage security levels (encryption options, server certificate management)
- View or change:
 - General server settings
 - General database/backend settings
 - Performance tuning options
 - Directory server activity
 - Directory replication configuration

The Directory Management Tool (DMT) provides a graphical user interface that enables you to manage information stored in directory servers. Use the tool to:

- Connect to one or more directory servers via SSL or non-SSL connections
- Display server properties and rebind to the server
- List, add, edit, and delete schema attributes and object classes
- List, add, edit, and delete directory entries
- Modify directory entry ACLs
- Search the directory tree

These tools are essential for setting up LDAP for use by Policy Director and provide extremely convenient facilities for maintaining the LDAP directories. In order to provide these, LDAP relies on a Web server and Web browser being installed.

DB2 and the Web Server must be installed and running before installing LDAP (see the installation instructions that come with the IBM SecureWay Directory). Additional information can also be found in the following IBM

Redbooks: *Understanding LDAP*, SG24-4986, and *LDAP Implementation Cookbook*, SG24-5110.

9.2.3 Policy Director prerequisites

Policy Director runs on one of three platforms:

- AIX Version 4.3.1 or higher
- Windows NT Server Version 4 with Service Pack 4 or higher
- Sun Solaris Version 2.6

The only exception is the NetSEAT client, which is supported on one of the following platforms only:

- Windows NT Server Version 4 with Service Pack 4 or higher
- Windows 98
- Windows 95

In addition to the DCE and LDAP requirements listed, the Management Console also requires Java Runtime Environment (JRE), which is bundled with the console package.

9.2.3.1 Operating system and DCE versions

The requirements for the operating systems and matching DCE versions are listed in the following for Policy Director Servers, the Management Console, and the NetSEAT client.

Servers

Operating system and DCE (one of the following):

- Sun Solaris 2.6 / Transarc DCE 2.0
- AIX 4.3.1 / IBM AIX DCE 2.2
- Windows NT 4.0 Server with Service Pack 4 / NetSEAT 3.0

On Windows NT, a Policy Director Server must have NetSEAT installed as its DCE lightweight client, and NetSEAT also needs to have a full DCE infrastructure, such as IBM DCE 2.2, in order to provide DCE services to the applications. If you are installing DCE on Windows NT, you will need to install it on an NTFS file system.

On Solaris, the Transarc DCE remote administration features must be enabled prior to installing Policy Director.

For detailed information of any required AIX PTFs or Solaris patches that DCE may need, refer to DCE documentation or any README file.

Management console

Operating System and DCE (one of the following):

- Sun Solaris 2.6 / Transarc DCE 2.0
- AIX 4.3.1 / IBM AIX DCE 2.2
- Windows NT 4.0 Workstation or Server with Service Pack 4 / NetSEAT 3.0
- Windows 95 and 98 / NetSEAT 3.0

For NT, the Management Console must have NetSEAT installed as its DCE lightweight client, and NetSEAT also needs to have a full DCE infrastructure, such as IBM DCE 2.2, in order to provide DCE services to the applications.

For detailed information of any required AIX PTFs or Solaris patches that DCE may need, refer to DCE documentation or any README file.

NetSEAT client

Operating System:

- Windows NT 4.0 with Service Pack 4
- Windows 95 and 98

NetSEAT needs to have a Directory Services Broker (DSB) as its proxy to CDS installed on a machine that has a full DCE client installed. A DSB is automatically installed when the Policy Director Management Server is installed.

9.2.3.2 LDAP prerequisites

For your convenience, the requirements for each platform are listed here. As many of the requirements are common to all platforms, those are listed separately first. Each platform section then contains only what is unique to that platform.

All Platforms

For *SecureWay Directory client*:

- 32 MB RAM is recommended.

For *SecureWay Directory server*:

- A minimum of 64 MB RAM (128 MB is recommended) on AIX and NT; a definite minimum of 128 MB is needed on Sun Solaris.

A Web server must be installed and configured. The following is a general list suitable for most platforms. Levels of individual products may vary slightly by platform. Check the README files for the latest information for each platform.

- IBM HTTP Server 1.3.3.1
- Apache 1.3.3
- Lotus Domino Go 4.6.2.5 or later
- Netscape FastTrack Server 3.01
- Netscape Enterprise Server 3.5.1
- Microsoft Internet Information Server 3.0 (Windows only)
- Lotus Domino 5.0 Webserver (AIX and Sun Solaris only)

To Administer the SecureWay Directory Server:

- A frame-enabled browser that supports:
 - HTML version 3.0 or later
 - Java 1.1.7(1.1.8 on AIX) features including JDK 1.1 AWT events
 - JavaScript 1.2
 - The browser must be enabled to accept cookies.
 - The following Web browsers support these specifications:
 - Netscape Navigator 4.07 or higher (4.08 is recommended)
 - Netscape Communicator 4.5 or higher
 - Microsoft Internet Explorer (MS IE) 4.0 plus service pack 1 or higher
- DB2 Universal Database - Personal, Workgroup, or Enterprise edition (plus Extended Enterprise edition on AIX) - Version 5.2 with appropriate FixPak.

Notes:

- DB2 Version 5.2 with an appropriate FixPak is included with the IBM SecureWay Directory for the platform.
- Space requirements vary. Please see the README files for each platform for information about calculating space requirements.

SSL GSKit:

The Global Security Kit (GSKit) is an optional software package that is required only if Secure Sockets Layer (SSL) security is required. For SSL capability, the IBM GSKit 3.01 package needs to be installed on the system.

The SecureWay Directory V3.1.1 alone does not provide the capability for SSL connections from SecureWay Directory clients. The SSL feature is added by installing the IBM GSKit package.

The SecureWay Directory server works without the GSKit installed. In this case, it only accepts non-SSL connections from SecureWay Directory clients. Similarly, the SecureWay Directory client works without the GSKit installed.

Windows

For *SecureWay Directory client*:

- Microsoft Windows 95, Windows 98, or Windows NT 4.0 with Service Pack 3 or higher; NTFS file system is required for security support.

For *SecureWay Directory server*:

Windows NT 4.0 with:

- Service Pack 3 (or greater)
- An NTFS partition

AIX

For *SecureWay Directory client*:

- AIX Operating System Versions 4.3.X. See the README (server.txt) file for information on supported versions of AIX.

For *SecureWay Directory server*:

In addition to the client requirements, the server requires the following:

- Java JDK (included with the AIX CD-ROMs)

Sun Solaris

For *SecureWay Directory client*:

- Solaris Operating Environment software Version 2.6. See the README (client.txt) file for the latest information on supported versions of Sun Solaris.
- Ensure that the code page conversion routines (en_US.UTF-8 1.0) are installed.

For *SecureWay Directory server*:

See the README (server.txt) file for the latest information on supported versions of Sun Solaris.

9.2.4 Installing Policy Director

Once you have DCE and (optionally) LDAP up and running, there is a comprehensive guide, the *IBM SecureWay Policy Director Up and Running Guide* (included with the product), which will take you step-by-step through the job of installing Policy Director. Therefore, we will not describe the installation process in all its details in this chapter.

From the point of view of installation, Policy Director essentially consists of three servers, a Management Console, and the NetSEAT client. Although there is more to it than this, it is helpful to keep this in mind when installing.

9.2.4.1 Installation Overview

The installation of Policy Director, in essence, consists of the following steps:

1. Determine which components of Policy Director are to be installed on which systems.
2. Verify that the system(s) are properly configured to support Policy Director and that all prerequisites are installed, configured, and running where they are needed.
3. Determine which system will run the Management Console.
4. Log on to the machine you will be installing on as a user with administrative privileges. UNIX users will need full root privileges.
5. If you are using LDAP, you will need to customize it a little before installing the Policy Director. Part of your planning should include deciding what customizing of LDAP needs to be done. This will optionally include setting up SSL communications between the Policy Director servers and LDAP if you want the security of SSL communications.
6. Install the Policy Director Server components. On the Windows platform, installation and configuration take place in the same step. On the AIX platforms, installation and configuration are typically two separate stages. On Sun Solaris, configuration is carried out at the same time as each package is installed.

If you are installing on AIX, configure the Policy Director Server components.

7. After installing the Policy Director Server components, there is some more configuration work to be carried out in LDAP. This includes giving the Policy Director security group daemon (or NT service) full control of the relevant LDAP suffixes.
8. Configure the CAS if you require CAS services, such as client-side X.509 certificate support.
9. Install the Management Console(s).

Please refer to the *Up and Running Guide* and any README documents for detailed instructions. This guide also contains instructions for setting up SSL between Policy Director and the LDAP directory server.

9.2.4.2 Policy Director installation: General considerations

The following is a list of considerations for installation, regardless of the environment being installed into.

There is a very useful table of installation scenarios and what must be installed in each case in the *IBM SecureWay Policy Director Up and Running Guide*. All servers need the base component, known as *IV.Base*. This is automatically installed with each server. All Policy Director Servers need a DCE client unless they are being installed on Windows NT, in which case, they require the NetSEAT client.

If LDAP is the User Registry, it will also be necessary to install an LDAP client on each Policy Director server machine. The LDAP server must be installed, configured, and running prior to attempting to install and configure Policy Director. If you are installing the first or only server in the secure domain, you must install the Management Server. The Management Console is easier to use if it is displayed on at least a 17-inch monitor with 1024x768 pixels resolution (or better).

The Authorization Server is installed on each machine where a custom application will be using the Policy Director authorization service. This server also provides the Authorization API for applications to make fine-grained authorization policy decisions.

If developing applications with the Policy Director Application Development Kit (ADK), keep the following in mind:

- It requires Policy Director package.
- IVAuthADK is installed on application machines.
- NetSEAT or a DCE client is required on application machines.
- The Authorization Server must be installed in the secure domain.

Configuration of the Policy Director Components is order dependent. On AIX, for example, when you enter the SMIT screen (SMIT fastpath *int_menu*) to configure the Policy Director, the components will be listed in the order they should be configured. (Unfortunately, the same SMIT panels do not allow you to review the settings once the configuration was made.) However, these can be viewed, and small changes can be made (with care) in the configuration files: *iv.conf*, *secmgrd.conf*, and *ivmgrd.conf*. These configuration files are briefly discussed in 9.2.6, “Managing server configuration files” on page 236.

9.2.4.3 Installing on a single system

As mentioned previously, it is recommended that, for most practical purposes, Policy Director should be installed over multiple machines. This will give many

benefits of reliability, scalability, and performance, as well as allowing for the installation of different servers in different areas of the network (see 9.2.4.5, “Installing with other products” on page 233, for an example).

You may, however, wish to install all the components on one machine, for example, for testing and training purposes.

For the purpose of writing this redbook, Policy Director and all the prerequisites were installed on one system. For your convenience, some of the major characteristics of this system are listed here. Note that this does not imply any suggestion or recommendation. This is simply a report of what worked in this test environment. In a different environment, the system needs may vary significantly:

Machine Type	7025-F50
Operating System	AIX 4.3.3
Memory	512 MB
Disk Space	17 GB – However, only about 3 GB are actually used by all the components. Remember that this is before adding users, resources, ACLs, and so on, to the database.
Paging Space	1 GB – Following the rule of using two times the amount of memory.

Such an installation provides good performance for evaluation, testing, and training.

One point to consider on a single system installation is the fact that a Web server is running to serve the LDAP administration GUIs, and WebSEAL will be running after installation. Both servers, by default, use the same port 80, which causes a conflict that needs to be solved by configuring either server to run on another port.

9.2.4.4 Installing on multiple systems

This is the recommended way to install Policy Director in a production environment rather than installing it all on one machine. This section will briefly list some points to take into consideration.

You may also want to set up a test/demonstration system spread across several machines prior to planning a production environment, in order to better understand the possibilities for fault tolerance and scalability. Indeed, this may be essential if you are exploring the deployment of Policy Director with other FirstSecure components or with third-party security products.

As mentioned previously, *IV.Base* and a *DCE client* are required on every machine that will hold a Policy Director server. *IV.Base* is installed

automatically with each server, but the DCE client is not. On a Windows machine, NetSEAT is the DCE client. Additionally, if you are using LDAP, the Management Server, the Security Manager, and the Authorization Server will all require an LDAP client.

If you are installing the first or only server in the secure domain, you must install the Management Server. You must only install one Management Server in each secure domain. Two Management Servers cannot coexist in the same secure domain.

All other Policy Director Servers can be installed multiple times to provide additional reliability and scalability. If you are using LDAP, you only need to configure it prior to installing the first Policy Director server (which will always be the Management Server). The basic configuration tasks for LDAP do not need to be done for each server, no matter how many times each of the other servers in the same secure domain are installed. There is, however, some server specific information contained in the LDAP directory.

The Security Manager has three major parts:

- *IV.Net* is always installed
- *IV.Web* is the WebSEAL component
- *IV.Trap* is the NetSEAL component

WebSEAL and NetSEAL can be installed on the same or different machines, but each of the machines will always require IV.Net.

Where multiple DCE servers are installed, the order in which the servers are accessed can be configured in NetSEAT. This allows for optimization of NetSEAT's access to the DCE servers in the secure domain.

NetSEAT can also be configured to recognize replicated front-end Policy Director servers, for example, WebSEAL. This allows for load balancing across the servers.

The Management Console also requires a DCE client, or, if it is on a Windows machine, the NetSEAT client.

The Directory Services Broker (DSB) is a DCE directory client proxy required by any NetSEAT and the Management Console. At least one DSB is required within the DCE cell. The first DSB is automatically installed with the Management Server. The DSB is installed on a server machine and is shared by many clients. Multiple DSBs can be installed in a cell to provide high availability and load balancing, but additional DSBs need to be installed manually.

The Credentials Acquisition Server (CAS) is automatically installed with IV.Net, the base part of the Security Manager.

9.2.4.5 Installing with other products

There are some dependencies to consider in a typical Policy Director installation. This is even more so when installing alongside other products, such as firewall and Trust Authority, and when considering whether to use LDAP or DCE as the User Registry.

Most installations will not be using Policy Director as a stand-alone product. In the majority of cases, there will be a firewall and associated products (such as IBM FirstSecure SecureWay Boundary Server). It is important to understand the interrelationships between all these products before beginning the installations. In this section, we will point out a few of these and, where possible, offer some guidance.

Firewall considerations

Policy Director is being implemented as part of a security solution. The company implementing it wants to keep secure all resources that could be vulnerable to attacks from outside. These resources are usually protected behind a firewall. The firewall prevents access by unauthorized users or processes. The firewall has to be carefully configured to allow legitimate access and prevent access by likely attackers. It is usual to have a firewall, known as a demilitarized zone (DMZ), protecting a group of resources. The DMZ is an area considered safer than just having a machine open to the Internet but not as safe as being fully behind the firewall.

In the most secure solutions, there are at least two firewalls with a DMZ in the middle. Figure 68 illustrates this:

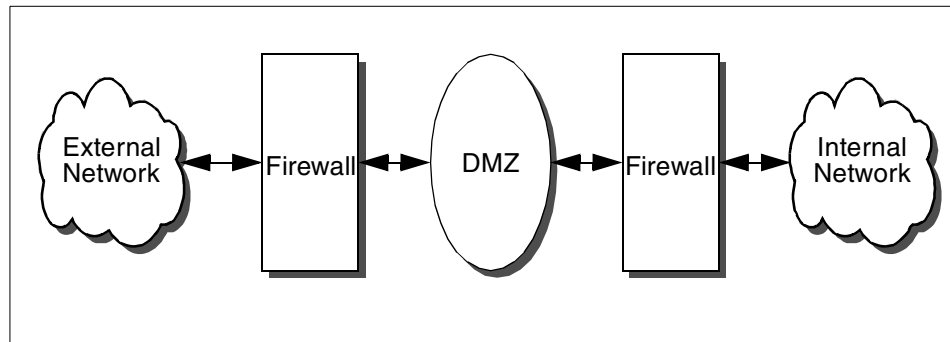


Figure 68. A secure firewall/DMZ solution

One of the resources that needs to be secured is Policy Director itself. If a hacker could get access to the Management Console, for example, he or she could set themselves up with maximum authority access to other systems. Therefore, the Policy Director resources really need to be in the intranet (the secure internal network) where it is highly unlikely that an external attacker can reach them.

But, Policy Director, and especially the Security Manager, is the entry point and the authentication/authorization agent for the whole IT infrastructure. If WebSEAL and NetSEAL sit inside the secure intranet, this means that user (client) requests have to penetrate the firewall before they are authenticated. Surely this defeats the whole object of having a firewall and a Security Manager in the first place.

Clearly, what has to happen is that the Security Manager and its agents - WebSEAL and NetSEAL - have to sit outside the firewall that guards the main intranet since users must be authenticated and authorized before being allowed to cross the firewall. However, this leads to another set of considerations. The firewall is now sitting between a client and its server. The DCE client that is providing the communications between the Security Manager and the Security Server and the Security Manager and the Management Server. Optionally, the firewall is also now sitting between an LDAP client and the LDAP server. The LDAP client, if installed, is giving the Security Manager access to the User Registry on the LDAP database.

The firewall will have to be configured to allow the clients and servers to communicate with each other. DCE presents some special challenges.

Even a basic DCE client interacting with the DCE core servers in a DCE cell involves TCP/IP traffic over several TCP and UDP port numbers. This means that the firewall filter rules will have to open more than just one or two ports in the way that filter rules are set up for FTP and TELNET. This problem is exacerbated because, in DCE, although whoever requests the connection (server or client) does so from a narrow range of ports, once the other party responds, they may do so over a wide range of port numbers. Also, the port numbers used by DCE, apart from port 135, are not well-known ports since they are dynamically allocated above 1024. The aim is to open up the minimum number of ports through the firewall to enable the DCE to run.

In order to have some control over the dynamic ports that DCE services use (even for the base services, such as a DCE login), it is necessary to set the environment variable `RPC_RESTRICTED_PORTS` before starting DCE services. This needs to be done on all the DCE server systems inside the cell that our DCE client outside the firewall is likely to communicate with. This will

include at least the CDS server and the Security server, but there may be others depending on your environment. Here is an example of how to set this variable:

```
RPC_RESTRICTED_PORTS=ncadg_ip_udp[6500-6550]:ncacn_ip_tcp[5000-5050]
```

This system environment variable tells DCE to use only ports 6500-6550 for UDP, and 5000-5050 for TCP protocols, respectively. For this example, we used the variable as shown above with a range of fifty ports for each protocol type, although in a larger production DCE environment, these range sizes are likely to be too restrictive.

The main implication of having multiple Policy Director servers (for example, WebSEAL) in one secure domain is that the DCE ports will need to be opened up on the firewall for each DCE client involved.

NetSEAT and the firewall

NetSEAT is a DCE client, amongst other things, and, as such, its ability to connect to the secure domain across the firewall has to be considered the same as a normal DCE client. Furthermore, for each Policy Director server it will connect to, the user can choose, when configuring NetSEAT, to use either TCP or UDP. Attention needs to be given to choosing the correct configuration in NetSEAT. For example, you may want to de-select UDP access entirely and specify the correct TCP port numbers for the firewall.

LDAP and the firewall

If an LDAP directory was chosen for the User Registry, and that LDAP server is behind a firewall, the firewall must be configured to allow LDAP traffic to pass. In the scenario shown in Figure 68, the LDAP directory server will most likely be in the secure network, while the Policy Director Security Manager resides in the DMZ. Therefore, the firewall on the right in Figure 68 will need to be transparent for LDAP. LDAP uses the default port 389 (or 636 if SSL is being used), which must be open through the firewall.

There is another interaction between Policy Directory and the firewall. As mentioned earlier, firewall proxy users can be administered by Policy Director. If this is the case, proxy users are stored in the LDAP User Registry (this is not supported on a DCE User Registry) from where the firewall retrieves that information. The firewall must be configured to use LDAP via the SBS Wizard in order to take advantage of integration with Policy Director.

If LDAP is not used, Proxy users are defined by the firewall GUI only. Such users cannot be used by the SurfinGate plugin or managed by the SecureWay Policy Director. Also, this does not provide centralized directory

storage of users. You may have to keep more than one database of accounts for the same users.

Trust Authority and LDAP coexisting

Trust Authority also depends on having a DB2 database installed. However, its prerequisites may, at any point in time, be slightly different to LDAP, and you will need to plan carefully if you wish to avoid having two separate DB2 databases. There may, of course, be good reasons to have two DB2 databases anyway. Using one database will cut down administrative overhead, but having two could lead to better overall reliability and scalability and could ultimately provide better performance.

9.2.5 Compatibility, portability, and interoperability

Several elemental features of Policy Director will enable portability to various computing environments.

Compatibility and migration

Policy Director R2 (subject of this book) supports all the functions in Policy Director R1 for backward compatibility. A migration utility is provided in Policy Director R2 to assist administrators to migrate their user data and GSO target data from the DCE Security Registry to the LDAP server.

Code portability

The Policy Director Management Console is written all in Java to ensure maximum portability.

DCE interoperability

Any two components of Policy Director on different platforms can communicate with each other as a result of the platform interoperability provided by DCE.

LDAP server interoperability

Policy Director supports the IBM SecureWay Directory Server. The Management Server has a framework to work with different directory services and databases; so, all the supported functions and interfaces do not have to be changed because an underlying directory or database changes.

9.2.6 Managing server configuration files

There are three key configuration files that Policy Director uses to configure the underlying system processes. Although these files have comments and default values, we will highlight a few of the key stanzas and parameters that are most useful. These files have the same format on each operating system that Policy Director is implemented on.

9.2.6.1 **iv.conf**

This file is similar to any HTTP server configuration file. The main sections of this file are used for defining:

- General Policy Director Configuration
- WebSEAL server configuration
- HTTP client configuration
- HTTPS client configuration
- Server operation
- Server files and directories
- Logging
- Auditing
- User directories
- Symbolic links
- Directory indexing
- Document caching MIME types
- Content encoding
- Plugins

9.2.6.2 **ivmgrd.conf**

This is the configuration file for the *ivmgrd* service (the Management Server main process). Amongst the parameters it defines are:

- Definitions of the DCE environment
- Naming the location of the master authorization database
- Contains information about the LDAP server configuration
- Defines the third party application object spaces

9.2.6.3 **secmgrd.conf**

This file manages the parameters for the *secmgrd* daemon. It specifies such parameters as TCP and SSL ports for listening on. It also specifies a parameter called Server RPC UUID. This parameter may need changing if more than one *secmgrd* service is running on one machine.

9.2.7 Testing the installation

The following tests can be performed to determine that the system is working correctly:

- Basic HTTP Web page retrieval. Pointing the browser at the port on which the WebSEAL server is listening should produce an easily identifiable Policy Director welcome page.
- Starting the Management console. Entering data should not cause any problems:

- Creating and saving users and groups
- Creating resources
- Copying a default ACL and editing it
- Authorization-challenging tests. For example, issuing a Telnet request to a server protected by NetSEAL or an HTTP request to a Web server protected by WebSEAL.
- SSL tests, for example, issuing an HTTPS request to a secure Web server
- Junctioning another Web server by creating a Smart Junction
- Testing GSO over a Smart Junction
- Testing connections from a NetSEAT client

9.3 SecureWay Boundary Server

The following sections describe what prerequisites and dependencies have to be considered when installing the SecureWay Boundary Server and its components, what basic configuration steps should be performed, and how its functionality can be tested. First, a brief introduction and a recap on firewall design leads the discussion of each of the components in the SecureWay Boundary Server.

9.3.1 Introduction

Before installing the SecureWay Boundary Server, the system administrator has to consider some aspects of the installation depending on the facts about the present and future network and the systems that will be running on it. He or she should consider the following items:

- What services are to be provided to the Internet?
- What services are to be provided to the intranet?
- What services are to be provided to the extranet or business-to-business network connections?
- Where are the servers going to be located?
- Are there different levels of security implied by the kind of user (for example managers, human resources personnel, development, engineering, or normal users) or the kind of server on the intranet?

In order to provide the network infrastructure for those services to be carried out safely, the system administrator will design a network keeping in mind these services and the ways he or she can secure them.

The administrator will need to set up different networks, security zones, demilitarized zones (DMZs), DNS servers, firewalls, HTTP proxies, mail servers, mail relays, and the like. The administrator also needs to consider where he or she locates those servers, and how will he or she defines the boundaries for these networks.

9.3.2 Basic firewall design

As a first step (if not already in place), a basic network and firewall design need to be in place. Please refer to Appendix C, “Basic firewall and name service design” on page 327 for some general guidelines on firewall and name service design.

A good picture of how such a design might look like is shown in Figure 69.

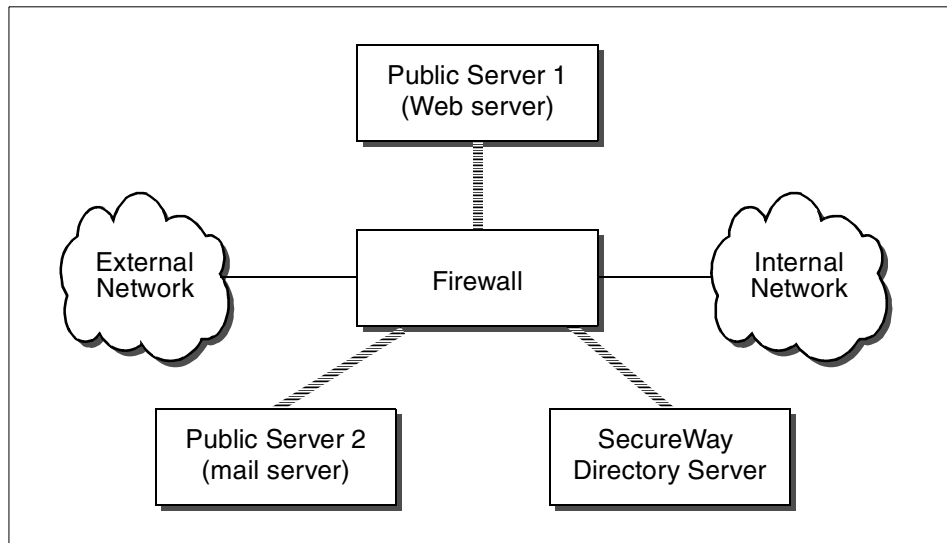


Figure 69. Sample firewall design

In this figure, we see two large networks attached to the firewall, each with different levels of security:

- The internal, secure network (intranet)
- The external, insecure network (we assume it is the Internet)

Also attached to the firewall are networks that are known as demilitarized zones (DMZs), and on these DMZs, there are servers that will provide services, such as e-mail and Web serving.

As discussed in to Appendix C, “Basic firewall and name service design” on page 327, you could also attach to some of the DMZs the utilities provided by SBS, such as WEBSweeper, MAILsweeper, ACE/Server, Norton AntiVirus, and SurfinGate. Note that in Figure 69 there are three distinct DMZs for three different classes of servers. The very basic DMZ layout uses one, some common installations incorporate two, and even more secured installations may incorporate three or even more separate DMZs.

9.3.3 IBM Firewall

As it would be beyond the scope of this book to provide detailed installation and configuration instructions, the following is an overview of the steps involved in setting up the IBM Firewall provided for gathering a general understanding. For a basic IBM Firewall setup on AIX or Windows NT, perform the following:

1. Ensure you have the prerequisites as listed on the latest firewall documentation.
2. Plan for the IBM Firewall setup. Decide in advance which functions of the firewall you want to use and how you want to use them. Refer to Appendix C, “Basic firewall and name service design” on page 327 for some general guidelines on firewall design.
3. Install the firewall code on the dedicated machine. Follow the instructions that come with the product and make sure to check any latest information.
4. Tell the firewall which of its interfaces are connected to secure networks. You must have a secure interface and a non-secure interface for the firewall to work properly. You can change the security status of an interface later.
5. If the firewall is going to be connected to the Internet, your Internet Service Provider (ISP) will provide you with a valid IP address for the Internet.
6. Set up the general security policy by accessing the Security Policy dialog in the System Administration folder.
7. Set up the domain name service and mail service. Efficient communication will not take place if you do not provide DNS resolution. Access these functions from the System Administration folder on the configuration client navigation tree.
8. Define key elements of the network(s) that are connected to the firewall using the Network Objects function in the configuration client navigation tree. Network objects control traffic through the firewall. Define the following key elements as network objects:

- Secure Interface(s) of the firewall
 - Non-secure Interface(s)
 - Secure Network(s)
 - Each subnet on your secure network(s)
 - A host network object for the Security Dynamics ACE/Server and the Windows NT domain servers, if appropriate
9. Enable services on the firewall. These are the methods (such as socks or proxy) by which users in the secure network can access the non-secure network. Which services get implemented depends on decisions made at the planning stage. Implementing a service often requires setting up some connections to allow certain types of traffic. For example, if you want to allow your secure users to surf the Web on the Internet by using the HTTP proxy, you not only need to configure the HTTP proxy daemon on the firewall, but you also need to set up connections to allow HTTP traffic.
 10. If you are running the firewall on Windows NT, then the installation process will disable NETBIOS. If you want to use a Windows NT domain password for authentication, you must configure the Windows client code that implements the ability to search trusted Windows NT domains for authentication purposes. The trusted Windows NT servers must have TCP/IP host names and addresses and have TCP/IP connectivity between them and the firewall. The firewall administrator needs to create connections between the firewall and the trusted Windows NT servers in order to permit traffic to flow between the two.
 11. If you will be using network address translation (NAT), first contact your ISP to obtain another registered Internet address for use with the many-to-one address translation. This address is in addition to the one we mentioned before. Check the documentation for NAT configuration procedures.

9.3.3.1 Installation of the IBM Firewall

The installation of IBM Firewall 4.1 is a two-stage process:

Note

For security reasons, you should never connect a firewall that has not yet been completely configured and tested to a non-secure network. At least the installation of the firewall software and the first basic configuration step should be applied before going “online”.

1. Install and configure the operating system and all required PTFs or service packs.

It is best to start with a fresh operating system install. That will assure you that unnecessary services are not installed. If the machine is going to also serve as a DNS server, be sure to install the TCP/IP server part of AIX.

Please refer to Appendix C, “Basic firewall and name service design” on page 327 and be sure to understand the issues on DNS setup before designing or setting up DNS servers.

2. Install and configure the IBM Firewall 4.1.

When installing the IBM Firewall 4.1, please keep the following points in mind:

- Be sure to read the documentation for the latest information on hardware and software requirements.
- Keep it simple and secure (KISS).
- Physically secure the firewall system(s) in a secured, locked area.
- Make a checklist of any changes to the configuration so that you can periodically check to make sure those settings are still the same.
- Consider how secure the network structure is between your firewall and the secure network.
- Run the minimum number of services necessary (remember: KISS).
- User IDs should be kept to a minimum.
- Additional IDs should be set up using the firewall interface.
- Remove any compilers, assemblers, or any other computer language that allow system calls.
- Remove tracing tools, for example, *tcpdump* and *iptrace*.
- Use the audit and logging functions to monitor the system.

9.3.4 Additional filesets

For the convenience of installation and administration, you might need to install the additional filesets shown below in Table 43 if the firewall runs on AIX.

Table 43. Additional AIX filesets

AIX filesets	Description
bos.net.tcp.server	TCP/IP server (DNS server and more)
bos.acct	Accounting service
bos.sysmgt.sysbr	System backup and bos install utility

Use `lslpp -l <fileset name>` to see if these are installed. Use `smit install_latest` to install them. Use the preview option in SMIT before every install and patch. It will help you identify problems before you actually begin modifying files.

Make sure that the latest security-related fixes for the operating system and any additional software being installed on the firewall machine are installed to avoid known problems and potential vulnerabilities. This also applies to the firewall software itself. Consult with IBM software support staff to get access to the latest fixes.

Apply these fixes before the firewall code is installed since the application of updates can de-harden or overwrite secure firewall components.

A safe way to install any PTFs or other fixes is to un-install the firewall software, install the updates, and then reinstall the firewall software. This assures that the firewall software does not get overwritten by any updates.

9.3.5 Basic configuration of the IBM Firewall

Because configuring the IBM Firewall IP filters is a relatively tangled task, it is good practice to prepare a worksheet describing, in the correct sequence, all required connections including anti-spoofing rules, deny-without-log of broadcasts, and so on. Then the rules and services that are not predefined should be created, the involved network objects should be created, and finally, the connections from the working sheet can be entered in reverse direction.

9.3.5.1 The Setup Wizard

The Setup Wizard aids you with the initial configuration of the firewall. It is especially helpful if you do not have an extensive knowledge of firewall configuration because it enables you to have a basic firewall configuration up and running quickly after installation.

The Setup Wizard appears automatically after you log on to the firewall for the first time. Thereafter, the Setup Wizard is available under the help menu item on the GUI. The Setup Wizard is optional; you are not required to use it to configure the firewall. The Setup Wizard guides you through the following fundamental tasks:

- Defining basic security policies
- System administration tasks having to do with interfaces, DNS, mail, and log setup

- Setup to allow secure users to access non-secure networks through the Web, Telnet, or FTP
- Creating an alert log
- Setting up some basic log monitor thresholds

The Setup Wizard can be helpful for getting started on a variety of firewall installations. However, depending upon your circumstances, the Wizard may not be recommended. The Wizard is not recommended for:

- Migrating a configuration from a previous version of firewall
- Setting up a demilitarized zone (DMZ) that involves designating two or more network interfaces as secure
- Setups that require more than one security policy for the secure networks

9.3.6 Proxy user administration through Policy Director

As part of the SecureWay Boundary Server, there is a component called the same, *SecureWay Boundary Server*. It is an add-on to the IBM Firewall that can be installed separately. This component provides the link for the IBM Firewall to an LDAP directory that contains proxy user and group definitions administered through Policy Director.

The SecureWay Policy Director can manage firewall proxy users if the firewall is installed with the SecureWay Boundary Server component installed and enabled. In this case, the firewall's proxy users are defined for the following firewall services:

- Telnet
- FTP
- HTTP
- Socks

The users and the associated policies are stored in an LDAP directory, which is used to store directory information in a central location for storage, updates, retrieval, and exchange. SecureWay Policy Director manages the firewall proxy users in the LDAP database.

The information that is needed to configure the IBM SecureWay Boundary Server for firewall is as follows:

- The host name and domain name of the IBM SecureWay Directory server that the firewall will use.
- The port number on which the IBM SecureWay Directory server is listening. The default port is 389.

- The SecurityMaster password for the IBM SecureWay Directory server.
- The domain name to use to distinguish the proxy users for this firewall. Any firewall using this name will administer the same set of users. Normally, you would use the fully qualified hostname of the firewall machine.
- The firewall administrator name used to access the proxy users stored in the SecureWay Directory. This name will be granted access to modify all proxy users created by SecureWay Policy Director.
- The distinguished name (DN) that the IBM SecureWay Directory uses as a root from which to start searching for firewall proxy users in the directory. This should be the suffix you created in SecureWay Directory to store firewall proxy users.
- A password for the administrator ID of the firewall to use when connecting to IBM SecureWay Directory server.
- A timeout value (in seconds) for how long to wait for a response from the LDAP directory.

The steps necessary to set up the SecureWay Boundary Server for firewall are explained in an example in 10.1.4, “Managing IBM Firewall proxy users” on page 283.

The system administrator will have to enable certain rules at the firewall level in order to allow the proper IP connections between the firewall, the Secure Boundary Server, and the LDAP database. The LDAP database server can be situated either on the internal “secure” network, or on a “secure” DMZ.

The connections that should be allowed between the firewall and the LDAP directory server (IBM Secureway Directory) are as follows:

- The source will be the involved adapter on the firewall, and the destination will be the LDAP directory server.
- The source port will be greater than 1023
- The destination port will be equal to 389 (or 636 if SSL is being used)
- The interface will be secure
- The routing will be local
- The direction will be outbound

As for the reply from the SecureWayDirectory server, the following connections should be allowed:

- The destination will be the secure adapter address of the firewall
- The source port is 389 (or 636 if SSL is being used)
- The destination port will be greater than 1023

- The interface will be secure
- The routing will be local
- The direction will be inbound.

You should then run the SecureWay Boundary Server Setup Wizard. Select the option to enable the firewall to work with SecureWay Policy Director.

9.3.7 SurfingGate

To prepare to use SurfingGate, you must have Windows NT Service Pack 5 installed. Ensure you comply with the latest hardware prerequisites that the shipped documentation mentions.

Perform the following to use SurfingGate:

- If you are using Oracle as the SurfingGate database, it must be installed and configured.
- If you are using a Windows NT firewall, you need to decide whether to use the proxy mode or the plugin mode.
- To enable SurfingGate plugin on IBM Web Traffic Express (WTE), install the SurfingGate plugin on the firewall machine and run the SecureWay Boundary Server wizard.

You need to create a connection to allow traffic to flow from the SurfingGate plugin to the SurfingGate server.

9.3.7.1 Configuring SecureWay Firewall to use SurfingGate plugin

The SurfingGate plugin is only supported on Windows NT. A connection will have to be created to allow the IBM Firewall to talk to the SurfingGate server. The SurfingGate server should be installed on the secure side of the firewall.

The connection is as follows for the outbound rule:

- The source will be the secure adapter address of the firewall.
- The destination will be the address of the SurfingGate server.
- The source port will be greater than 1023.
- The destination port will be equal to 3141.
- The interface will be secure.
- The routing will be local.
- The direction will be outbound.

The inbound rule is as follows:

- The source will be the address of the SurfingGate server.
- The destination will be the secure adapter address of the firewall.

- The source port will be equal to 3141.
- The destination port will be greater than 1023.
- The interface will be secure.
- The routing will be local.
- The direction will be inbound.

You will also need to configure the SurfinGate server to allow the data that will be scanned. On the SurfinConsole, you need to check the **Plugin mode** option under the General tab.

9.3.7.2 Configuring SBS to enable the SurfinGate plugin

This applies to Windows NT only. Select **Start->Programs->SecureWay Boundary Server** to bring up the SBS Wizard.

1. Select the option **Configure the Firewall HTTP Proxy to pass authentication information to the SurfinGate plugin.**
2. Complete the dialog.

9.3.7.3 Configuring SurfinGate

On AIX, you can only use SurfinGate as a chained proxy. On Windows NT, you can use it as a chained proxy or as a plugin for the firewall HTTP proxy.

As a chained proxy, your installation might be as shown in the following figure:

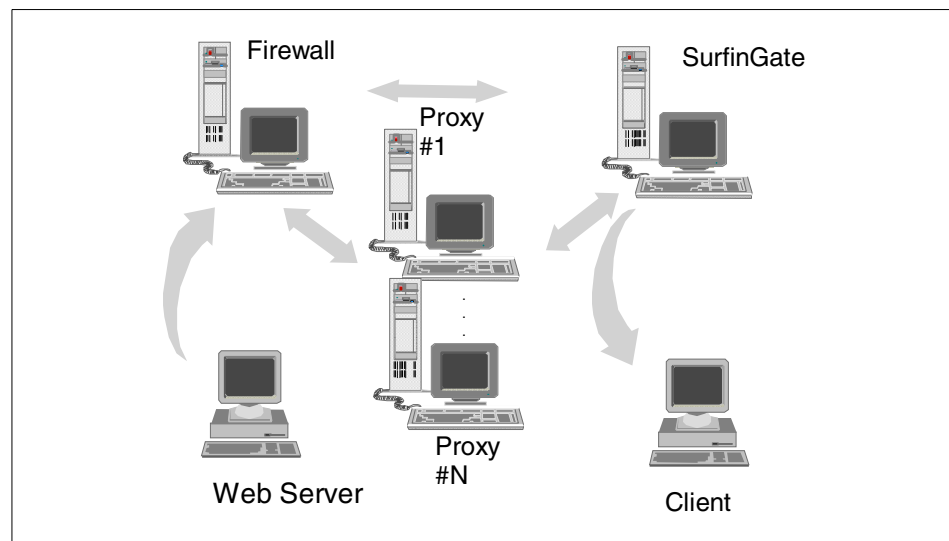


Figure 70. Chained proxy setup

The client Web browsers need to be configured to use SurfinGate as a proxy for HTTP, HTTPS, and FTP. Be sure to specify the port number on which SurfinGate is listening (default is 8080).

On the SurfinConsole, you will need to check the **Proxy Mode** option under the General tab. You should also enter the address and port number of the firewall's HTTP proxy in the Next Proxy field on the Proxy tab. Alternatively, if you have additional proxies already defined, you might point SurfinGate to them as the next proxy.

If you want to configure SurfinGate to work as a plugin for the firewall HTTP proxy, which is only supported on Windows NT, then the configuration is as shown in Figure 71.

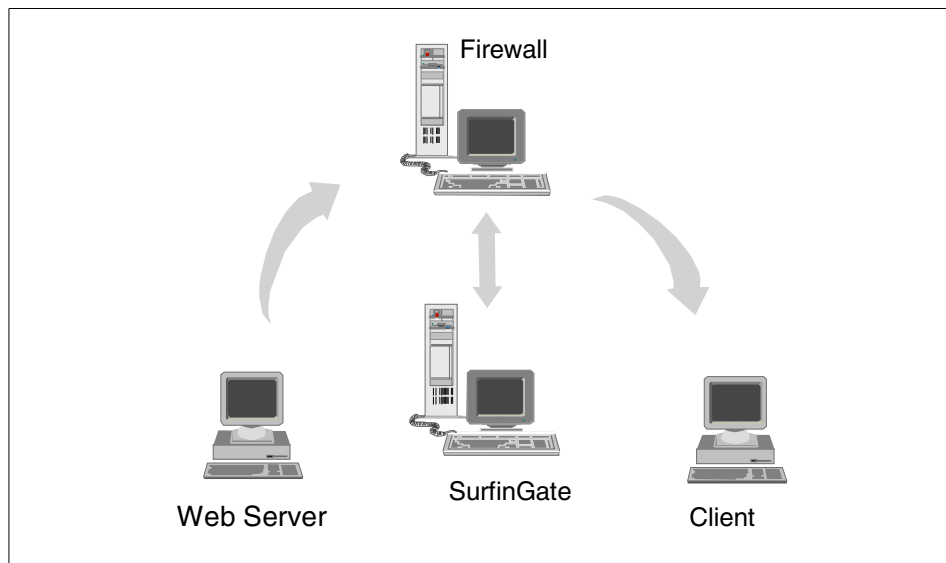


Figure 71. Plugin setup

The client Web browsers need to be configured to use the firewall HTTP proxy as the proxy for HTTP, HTTPS, and FTP. Specify the port number on which the firewall HTTP proxy is listening (default is 8080).

On the SurfinConsole, you will need to check the **Plugin Mode** option under the General tab.

9.3.8 Security Dynamics SecurID authentication

As mentioned earlier, an ACE/Server and two user licenses, including two *SecurID* tokens, are included with SecureWay Boundary Server. To install the ACE/Server, perform the following checks:

- Check on the documentation for the latest hardware and software requirements.
- If you are using a name server, such as NIS or DNS, the ACE/Server system requires that an ACE/Server's primary hostname (also known as its boot name) be the first name in any list of aliases for that machine. This is a requirement for both master and slave servers.
- The ACE/Server machine itself must be physically secure. Only trusted personnel should have physical access to the master and slave servers. Consider special utilities for doing secure Telnet and FTP sessions to that machine. Search the Web with your favorite search engine for the keywords "SSH" or "secure telnet" for further information.
- Only knowledgeable and trusted administrators should have accounts on the master and slave ACE/Servers.
- On AIX, check that in the file `/etc/security/limits` the value for `fsize` meets the minimum requirement of 4194303.
- A reliable and accurate system time is critical to proper ACE/Server operation.
- Some services must be added in the `/etc/services` file. Check the documentation for details.
- Add your server host name in the `/etc/hosts` file.
- You should create a UNIX group whose members are all those users who will be registered in the ACE/Server database as administrators. Select one of those members to be named as the owner of all ACE/Server files. This user ID has to be listed in the `/etc/passwd` file.

9.3.9 MIMESweeper

As discussed in 5.4.2, "MAILsweeper" on page 135, MIMESweeper consists of two components, MAILsweeper and WEBSweeper. These are discussed in the sections that follow.

9.3.9.1 MAILsweeper

As a general consideration, if you configure MIMESweeper, MAILsweeper and WEBSweeper need to be on separate machines.

Perform the following tasks before configuring MAILsweeper:

- Determine the mail domains being used internally. MAILsweeper and the firewall mail exchanger must be configured to accept mail for each of these mail domains.
- Determine which secure mail servers support each of your domains. MAILsweeper must be configured to forward mail addressed to any of your mail domain to the correct secure mail server.
- Determine the address of the MAILsweeper server. Each of your secure mail servers must be configured to forward the mail received from internal clients to the MAILsweeper server.
- Determine the address of the firewall or your outbound mail routers. MAILsweeper must be configured to forward mail addressed to external domain to the firewall mail exchanger or any outbound mail routers.
- If you do not want clients to be able to bypass SMTP checkings performed by MAILsweeper, you will need to set up a connection on the firewall to limit SMTP access to the outbound mail routers to only the MAILsweeper server.
- If you install *MAPI32.dll* from the Windows NT CD-ROM, and then install Microsoft Management Console 1.1 from the MIMESweeper CD-ROM, the correct version of *MAPI32.dll* is overwritten with a back level version installed with Microsoft Management Console. After installing the Microsoft Management Console, make sure you install *MAPI32.dll* Version 4.0 or later. This component is normally found in the Windows Messaging component.

9.3.9.2 WEBSweeper

Perform the following tasks before configuring WEBSweeper:

- Determine the address of the WEBSweeper server. This will be needed by each of the client Web browsers in the network. The browsers must be configured to use the WEBSweeper server as their proxies for HTTP, HTTPS, and FTP. If DNS is being used, you can set up a name, such as proxy.mycompany.com.
- Determine the address of the secure interface of the firewall. WEBSweeper must be configured to forward proxy request to the HTTP proxy residing on the firewall. If the HTTP proxy is in another location, you need to determine this address.
- If you do not want clients to be able to bypass Web content filtering, you will need to set up a connection on the firewall to limit proxy access to your WEBSweeper and/or SurfinGate servers.
- Do not install WEBSweeper on the same machine as MAILsweeper.

9.3.10 Dependencies

When installing all the components for a security and Internet infrastructure, such as the SecureWay Boundary Server, MAILsweeper, WEBSweeper, ACE/Server, Norton AntiVirus, SurfinGate, Web servers, and mail servers, you should be aware of what IP connection these utilities need to establish among each other and to or from the internal network and to and from external networks in order to operate as expected. You should write down these services that each machine is to provide and to what IP connections these services refer. Then, you need to create the rules at the firewall that defines these connections. For example, Web server machines will be contacted mainly on port 80 using a TCP connection that will be started by the clients on the Internet. Also, you might be hosting secure pages to provide the security to carry e-business on the Internet. Those connections are normally directed to the TCP port 443 on the Web server. From this example, you can write down two kinds of connections that you will need to allow through the firewall to achieve a functional Web server.

The remote administration of the server machines can be useful, but you should always keep an eye on security and check how those remote administration utilities perform their communications. Some administration utilities, such as the firewall configuration utility, allow for encrypted sessions. If you are using UNIX servers, you might avoid using plain Telnet or FTP sessions by using replacement utilities, such as SSH or secure Telnet. Search the Web for information about these items using your favorite search engine or look, for example, at:

<http://www.ssh.fi>

9.3.11 Testing the installation

The following is a list of things you might want to check to see if an SBS installation is working properly:

- Performance – You should monitor the performance of every component of the SBS complex if you are suspicious of a performance problem. SBS is designed with scalability in mind; so, if any component becomes a bottleneck, you should check for misconfigurations and/or consider doing an upgrade on a particular machine or get more machines to do that particular job.
- DNS – Internal hosts must be able to resolve internal and external names, which is the same for the firewall resolver. External hosts should not resolve internal addresses.

- HTTP – You should test HTTP, HTTPS, and FTP from internal machines, check what info is exploited in outgoing requests, and check what passwords are used with anonymous FTP.
- SMTP – Test mail in both directions (for example, by using an echo mailer, such as echo@tu-berlin.de) and what information is exploited in outgoing mails (such as names of your internal machines or internal domains that should not be known on the Internet).
- NSA – Network Security Advisor can check your firewall from both sides with policy “firewall”; it can also scan other hosts of SBS and in DMZ.
- Other utilities – You should consider getting a utility to also check for viruses in corporate-internal mail. It is very usual that a person might send an infected file, such as a word processor file or worksheet file, that he or she did at home and attached it on a mail to some other person inside the company. If you are not using SMTP for internal e-mail delivery, then you should look for a utility to check on your mail server for viruses. Remember that a considerable number of attacks originate from inside an organization’s network. You should plan for this also.

9.4 Intrusion Immunity

Intrusion Immunity (II) is one of the five building blocks that constitute the IBM SecureWay FirstSecure framework. II consists of an intrusion detection component (Tivoli Cross-Site for Security) and a virus detection component (Norton AntiVirus Suite). The Norton AntiVirus Suite from Symantec Corp. (<http://www.symantec.com>) is included with IBM SecureWay FirstSecure to complement intrusion detection on the network, which is done by Tivoli Cross-Site for Security. The following is an overview of the installation planning and the installation steps required to install Tivoli Cross-Site for Security.

The installation of Tivoli Cross-Site for Security involves several planning and execution steps that are outlined in the *Tivoli Cross-Site for Security Installation Guide* that ships online on the product CD-ROM. This guide contains separate sections for each supported platform: IBM AIX, Sun Solaris, and Windows NT.

To recap, Tivoli Cross-Site for Security consists of three different components. As part of IBM SecureWay FirstSecure, all these components are available for AIX, Sun Solaris, and Windows NT:

- The *Cross-Site Security Server* usually runs on one dedicated machine where it requires a Netscape Enterprise Web server and an IBM DB2 or

Oracle database. The security server actually consists of two components, the *Cross-Site Management Server* and the *Cross-Site for Security Server*, which must be installed on the same machine.

- The *Cross-Site Management Console* runs on the administrator's workstations or directly on the management server.
- The *Cross-Site Agents* are installed on one or more machines that are determined to analyze the network traffic.

Installation steps and considerations for each component are explained in the following.

9.4.1 Prerequisites

The detailed hardware and software prerequisite for Tivoli Cross-Site are described in the *Tivoli Cross-Site for Security Installation Guide* and in any *README* files or *Release Notes* that come with the product or are available at the following Web site (registration required):

<http://www.cross-site.com/support/docs>

You should always check this Web site for latest information about the product and last-minute information. The *Tivoli Cross-Site for Security User's Guide* is, along with other material, also available for download from the above Web site.

The basic prerequisites for Tivoli Cross-Site for Security are provided below for your convenience and for planning purposes (for more details, such as memory requirements and any patch levels, please check the *Tivoli Cross-Site for Security Installation Guide* and the actual *Release Notes* at the time of installation):

- AIX and Sun Solaris require the *Java Development Kit (JDK)* on the Security Server, the Management Console, and the agents.
- The Security Server requires *IBM WebSphere Application Server* that is packaged with the Tivoli Cross-Site for Security product.
- The WebSphere Application Server, in turn, requires a Web server. At the time of writing, Tivoli Cross-Site for Security is supported with *Netscape Enterprise Server 3.62* only. This Web server is not included with the product and must be purchased separately. A test and trial version can be downloaded, for example, from:

<http://www.ipplanet.com>

- The Security Server requires either IBM DB2 or Oracle as a relational database management system (RDBMS). These are not included with the

product and must be obtained separately. The RDBMS can be installed locally (on the same machine) or remotely. A remote installation may be considered for performance reasons.

- A Cross-Site *license key* is required to install and run the Security Server (actually for the Management Server as part of the Security Server, to be more precise). The license key can be obtained online from:

<http://www.cross-site.com/support>

You will be required to fill in some information that can be found in the product package, such as a reference number and a customer number. The *Tivoli Cross-Site for Security Installation Guide* and the *IBM SecureWay FirstSecure Planning and Integration Guide* contain additional information on where to find the required information and how to obtain the license key.

- On AIX, the agent requires the TCP server fileset (bos.net.tcp.server) to be installed.

With these prerequisites in place, the installation can take place as described in the following section. If these components are to be installed in a diverse network, you might check 10.3, “Intrusion Immunity” on page 294 first to understand some implications if network boundaries are involved.

9.4.2 Installation

This section overviews the installation of the Tivoli Cross-Site for Security. Only a generic description is provided here for your convenience and planning purposes, and you should always check the *Tivoli Cross-Site for Security Installation Guide* and the applicable *Release Notes* for more detailed step-by-step descriptions and up-to-date information.

9.4.2.1 Security Server

After verifying that all the prerequisites are in place, the following categories should be considered when required to install the Security Server:

Database Management System

As mentioned above, the database can be installed on the same, or on another, machine. The latter choice might be considered for performance reasons. The *Tivoli Cross-Site for Security Installation Guide* lists and explains some specific tasks for IBM DB2 or Oracle that need to be performed in order to prepare the RDBMS for Cross-Site. If the RDBMS is installed on a remote system, an RDBMS client must be installed and configured in order to access the remote database.

Depending on your installation, you might consider creating a new database instance to separate the Cross-Site database from other databases.

Netscape Enterprise Server

The Netscape Enterprise Server is required to run the servlets that serve as an interface to the agents and other components of Tivoli Cross-Site products. Please follow the installation instructions that come with the server. In general, installing the Web server is relatively easy and can be kept minimal. It is important to understand that the provided installation script (*ns-install*) only installs and configures an *admin server*. This admin server is then started and accessed through a Web browser in order to configure the actual Web server(s). The admin server, if not used, should be stopped for security reasons.

If secure communication is required (which is recommended), the Web server needs to be configured to support SSL using an appropriate server certificate. There are several options to obtain a valid server certificate including:

- A temporary test certificate can be used. This is only recommended for short-term test installations. *Verisign* (<http://www.verisign.com>), for example, offers such test certificates for free. Such certificates expire after a few days or weeks.
- An official server certificate, generated by a well-known Certificate Authority (CA), such as *Verisign*, *Entrust*, or *IBM World Registry*, to mention just a few. These are, in general, not free of charge but offer a convenient and standard way to obtain and maintain server certificates.
- Private (self-generated) certificates can be used as well. These can be generated, for example, in an organization's own *Trust Authority* secure domain (see also Chapter 7, "Public Key Infrastructure (PKI)" on page 167).

Although the Netscape Enterprise Server supports importing externally created private keys, the normal procedure is to create a key pair with a command line tool (*sec_key*) that comes with the Netscape Enterprise server. Then, a certificate request is composed through the admin server that must be forwarded to the Certificate Authority (CA) of choice. This can be done through e-mail or by copy and pasting the request content into a browser form entry area as, for example, when requesting a test certificate from Verisign. After receiving the certificate, the Web server must be configured through the admin server to use a key file and the corresponding certificate.

Cross-Site Management Server

After the prerequisites are in place, the Cross-Site Management server can be installed and configured. Depending on the platform (AIX, Sun Solaris, or Windows NT), different installation procedures apply. You should make sure that you have the license key ready for the configuration of the Management Server (see 9.4.1, “Prerequisites” on page 253). On Windows NT, the installation and configuration is done using a series of InstallShield dialog windows, while on AIX and Sun Solaris, separate configuration scripts need to be run after the installation. The following information is required for the installation and configuration (not applicable equally for all platforms):

- An installation directory.
- The installation path for the JDK and the Netscape configuration files.
- Communication protocol to access the Management Server (HTTP or HTTPS).
- Name and port number for the Management Server.
- Key file and password if SSL is being used.
- Name of the administrative domain.
- Name, address, and description of this Management Server.
- A directory as a target for console and agent installation packages that can later be used for downloading such packages.
- Database details, such as type (DB2 or Oracle), instance name, installation directory, database user name and password, and database name.
- Optionally, a remote X-server name or IP address if accessed remotely.
- Optionally, a path to a Web browser that is launched automatically after configuration that allows you to enter the license key.

After successful completion of this configuration script, the Management Server is ready, and the Cross-Site Security Server can be configured using a separate configuration script. This script does not require any user-input.

Management Console

Before installing the Management Console, the prerequisites must be in place, and the minimum requirements (see the *Tivoli Cross-Site for Security Installation Guide*) must be met. If it was chosen during the Management Server installation to have installation images stored on that server, the installation image can be downloaded from that server through a standard Web browser. Otherwise, the installation can be done with the product CD-ROM. Windows NT uses an InstallShield installation and configuration process, while on UNIX (AIX and Sun Solaris), it is done through script programs. The configuration requires the following information to be entered:

- Management Server hostname, protocol (HTTP or HTTPS), and port.
- Alternate keyring file, if such a one had been created.

Note

The *alternate keyring file* (see above) may need some explanation. Tivoli Cross-Site for Security comes with a default keyring file that contains the root certificates of most well-known Certificate Authorities (CAs), such as Entrust, Verisign, and so on. A CA's root certificate is required for secure SSL communication using a certificate signed by that CA. If the server certificate was signed by one of these well-known CAs, nothing needs to be done; the Management Console will accept such certificates. If, however, the certificate was signed by another CA, such as an organization's private CA, the CA's root certificate must be added to the default keyring file. Tivoli Cross-Site provides a tool that allows you to create a keyring file that not only contains the standard CAs' root certificates, but also any new ones that you specify. Such an alternate keyring file then needs to be used for the console and be distributed to the agents (see below).

After successful configuration, the console can be run. It prompts for a user ID and password to log in to the Management Server and to launch the graphical user interface.

Cross-Site Agent

The Cross-Site Agent is a background application that does not have a user interface. It communicates with the Security Server to retrieve configuration details and to forward event information. From the Security Server point-of-view, a Cross-Site Agent represents a user. Therefore, planning is required on whether each agent shall be managed as a single user, or whether multiple agents share the same user properties, which is perfectly possible and permissible. When the Security Server is installed, two users are defined by default: *install* and *admin*. Either of these users can be used to install agents. If you decide to use a new user for an agent, this user has to be created before the installation of the agent takes place. Make sure that any new user for this purpose is assigned the *install* role.

The installation and configuration is done through InstallShield on Windows NT or by means of scripts on UNIX. If it was chosen during the Management Server installation to have installation images stored on that server, the installation image can be downloaded through a standard Web browser from that server. Otherwise, agent installation can be done with the product CD-ROM as well. The InstallShield or configuration script require the following information for successful configuration:

- Management Server hostname and port
- User ID and password for the installation user (see above)

- Alternate name for the agent (optional)
- An alternate X-server display for displaying a welcome screen (optional, UNIX only)
- Alternate keyring file, if such a one had been created (see the note under “Management Console” on page 256)

After installation and configuration, the agents are started automatically on the target systems and do not normally require any special attention. Configuration management is done through the Management Console.

9.4.3 Testing the installation

A good (and important) test for the success of the installation and configuration is certainly to watch for any error messages during the installation and configuration process. Then, any newly configured agents must show up in the Resources tab of the Global or Security view of the Management Console.

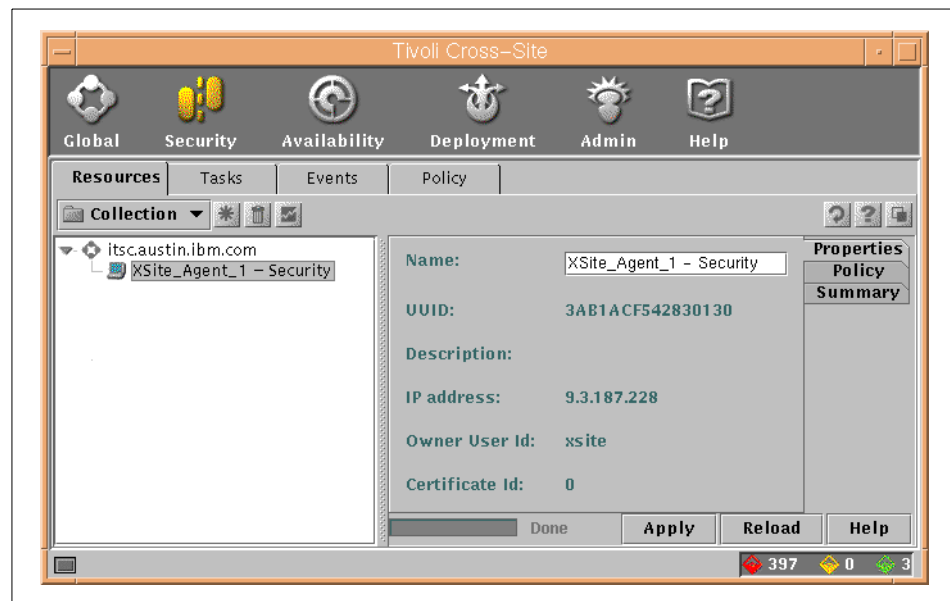


Figure 72. Management Console with new agent

Figure 72 shows an example of the Management Console after a new agent (*XSite_Agent_1*) has been added (do not forget to click on the **reload** icon).

A next basic test is to watch whether security events show up under the Events tab of the console. Under normal circumstances on a network with at least some traffic, it should not take long until some events become visible.

The above is only a very basic test to see whether the installation generally works. More tests might be necessary, especially when customized events are being tracked.

9.4.4 Operation

Although daily operation is beyond the scope of this book, the following should be considered after successful installation:

Database – Tivoli Cross-Site uses a database (IBM DB2 or Oracle) primarily for the storage of events. Over time, any table space limits may be exceeded, and the reliable operation of the product becomes impossible. Database monitoring is, therefore, an important maintenance task for a reliable operation of Cross-Site for Security.

Agents – Agents work in a passive way, and it might not become obvious when an agent stops operating normally. It is, therefore, good practice to monitor the proper function of agents either through the console (are there events generated by an agent?) or by additional means, such as process monitoring on the agent systems.

Alert forwarding – Events can be configured such that they generate an e-mail message to a specified e-mail address. Or, if a Tivoli Enterprise Console (TEC) is being used, events can be forwarded to show up on the TEC. Please refer to the *Tivoli Cross-Site for Security User's Guide* (included online on the product CD-ROM) for details on the configuration.

9.5 Trust Authority

The IBM SecureWay Trust Authority (TA) is the building block of the IBM SecureWay FirstSecure framework that supports a Public Key Infrastructure (PKI). Please refer to Chapter 7, "Public Key Infrastructure (PKI)" on page 167, for a general introduction to Trust Authority.

Before you plan for and install Trust Authority, make sure that the following product documents are available:

- *IBM SecureWay Trust Authority Up and Running*
- *IBM SecureWay Trust Authority Configuration Guide*
- *IBM SecureWay Trust Authority System Administration Guide*

Most conveniently, these documents, and others, can be viewed online at:

<http://www.ibm.com/software/security/trust/library>

In addition, and most important, make sure that you have the latest *Readme* document available at the time of installation. Registered customers can access the Readme document at:

<http://www.ibm.com/software/network/support/updates>

This section of the redbook does not repeat all the information contained in the documents listed above, but rather gives you an overview for your convenience and for planning purposes. Some explanations are given here that help you understand the single steps throughout the installation and configuration process. Bear in mind that some details may change as maintenance packages are released for the product, and, thus, it is important to have the latest Readme document available.

9.5.1 System environment

When deployed in a production environment, a Trust Authority installation becomes a very critical element in an organization's security infrastructure because TA issues and manages certificates that end entities (users and other resources) rely on. Having this in mind, it becomes inevitable that the TA system environment must be protected from potential attacks. Notice that the TA's internal trust model already involves a high level of safety, but it cannot guard against physical attacks or negligence of the involved personnel (see also 7.2.2, "System configurations" on page 177).

For the best security, TA systems should be installed in a physically secured location and network access should be restricted to authorized purposes only. This can best be achieved by installing TA on a separate network that is connected only through a firewall. This is discussed, for example, in 5.2, "Defining network boundaries" on page 122 and also in 10.2.4, "Boundary services on the internal network" on page 292.

The server components of Trust Authority are supported on Windows NT and IBM AIX. Because much additional internal processing is required for maintaining the internal trust model (for example, encryption for communication between the components) and because the systems also run a local copy of DB2, it is highly recommended to run Trust Authority on well-performing systems. Suggested minimum configurations for single-system installations are:

- RS/6000, model F40 with two processors, 10 GB disk, 256 MB memory
- Intel Pentium II system, 450 MHz, 10 GB disk, 256 MB memory

The *Up and Running Guide* contains some suggested system configurations for different scenarios and expected loads. Bear in mind, however, that it might be difficult to predict the expected load and, thus, like in many other new environments, planning should involve some basic system performance analysis and tuning activities.

The use of the IBM SecureWay 4758 PCI Cryptographic Coprocessor is optional. It is supported on AIX systems with PCI bus only.

9.5.2 Software requirements

The software requirements for Trust Authority become more evident after a brief description of the components' technical implementation (please refer to Figure 62 on page 174):

- The Trust Authority server components support IBM AIX (4.3.2) or Windows NT 4.0 (with Service Pack 5). On AIX, the C++ runtime library, xIC.rte (3.6.4.1), must be installed.
- The CA and Audit servers run as daemons and require IBM KeyWorks for their internal security, such as decryption and encryption. Also, the CA and Audit servers maintain their own data store, which is IBM DB2 Enterprise Edition 5.2 (FixPack 10).
- The RA runs as an IBM WebSphere application and, therefore, requires IBM WebSphere Application Server, Standard Edition (2.0.3.1). The WebSphere Application Server, in turn, requires the Java Development Kit (JDK), an IBM HTTP Server, and the IBM Global Security Kit (GSKit) for SSL communications. The RA also requires KeyWorks for data security, IBM DB2 Enterprise Edition 5.2 (FixPack 10) as data store, and the IBM SecureWay Directory (3.1.1) for certificate and CRL publishing.
- The RA desktop runs as an applet in a Web browser on a Windows 95, 98, or NT system. No special requirements other than enough system resources and capable browsers are necessary.
- The TA client runs on a Windows 95, 98, or NT system with sufficient system resources.
- The *Up and Running Guide* lists Web browser requirements to run the Setup Wizard and the RA desktop. Generally, it is a good idea to run current releases of either Netscape's or Microsoft's browsers (not included with Trust Authority), which meet the requirements. The browsers must be enabled to run Java and JavaScript applications.

The Trust Authority media package contains the required prerequisite software in their correct release level including:

- IBM DB2
- IBM KeyWorks
- IBM WebSphere Application Server (including JDK, HTTP Server, and GSKit)
- IBM SecureWay Directory

As with the hardware requirements and precise installation procedures, details might change once maintenance packages for the product are released. Please review the *Readme* document for the latest information.

9.5.3 Installation and configuration

Like with most other, non-trivial applications, especially when some or all of Trust Authority's customization features are utilized, it is recommended to run a separate test and training installation. The following is an overview, step-by-step description of the installation process for a single-system environment. Please refer to the *Up and Running Guide* and the *Readme* documentation for additional information about setting up Trust Authority in a multi-system environment.

1. Ensure the hardware and operating system prerequisites are in place. Make sure that TCP/IP communication is set up and working including host name resolution. On AIX, make sure the correct level of xLC.rte (3.6.4.1) is installed. On both platforms, ensure JDK is installed and at the level included with the Trust Authority media package.
2. Install the IBM SecureWay Directory Server. If you choose to install it on the same machine as the Trust Authority servers, do not do any basic configuration, such as specifying an administrator account, password, or default database. Trust Authority will do this later on.
3. Install DB2 Enterprise Edition and Client Application Enabler with the appropriate fix pack. DB2 must be set up to support UTF-8 code set. On Windows, this is achieved by an environment variable *DB2CODEPAGE*, which needs to be set to *1208*.
4. On Windows NT, create a new user account (for example, *cfguser*) with administrative privileges. Define a password for this user account and make sure that **User Must Change Password At Next Logon** is not selected. On AIX, such a user account is created automatically.
5. Install the HTTP Server. The HTTP Server (Version 1.3.3) is included with the media package. On Windows NT, make the user account, defined in the previous step, the owning user of the HTTP Server service. On AIX, also install the HTTP Server SSL Module.

6. Install WebSphere Application Server. Choose HTTP Server 1.3.3 as the Web server in the configuration.
7. Ensure that the HTTP server is not started automatically at system startup. Trust Authority starts and stops the HTTP Server as needed. On Windows NT, change the *start* property in the Services folder of the Control Panel. On AIX, remove any entries in /etc/inittab that start the HTTP Server.

8. On Windows NT, log on as the newly created user (see step 4.).

9. Install Trust Authority. On AIX, this is most conveniently done through SMIT. On Windows, the installation is done with an InstallShield guided process. (At the time of writing, known error or warning messages may show up on AIX, which are not critical. See the *Readme* document.)

On AIX, select the server components to install (most likely all), and on Windows, InstallShield gives you the option to select different servers. You may choose to not install the client and the RA desktop if you want to run them on another machine.

After the installation, three more steps need to be performed for basic configuration.

10. A post-installation process must be run. This configures and initializes the database and the Trust Authority components. Make sure that the environment variable %TEMP% is set on Windows. Run the CfgPostInstall program (on Windows, this can be done through **Start -> Programs -> IBM SecureWay Trust Authority -> Post Installation Configuration**).

Follow the configuration messages. Although there might be warnings or even errors shown, you should watch for a “successful” indication at the end.

11. Run the Setup Wizard. The Setup Wizard is an applet that runs in a Web browser. For performance reasons, this could be run on another machine. To run the Setup Wizard, point the browser to:

`https://<TA server name>:81`

(Notice the **https**, not **http**.) Follow carefully the instructions on that page to download and install the *swingall.jar* file. Then, on the same page, click on **CfgSetupWizard.html**. Be patient, it may take a while to download the wizard. The wizard guides you through a series of dialogs:

- The Trust Authority server name, port numbers, and DN. The information given in the DN will be used later as identifying information contained in the self-signed CA certificate (root certificate). The ports should be left as default unless there are strong reasons not to accept them.

- Selection of encryption defaults and key lengths. Unless there are reasons not to do so, you should accept the defaults.
- Information about the LDAP directory server. This includes directory root and administrator information and passwords. If the LDAP directory is on another machine, and/or the directory is already configured, refer to the document *Using the SecureWay Directory With Trust Authority*, available at the Web site mentioned in the introduction to this section.
- A *Registration Domain Name* that is the name for your security domain along with registration domain language. The registration domain name must be unique and must not contain any blanks or other special characters (see the *Readme* document for details). The registration domain name will become part of the URL that users will need to access this Trust Authority installation.
- Information (name and port) about the Web server running on the system to access the various Trust Authority functions. It is recommended to accept the defaults unless there are strong reasons not to do so.
- A port number for TA client accesses.

Upon completion, the Setup Wizard shows a summary and then stores the entered information on the Trust Authority server.

12. The last step in setting up Trust Authority is to run the `CfgStart` command that does the final configuration. On AIX, this command is run automatically after finishing the Setup Wizard. On Windows NT, run the command (`CfgStart`) from a Command Window within the bin directory of Trust Authority (for example, `c:\Program Files\IBM\Trust Authority\bin`).

There might be warning or error messages produced during the execution of `CfgStart`. Make sure that it says “successful” at the end of the process. You can view the log files. On AIX, they are in `/usr/lpp/iau/logs`, and on Windows NT, they are in `<install_dir>\logs`, where `<install_dir>`, by default, is `c:\Program Files\IBM\Trust Authority`.

This completes the installation and basic configuration of Trust Authority. The configuration process includes the necessary configuration for the HTTP Server (including SSL) and the LDAP directory (unless an existing LDAP directory is being used, in which case, there are some additional, manual steps to be done). The configuration process also creates a self-signed Trust Authority CA root certificate that will be required in Web browsers to communicate securely with Trust Authority.

It is recommended, after completion of the installation, to run a quick test (see the next section) before the Trust Authority server is restarted.

9.5.4 Testing the basic installation

After the installation and basic configuration are completed, a simple test can be performed to see the proper function of Trust Authority. Since Trust Authority approves Web browser certificates with a validity period of one year automatically, the Trust Authority installation, at this point, is ready to approve and issue such certificates.

To test this, point your browser at:

```
http://<TA server name>/<security domain name>/index.jsp
```

For example, if the TA server name is *alpha.beta.gamma.com*, and the security domain name (as defined through the Setup Wizard before) is *MyDomain*, the URL would be:

```
http://alpha.beta.gamma.com/MyDomain/index.jsp
```

The upcoming page has a link for downloading the Trust Authority's self-signed root certificate. This should be done as the first step because Trust Authority is a new CA that the browser needs to be made aware of. Downloading the root certificate, however, is not mandatory but recommended in order to avoid unnecessary "new site" pop-up warnings from the browser on subsequent connections to the Trust Authority server. At the bottom of the same Web page, there is a *Certificate Enrollment* dialog box as shown in Figure 73. Accept the defaults to enroll for a Browser Certificate and click on **OK**.

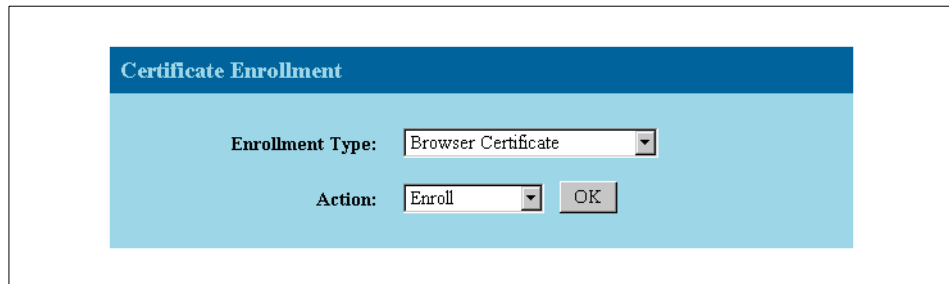


Figure 73. Certificate enrollment dialog (partial page)

The next page (notice that this is now a secure HTTPS connection to the Trust Authority server) presents the default enrollment form for a certificate. Do not change the certificate type because not all certificate types are

auto-approved. It should be a 1-year Web client authentication certificate. Fill in the information as requested and click on **Submit Enrollment Request** at the bottom. The browser will now create a key-pair (you may be informed by a pop-up message), of which the public key is being included with the certificate request, and the private key remains local in a password-protected file.

The next page is important because it contains the request ID that was generated in response to your request. If you entered a challenge question and response on the previous enrollment dialog, you will be required to enter the challenge response here. The bottom part of that page is shown in Figure 74.

Check Enrollment Status	
This Request ID was generated for you when you submitted your enrollment form.	Request ID: 9FPqEPr\byOuRtea.JySODA==
You did not specify a Challenge Question on your enrollment form.	Challenge Question:
If you specified a Challenge Response during enrollment, type it here, exactly as you typed it on the enrollment form. The Challenge Response is case-sensitive.	Challenge Response: <input type="text" value="Vodka and Gin"/>

Figure 74. Check enrollment status form (partial page)

After clicking on **Check Enrollment Status** at the bottom of the page, the certificate will be downloaded to the browser automatically, and the text on the upcoming page will indicate this. If this is not the case, you may have been too fast; allow Trust Authority a little time to create and sign the certificate.

The correct receipt of the certificate can be verified through the configuration settings of the browser. With Netscape Communicator, for example, click on the **security** icon and then select **Yours** in the Certificates list. If everything worked fine, the new certificate appears in the list. Notice when viewing the details of the new certificate that the Trust Authority is displayed as the CA that issued this certificate.

If the test was not successful, there might be several reasons. First, ensure that the test was done correctly and there was no obvious indication of an error. Then, check the log files in Trust Authority as to whether they indicate a point of possible misconfiguration. If all problem determination fails, consider installing Trust Authority from scratch.

If the test was successful, it is recommended to stop Trust Authority, restart, the system, and rerun the same test. This ensures that the configuration, as well as the start and stop procedure, works correct.

9.5.5 Running the RA desktop

The installation and basic configuration as described in the previous sections brought Trust Authority up to a point where certificates could be requested and auto-approved (some certificates are, by default, auto-approved).

But, so far, we do not have the administrative RA GUI (the RA desktop) to administer the certificate handling of Trust Authority. This requires an additional step that is described next. Make sure that the RA desktop software is installed on the system from where it is to be launched. The RA desktop software can be installed by running the RADInst.exe file from the product CD-ROM. While installing the RA desktop, the InstallShield prompts you for the RA's URL. If the RA server's name is *alpha.beta.gamma.com*, and the security domain name is *MyDomain*, the URL is (default port assumed):

```
https://alpha.beta.gamma.com:1443/MyDomain
```

As the first step to set up the RA desktop, you need to request a browser certificate from the browser where the RA desktop is to be run. This is a requirement for Trust Authority in order to authenticate the RA administrator. To do this, follow the same procedure as described above for testing the proper function of Trust Authority.

Next, you need to locate a certain record in the database in order to add this user to the administrator list. As the user with access privileges to DB2, open a DB2 command session and enter the following commands:

```
db2> connect to pkrfdb
db2> select last_name, credential_uid from requests
```

(Depending on the number of requests in the database, you may have to limit the number of output records with additional *where* clauses (see the *System Administration Guide*.) Locate your request and note the credential_uid.

Then, go to a command prompt and change to the directory where the `add_rauser` command is located. Enter the following command:

```
add_rauser <path>/domain.cfg <security domain name> <credential_uid>
```

For example, if your security domain name is *MyDomain*, the command on AIX could look like:

```
add_rauser /usr/lpp/iau/pkrf/etc/domain.cfg MyDomain eDgf4dqdbkKd8Hhs==
```

And on Windows NT:

```
add_user c:\Program Files\IBM\Trust Authority\pkrf\etc\domain.cfg MyDomain
eDgf4dqdbkKd8Hhs==
```

(The above represents one single line.) This command adds the user whose certificate credential UUID was added to the list of RA administrators. After this configuration step, you can start the RA desktop from the very same browser from which the certificate was requested (and then received). To start the RA desktop (Windows only), go to **Start -> Programs -> IBM SecureWay Trust Authority -> RA Desktop**.

Allow some time for the applet to load, and the RA desktop, shown in Figure 75, appears.

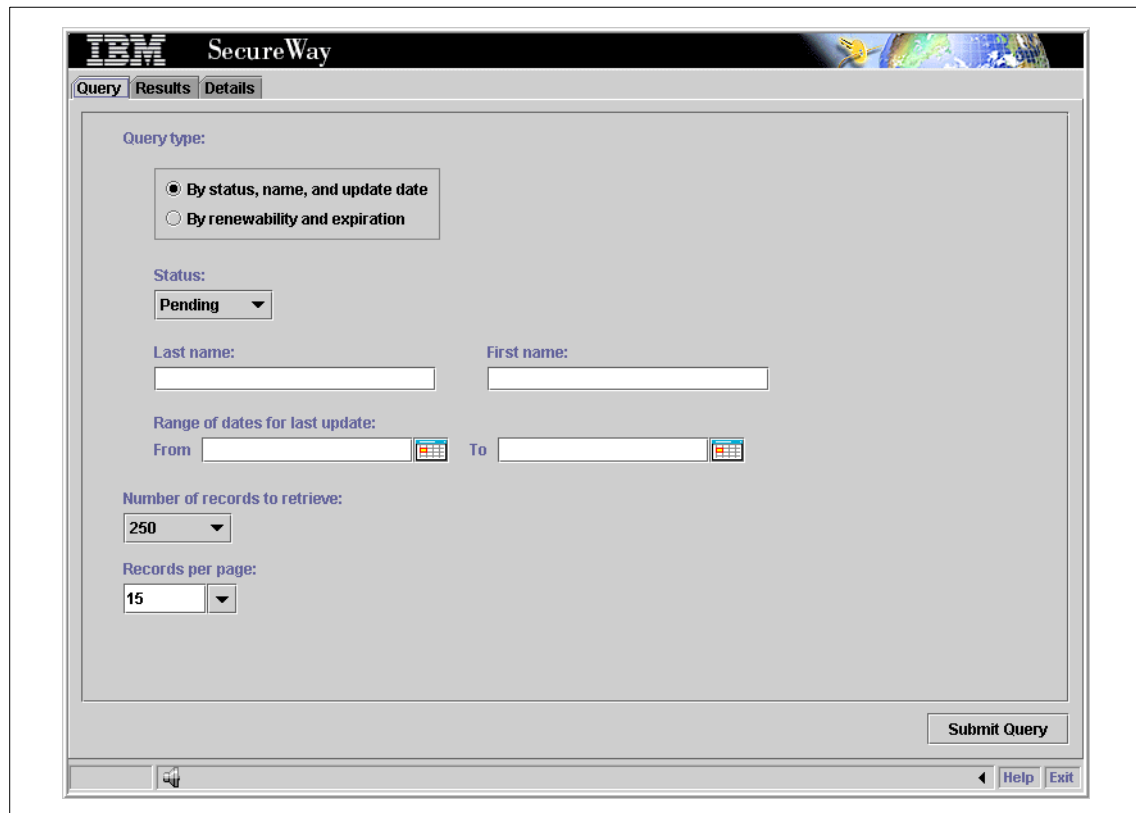


Figure 75. RA desktop (initial screen)

Through the RA desktop, the RA administrator approves certificate requests, revocations, and performs other actions on certificates. Please see the *IBM*

SecureWay Trust Authority Registration Authority Desktop Guide, available online at the URL given in the introduction to this section.

9.5.6 Additional customization

Many of the operational functions of Trust Authority are customizable. This includes:

- Enrollment forms for certificate requests
- Port numbers of the servers (by default, Trust Authority uses ports 80, 443, and 1443)
- Audit log settings
- Frequency of CRL publishing to the LDAP directory
- Various settings in the configuration (*.cfg) and initialization (*.ini) files

It is beyond the scope of this book to explain the many customization options of Trust Authority. Please refer to the *IBM SecureWay Trust Authority System Administration Guide* for more information.

9.5.7 The Trust Authority client

The TA client can be installed on Windows 95,98,NT machines by running the TACInst.exe program available on the product CD-ROM. The InstallShield process guides you through the installation. During the installation, a prompt asks for the password of a virtual Smart Card. The TA client supports the concept of a virtual Smart Card that functions like a real Smart Card but is held virtual, that is, on a file rather than a physical device.

After successful installation, the TA client provides similar basic certificate handling functions, such as a modern Web browser and serves as GUI interface, to manage certificates from the end entity's (user's) point of view. This includes:

- Certificate enrollment
- Certificate download, import, and export
- Process preregistered certificate enrollments
- Certificate revocation
- List certificates

See the *IBM SecureWay Trust Authority User's Guide*, available online, for more information about the TA client.

9.6 Toolbox

The SecureWay Toolbox does not ship with the media package of the IBM SecureWay FirstSecure product. Included in the media package, however, are instructions on how the SecureWay Toolbox can be downloaded from an IBM Web site. This way, IBM has more freedom to improve the components that are included in the Toolbox.

To recap, the Toolbox includes the following components:

- The IBM KeyWorks Toolkit and Policy Table
- The IBM Key Recovery Service Provider
- The IBM Key Recovery Server
- The IBM Policy Director API
- The IBM SecureWay X.509 Public Key Infrastructure for Multiplatforms
- The IBM LDAP Clients for C and Java
- The IBM SSL Toolkit

Please follow the instructions included in your media package to download the components of your needs. Installation instructions are contained in the downloadable files. Also contained is a README file that lists the hardware requirements and software prerequisites. In general, there are no prerequisites besides other Toolbox components; for example, the IBM Key Recovery Server requires the KeyWorks Toolkit. Only the SecureWay X.509 Public Key Infrastructure for Multiplatforms (sometimes referred to as *PKIX Toolkit*) requires JDK 1.1.6.

The IBM KeyWorks Policy table and other components of the Toolbox enable or contain cryptographic services that are subject to U.S. export regulations. Thus, different versions are provided depending of the country of destination or other criteria. For this reason, proof of your country or other eligibility will be required for downloading components of the SecureWay Toolbox.

For more technical information on the toolbox components, please refer to the documentation that is available with each component.

Chapter 10. Deploying FirstSecure in practical scenarios

This chapter provides additional practical information on the installation and setup of the IBM SecureWay FirstSecure components beyond what has already been addressed for each specific area. Since the individual requirements of each environment are different, as are the prerequisites and corequisite products, the focus of this chapter is to provide the FirstSecure administrator with some information that may be relevant when placing these products within his or her environment.

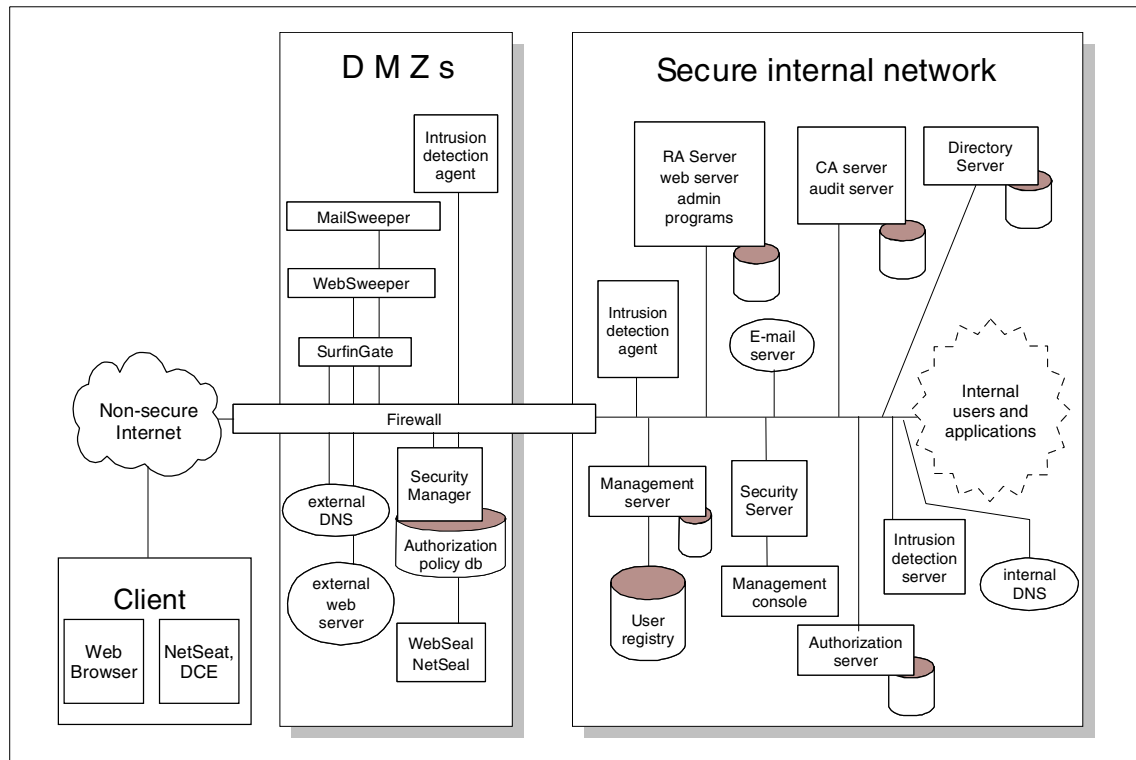


Figure 76. IBM SecureWay FirstSecure - The overall picture

The diagram shown in Figure 76 depicts a hypothetical installation of the various FirstSecure components in one possible scenario, where a firewall separates the secure network and the non-secure Internet through which the client users connect. Although the figure depicts only single instances of most FirstSecure components, these products are scalable and can be combined in many ways to provide optimal protection of sensitive resources. For example, in most enterprise environments, several firewalls are usually

recommended to compartmentalize resources, which effectively bounds the amount of information or resources exposed if a firewall's security is breached. Note that the layout shown in Figure 76 only depicts the firewall as a single unit that spans across one or more hypothetical DMZ.

It is also likely that most organizations implementing IBM SecureWay FirstSecure will probably have some existing security solutions in place, which may necessitate a partial or staged migration. In these instances, other products may be providing similar functions to corresponding FirstSecure components, perhaps even changing the desired relationship between them.

The sections that follow discuss this hypothetical installation. Explanations are given about the individual components of the IBM SecureWay FirstSecure and why they have been placed at the particular points in the scenario. Alternatives are discussed, as well as further options to accommodate larger or smaller environments.

Since the SecureWay Toolbox constitutes components for a development environment that is primarily used in a development and test lab, it is not further discussed in this chapter.

10.1 Policy Director

We will now look at the deployment of Policy Director in a hypothetical scenario. Amongst the areas to consider are:

- Where in the overall topology of the security installation, should the components of Policy Director be installed?
- What, if any, are the technical implications of locating components across firewalls?
- What options are there for scalability, availability, and load balancing?
- What integration of Policy Director and other products can an IT organization benefit from?

We will look at these areas in more detail in the sections which follow.

10.1.1 Locating Policy Director components

In Figure 77, note that there are Policy Director components within the secure internal network (which will usually be referred to in this chapter as the *secure area*) and in the DMZ. This is a hypothetical layout, and there are other ways to locate the components. Let us start by noting where specific components are located.

- Clients are, generally, outside of all firewalls. We have shown, in this case, that *clients* refer to both the DCE clients (or NetSEAT) and the standard Web browsers.
- The Management Server, Authorization Server, and DCE Security Server are all in the secure area. The User Registry is also in the secure area.
- The Management Console is shown inside the secure area.
- The Security Manager (with WebSEAL and NetSEAL) is inside the DMZ.

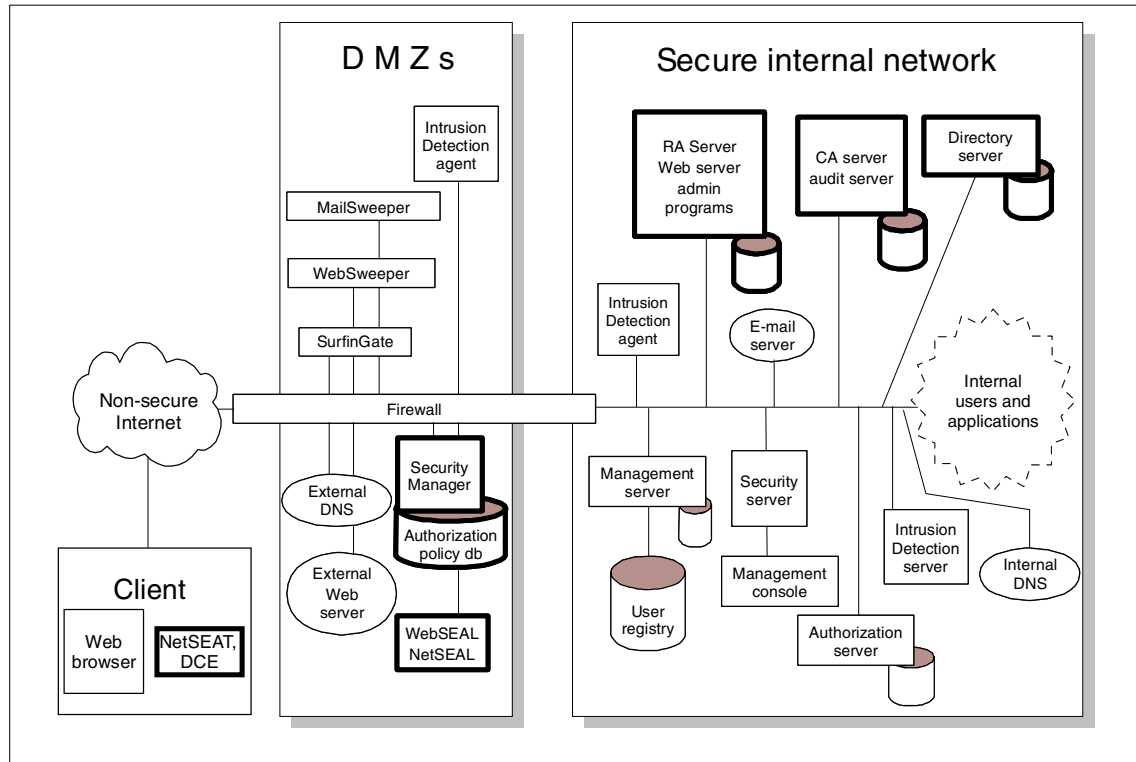


Figure 77. Policy Director Components

As discussed in Chapter 9, “Installation planning and considerations” on page 217, the DMZ is an area that is considered more secure than the unprotected Internet but not as secure as the secure internal network. So what needs to be considered when placing the components in any particular place?

When designing such a setup, there are very rarely clear and straightforward answers. It is always necessary to balance the risks of attacks and the business need for protection against the business needs for accessibility,

response time, and so forth. What we discuss here is not a definitive statement but only some of the considerations involved. The reality in any one organization may be very different; this section is intended only to raise awareness of some of the issues, both from a business or security standpoint, and from a technical perspective that may arise.

We first look a little more closely at where specific components have been located in the hypothetical scenario and at the business or security reasons that might be considered in this decision. Then, in the next section, we will look at some of the important technical considerations, focusing mostly on considerations for installing client-server pairs across a firewall.

Clients

The *clients* referred to in this section are standard Web browsers or NetSEAT clients that want to access the services behind the firewall(s). Note that there may be clients inside the secure area itself. For example, if the Policy Director Management Console is being used inside the secure area and running on Windows NT, it will be running a NetSEAT client for DCE communications. In the Internet world, however, a client is usually thought of as an external agent wanting access to internal resources.

In the scenario described in Figure 77, some of the systems are both clients and servers. For example, the Security Manager is providing a server service to users accessing the network with standard browsers and NetSEAT clients. However, it also acts as a client to the Security Server (through the DCE client installed on it) and, if used, to the LDAP server. Interestingly, in our scenario, in all cases, the client/server relationship takes place across a firewall. Section 10.1.2, "Technical implications" on page 277 covers the technical implications of this situation.

Management, authorization, and Security Server and User Registry

As discussed in Chapter 9, "Installation planning and considerations" on page 217, the information held in the Policy Director databases and the tools to change and manipulate that data are key resources in the corporate infrastructure. These, therefore, need protecting just as much as the applications and business data. So, where possible, it makes sense to put resources into the secure area. No one should be able to access these resources without being authenticated and authorized to do so.

It would be useful to consider, at this point, why resources are put into the DMZ.

When a hacker wishes to attack an organization, one popular method is to obtain access to one of the organization's systems, often a company's

external Web server, and then use the processes serving them to do non-legitimate work on that system or to access other systems. Experienced hackers can cause mayhem if they can obtain such access. From a security standpoint, one of the first things to do is to make it hard for them to obtain such access in the first place. Next, harden the operating system so that if they do obtain access, the damage that can be done is limited. However, assuming that they have obtained access and found a resource on the computer they can use, make it as difficult as possible to break out of that environment and into any other computer. By putting a system into the DMZ, it provides some protection against a hacker entering the system in the first place, and if they do manage to get in, makes it that much harder to access other systems.

It makes sense to put any system that could be an obvious target into the DMZ. External Web servers are, by their very nature, public access points. They also usually need to have a back-end connection to other systems in an organization as the source of the data they present to the public. By putting them behind a firewall, some protection is afforded against the Web server being hacked into. However, users have to be let into the DMZ to access the Web server, and so isolating the Web server as much as possible from the other systems in the network is sensible. For this reason, some organizations set up multiple DMZs so that if a hacker has broken into one, the damage is limited. See Appendix C, “Basic firewall and name service design” on page 327 for more details on firewall and DMZ design.

Another reason to put a resource into the DMZ, rather than the secure area, would be to provide better response times to users on the external network. Inevitably, the more barriers a process has to pass through, the slower it will be in terms of response time. This is another reason to put a Web server in the DMZ. They are being accessed very frequently, and response time is a high priority. Care should be exercised in deciding what Web resources are made available on a Web server in the DMZ. Sensitive information, such as that held on an internal Web server, should be in the secure environment.

Would there ever be a reason to put Policy Director servers (other than the Security Manager (WebSEAL/NetSEAL), which we will discuss shortly) into the DMZ?

Obviously, no one should be accessing the data or tools unless they have a need to so that if an organization is using Policy Director to maintain information about internal users, there would seem to be little need to put this anywhere but in the secure domain. However, one of the main advantages of Policy Director is the ability to define users who will be accessing resources,

such as Web servers and FTP hosts, and who are likely to be external users, not staff within the organization.

Given a population of internal users, and another of external users, it may make sense to have two Policy Director installations, one for each population. There are many reasons why this makes sense. One is that the characteristic access and usage patterns of the two populations are likely to be quite different. Also, the environments they access (what resources are being accessed, what operating systems and applications are involved, and so forth) are likely to be quite different. This means different defined resources in the object namespace, different ACLs, different ACL inheritance patterns, different groups, and group memberships. Indeed, the two populations and the environments they access may have little or nothing in common. Note that none of the above are technical reasons, but rather organizational or administrative.

There are implications for DCE and LDAP in having two or more Policy Director installations. These are briefly discussed in the next section, 10.1.2, “Technical implications” on page 277.

If there are external users defined through Policy Director, these will be trying to access resources either in the DMZ or the secure area, and Policy Director will be providing the authentication and authorization services. In many business contexts, these external users will be customers, and the organization may consider it important to provide fast response times. In this case, there may be reasons to put some of the Policy Director components into a DMZ. This would certainly be a good reason to have two or more Policy Director installations so that the Policy Director for internal users could be in the secure area, while the Policy Director for external users can be in the DMZ. However, since the results of any successful external attack on the Policy Director that defined and controlled external users could have consequences for the organization’s customer base, this should be carefully considered.

Management Console

The Management Console is usually thought of as belonging in the secure area or the DMZ. However, there is an increasing pattern in the workplace for employees to be mobile. Many people now do not work in the same office every day. It would be possible to put the Management Console onto a laptop computer and allow an administrator to log into the network from outside. Since the communication is over secure DCE/RPC, it is a secure communication. As no one can access the Management Console without a correct user ID and password, it is protected this way too. Again, see 10.1.2, “Technical implications” on page 277 for discussion of DCE implications.

Security Manager (WebSEAL and NetSEAL)

The Security Manager is the guard at the gate. It is through this Policy Director component that Web and network access is allowed or denied. In Chapter 9, “Installation planning and considerations” on page 217, we briefly discussed the reasons why the Security Manager will typically sit in the DMZ. The main point to remember is that the Security Manager in its WebSEAL incarnation is a Web server. This means that all the arguments that apply to a Web server also apply here. Also, since the Security Manager is the access and authorization point protecting the target resources, it needs protecting itself from any external attacks. If an attacker could access it, they may disable it or use its resources for further attacks. But, if it sat inside the secure area, that would mean that unauthenticated users would have been allowed inside the secure area prior to being asked to authenticate. Clearly, this would be a high risk. Therefore, the Security Manager is usually in the DMZ.

This is also a reason why it is usually better wherever possible to put the other Policy Director servers in the secure area. When the user has crossed the first firewall, in which case they are inside the DMZ, the only limit on their access has been that provided by the firewall itself. They have not been authenticated by Policy Director. Given the sophistication of some hackers, it could represent an unacceptable risk to have a user access the area of the network where the Management Server (or another component) sits before they have been identified as someone who should have access to that component.

Notice here that we have talked of users *crossing the firewall*. This means that the firewall has allowed communication between the process acting on behalf of the user (the client) and a process inside the firewall. This itself leads on to a different discussion, that is, the technical implications of putting components on different sides of a firewall.

10.1.2 Technical implications

Recall from 9.2, “Policy Director” on page 221 that each server in the Policy Director Secure domain is running a DCE client and, optionally, an LDAP client also. These clients have to be able to talk to their servers in order to provide function to the Policy Director applications they are supporting. For the Policy Director servers within the secure area, this is obviously not an issue. However, a Policy Director server in the DMZ will need access across the firewall. If a user wants to access a Web site within the DMZ, the firewall has to allow this too.

10.1.2.1 DCE and LDAP across the firewall

In Chapter 9, "Installation planning and considerations" on page 217, we briefly discussed the implications of DCE and suggested the use of the `RPC_RESTRICTED_PORTS` environment variable to restrict the number of ports the DCE server actually use. This is necessary because the DCE server will use any one of a wide range of ports as it needs them. On a typical firewall, you can specify access or denial not only by protocol, but by port number too. So, one way a firewall protects resources is by limiting which of their port numbers can be accessed through any particular protocol. Generally, you would not want to allow access to too many ports on any system; otherwise, there is hardly any point in having the firewall. But if you restricted port access on the firewall, without limiting the range of ports the DCE server will use, you will encounter failures when one DCE component tries to access another across a port that has been disallowed at the firewall.

So how would this affect the administrator trying to set out the scenario in Figure 77?

Installation

The first place that an administrator is going to meet the technical issues is right at the beginning. Attempting to install any DCE client across a firewall (note that this would apply to the Security Manager, for example, sitting inside the DMZ but outside the secure area) implies that the installation process will need to have access to the DCE server in order to complete the installation.

Operation

Once the client is running, it will need continual access to the DCE server.

Management

Given modern working patterns, the administrator may wish to have a laptop computer with a Management Console on it for use when on-call or working from a remote site or from home. This console will have a DCE client (or NetSEAT) providing the secure link and will need access across the firewall.

The same need for client-server communication across a firewall applies also to LDAP, but DCE is a little trickier to manage. This is because, as mentioned before, DCE can use a wide range of high-numbered ports. Also, every DCE command, such as an attempt to log in to DCE itself and run DCE commands, opens and binds another port.

As can easily be verified by testing, even installing, setting up, and running a few components, such as a Management Console and a WebSEAL server, will cause DCE to quickly use up well over one hundred ports as myriad calls to DCE processes are made during installation and setup. In a typical

production environment, this could easily be thousands. The same pattern will apply once the DCE clients are being used for day-to-day running of the business.

A normal DCE client operates the same way as the DCE server with respect to port usage. Therefore, both server and client may initiate requests from, and receive requests at, any one of the thousands of ports. However, it is possible to issue the `RPC_RESTRICTED_PORTS` environment variable on the clients too. This is not the case with NetSEAT though, which is a lightweight client and, therefore, does not perform all the functions of a full DCE client.

LDAP, by contrast, tends to have one port (default is 389, or 636 for SSL) that it uses on its server, which makes it relatively easy to manage.

So far, this looks like a problem. How can we give the access we want to give to users and still secure our IT resources? In fact, it may not be difficult at all.

First, we need to clarify some terminology. In the following discussion, it would be confusing to talk of *client ports* or *server ports*, owing to the fact that DCE clients underpin Policy Director servers. Equally, it would be confusing to refer to *source* and *target* ports, as in TCP/IP. The port that was the source of a message a moment ago, may become the destination of the return message now. The important concept is that there is a group of resources that users want to access, and that the firewall is trying to protect. There is also a group of resources, such as a user's laptop computer, being used to access the protected resources, but that are not themselves protected by the firewall. We will refer to these as *protected resources* and *unprotected resources*, respectively.

The first thing to consider is that, in the real world, it is less usual to want to limit traffic by port number on the system that is outside the firewall's protection (the unprotected resource). Usually, the important thing is to protect the port on the protected resource. After all, does it really matter if a NetSEAL server wants to send a message back to a NetSEAL client supporting a user outside the firewall on the client's port 48321? Probably not, unless you have a need to protect the client from attack also (in which case, perhaps it should not be outside the firewall in the first place). You would be much more likely to care whether the NetSEAL server itself was being accessed on a port in which it should not be. So, the real restriction on ports should be done from the point of view of the protected resource ports. The firewall can then be configured to only allow access to the ports specified (see 5.3, "Secure boundary services" on page 124 and 10.2, "SecureWay Boundary Server" on page 287 for more details about configuring the firewall.)

As it can easily be verified in a test installation, the following typical activities do not cause any traffic that cannot be controlled:

- Installation of a DCE clients across a firewall
- Setting up and run a Management Console across a firewall
- Setting up a Security Manager and access a Web server via WebSEAL

The generated DCE traffic, such as from NetSEAT to the Directory Services Broker, or to the Management Server and the DCE User Registry, is sent or received on ports on the protected resource that are bound to those services. NetSEAT, for example, wants to talk to DSB as one of its major activities (DSB is NetSEAT's proxy to the DCE CDS). There are other ports involved, but these ports are on the unprotected resource machine.

There is only a small number of ports on the protected resource side actually being accessed. Usually, the portmapper simply allocated the next available port, though occasionally a port with a very different number might appear. The implication is that if these ports are known, it would be relatively straightforward to set up the firewall to protect the system while allowing the necessary communications. Once the main service ports are identified, and the RPC restriction mechanism (using `RPC_RESTRICTED_PORTS`) is issued to limit the use of the other ports (which were acquired as needed by sporadic DCE requests), the firewall can be set up easily.

One key feature that needs to be mentioned is the DSB. The DSB acts as a proxy to the DCE CDS and is one of the reasons that NetSEAT can be lightweight. There can be multiple DSBs in an environment for load balancing, and it is possible, and sometimes desirable, to have a DSB actually on the NetSEAT client. At installation of NetSEAT, it is possible to de-select use of CDS. If the DSB is on the NetSEAT client, CDS would be directly involved, and this may generate more DCE traffic, although the ports can still be limited.

What makes it a little more difficult is that the DCE services do not necessarily bind to the same ports after rebooting. Therefore, when the server (or the client on a Policy Director server) is recycled, these ports may change.

A useful command that can be run on any DCE client that shows the ports associated with the main services in DCE is:

```
rpccp show mapping
```

Installing the client code within the DMZ may be a tricky operation, unless the firewall can be switched into debug mode (which allows all traffic through)

while the installation is taking place. If you are planning a full installation, it may prove judicious to try out a few of these kinds of installations in a test network with a firewall that is only used for testing purposes, and then observe if there are any patterns you can take advantage of.

LDAP, as stated, becomes relatively straightforward because it is accessed on one port only.

10.1.2.2 Further firewall considerations

The NetSEAT client may be deployed across the firewall to provide secure access to network resources via NetSEAL, and the same considerations will apply here, both at installation time and for day-to-day running of the system. Again, this could be quite easily tested to determine the actual needs of any particular environment. As in all matters related to IT, different environments can produce different results. NetSEAT, however, can communicate with SetSEAL by using SSL, in which case the above considerations do not apply; only SSL traffic has to pass the firewall.

Web access is a major feature of e-business, and this has to be considered also. Most Web servers are configured to listen on one port for all requests, and again this should make protection relatively straightforward. If a secure link is needed, SSL is configured to run on a dedicated port, so again, only the access to that one port by the HTTPS protocol should be needed.

One final consideration in Policy Director is access by third-party applications to the Authorization server. The same basic rules for firewall configuration generally apply as explained above.

In addition, there may be a need to allow for miscellaneous traffic, such as NETBIOS, but now we are in the realm of predictable traffic using a handful of ports.

10.1.2.3 DCE and LDAP considerations for multiple Policy Directors

As discussed previously, there may be good reasons to have two or more installations of Policy Director. One may be scalability, another may be to administer two very different user populations. Remember that the Management Server can be installed only once in any DCE cell. Therefore, two Policy Director installations will mean two DCE cells. It would seem sensible to separate the user populations in DCE terms, too. Unless there is a need for two or more groups of users, from different populations, to access the same resources (in which case, one would question whether these are really separate populations), separating the DCE environments would provide an extra level of security. It is, of course, possible to set up two or more DCE cells in a trust relationship, which would allow users in one cell to access

DCE services in another. This has no impact on Policy Director and does not affect scalability (that is, the Policy Director only provides services to its own cell partners). However, if an organization is running DCE services for other purposes, they may want to have two separate Policy Director installations, but allow access between cells for other DCE services.

In the current release, two Policy Directors will require two LDAP installations, three Policy Directors would require three LDAP installations, and so on.

10.1.3 Scalability, availability, and load balancing

Any IT organization wants to have the ability to scale a particular system up or down to meet business needs. It is a waste of money to have too large a system, and this could have serious impact on the business if it is too small.

Policy Director is very scalable owing to the fact that any server, apart from the Management Server, can be replicated within the DCE secure domain. This replication allows for increased usage and can help to provide high availability since, in most cases, if one server goes down, the others can take the additional load.

This is especially true, for example, in the case of the Security Manager. It is possible, and highly desirable, to have multiple WebSEAL and NetSEAL managers in the DMZ. If one goes down for any reason, this should be transparent to the user.

The only server that cannot be replicated is the Management server, which makes it important to put this server on a reliable machine. One of the main tasks of the Management server is to maintain the master Authorization Policy Database. It also ensures this database is replicated throughout the system since each Authorization server and each Security Manager has their own replica of the database, which the Management Server ensures is up to date. In this way, the servers needing access to this information have it available without having to wait for a response from the Management server, and if the Management Server should go down, they can operate without it.

Any in-depth discussion of load balancing would be outside the scope of this book, but we shall briefly mention a few methods:

- The SecureWay Network Dispatcher can be used for load balancing between Security Managers as well as firewalls.
- The Security Managers could also be installed on an HACMP system.
- NetSEAT can be configured to recognize and load balance over multiple NetSEAL servers.

10.1.4 Managing IBM Firewall proxy users

One of the new features in this release is the ability to use Policy Director to define and maintain proxy users on the firewall. Furthermore, Policy Director can be used to maintain proxy users on multiple firewalls. This will reduce the administrative overhead caused when defining proxy users on each firewall individually and the further overhead of maintaining multiple directories.

We will briefly describe setting up proxy users on a firewall. This is not a comprehensive guide but is here only to provide assistance to an administrator trying to do this task for the first time.

As always, the first step is to plan exactly what is wanted. What firewalls do you want to include in this? What users are common to all firewalls, and what users should only have access on specific firewalls? Finally, if you refer to the *SecureWay Policy Director Up and Running Guide*, you will find that one of the LDAP configuration tasks is to define a suffix in LDAP where your users will be stored in the directory. Is this the right suffix? Would you want to use different suffixes for each firewall? And so on.

There are two prerequisite tasks to carry out before actually beginning setup.

The first task is to edit a file on the Management Console system. The file is called `console.properties`. The line that specifies:

```
6, ProxyUsersTaskView = IV.ProxyUserTask.ProxyUsersTaskView
```

is commented out and needs to be uncommented. If the Management Console is then started, it will look something like Figure 78 (for clarity, only the right hand pane has been expanded):

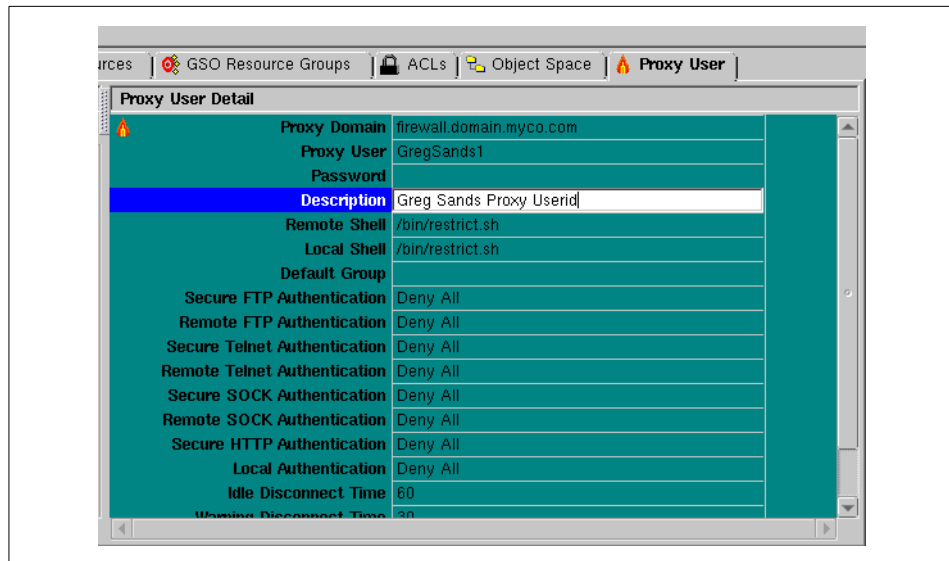


Figure 78. Management Console after editing properties file

Note the new tab, Proxy User, at the upper right. You will not yet be able to add proxy users; however, there is still some work to do.

Next, you have to mount the Policy Director CD-ROM on the LDAP server and find the file `puschema.def` in the `/schema` directory. This file contains some additional schema that need to be added to the LDAP directory to support this function. It is added using the command:

```
ldapmodify -h <ldap server> -p 389 -D <user> -w <password> -f puschema.def
```

`<user>` and `<password>` specify a DN and password for a user with administrative authority).

Now, we can get on to configuration. Ensure that all suffixes required are added to the LDAP directory. You will need the suffix information when you configure the SBS Wizard on the firewall. If you want, you can use a different suffix for each firewall, but beware. This would mean that your firewalls could not share users; every user would need to have a definition on every firewall. It is possible to group common users together and define users specific to one firewall another way; so, you do not need to use suffixes to do that.

Every proxy user will need to be associated with one Policy Director user. Furthermore, the definition of the Policy Director user (defined as an ordinary user, see Chapter 4, "The Policy Director (PD)" on page 53 for details on

doing this) will need, in its LDAP definitions, to be defined with the suffixes to be used for the proxy. If the administrative load is to be kept to a minimum, the safest method is to use one set of suffixes for all users, whether plain Policy Director users or proxy users.

The next task is also carried out in the DMT. At some point in the SBS Wizard configuration, it asks for the password for the *SecurityMaster*. This is a specific user in LDAP that was setup when Policy Director was configured. It can be viewed as shown in Figure 79:

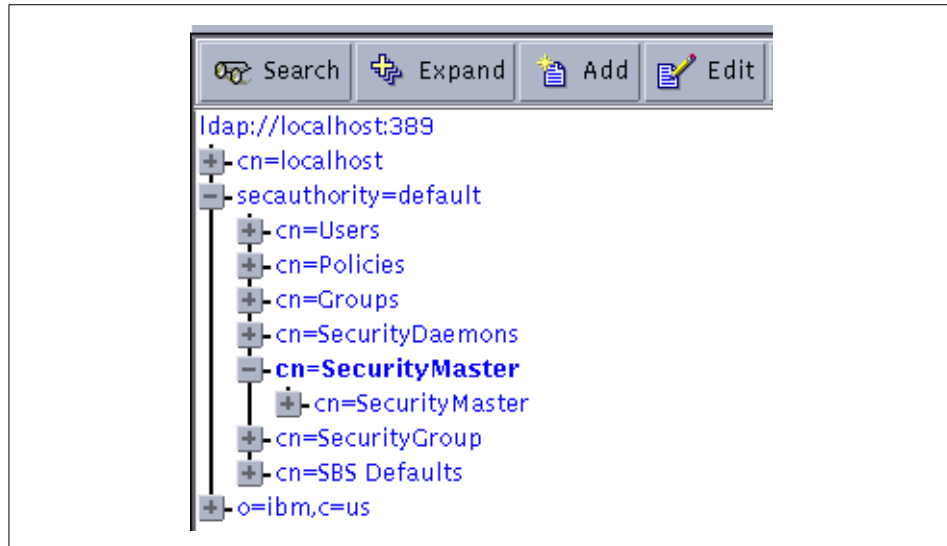


Figure 79. Detail of Directory Management Tool view

This user needs to have the authority to add proxy users. By selecting the top-level `cn=SecurityMaster` and clicking on **Edit**, a screen is presented that allows the administrator to change the password (the one supplied by default is very secure and extremely hard to remember). Once this has been changed, the administrator will need this password to configure the SBS Wizard.

Before attempting to define any proxy users, it is highly advisable to run the SBS Wizard on every firewall that is to be integrated with Policy Director. Setting this up will put some more additional entries into the LDAP directory for each firewall.

Once the wizard has run, the following command, run from the firewall machine, will show a list of information if the access to LDAP has been correctly configured:

```
ldapsearch -h <ldap server> -D <user> -w <password> -s base -b "" \
(objectclass=*) "
```

(All quotes are double quotes.)

If this does not produce a list of information, there is a problem with the LDAP connection.

We are now ready to start actually adding proxy users. Refer back to Figure 78. This shows the definition of Greg as a proxy user in the domain `firewall.domain.myco.com`. Prior to setting Greg up with his proxy user ID, it was necessary to define him as a Policy Director user as shown in the following (Figure 80):

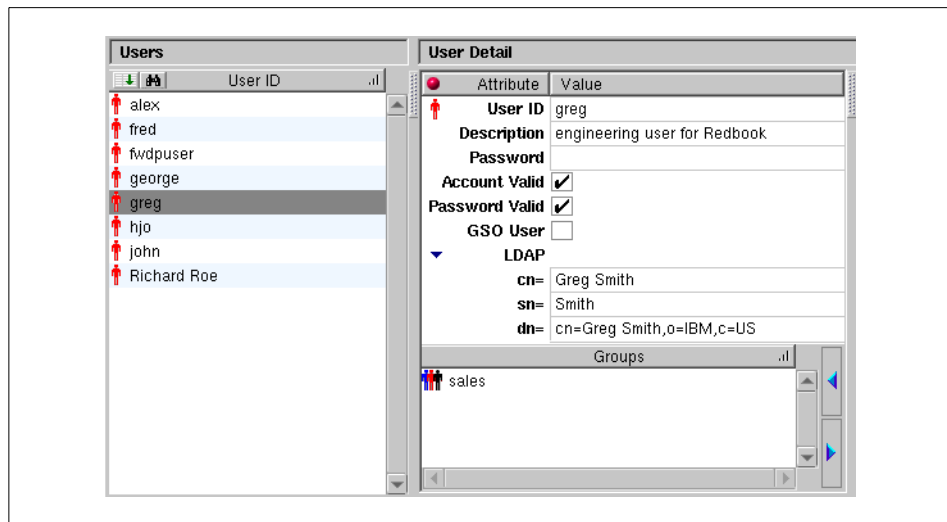


Figure 80. Initial user setup detail

So, to define a proxy user, you must first define a Policy Director user, ensuring they have the relevant LDAP information in their definition.

Once you are in the Proxy User screen, the first field to define is the *proxy domain*. It is here that you determine whether this user will be enabled to use multiple firewalls or only one. Note that we called Greg's proxy domain `firewall.domain.myco.com`. If we had instead used `domain.myco.com`, and had used this on several firewalls when configuring them through the SBS Wizard,

Greg would now be authorized as a proxy user on multiple firewalls. This is why it is important to plan what users will be accessing what firewalls prior to carrying this out. It is possible to save considerable administrative overhead by the use of a common Proxy Domain name.

10.2 SecureWay Boundary Server

This section elaborates on aspects on the options on how the SecureWay Boundary Server (SBS) integrates in our hypothetical scenario. The following figure, Figure 81, highlights the components that constitute the SBS.

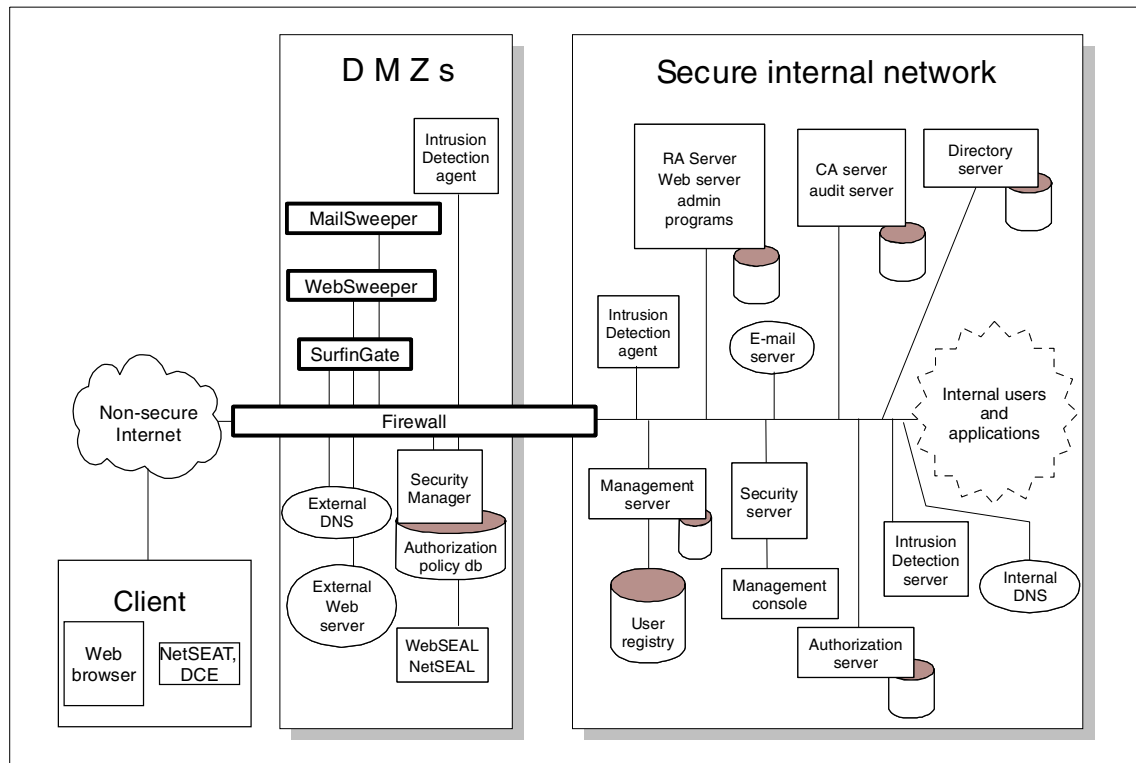


Figure 81. Secureway Boundary Server components

The SBS components that we see in Figure 81 are as follows:

- MIMESweeper, which is composed by WEBSweeper and MAILsweeper
- The IBM Firewall
- The Secure Boundary Server to attach the IBM Firewall to an IBM SecureWay Directory server

- SurfinGate to protect the internal network from malicious JavaScript, ActiveX, and Java code.

You should bear in mind that the field of network design, firewall implementation, and network design is in continuous evolution. Please refer to Appendix C, “Basic firewall and name service design” on page 327, for some general guidelines on firewall and network design issues.

In the previous figure, Figure 81, we see different DMZs all connected to a firewall. Do not assume the firewall is just one stand-alone machine; it can also be a cluster of machines because of performance considerations and also because of high availability considerations. Think of it as a firewall complex, which can be build on only one machine or possibly more.

As discussed in Appendix C, “Basic firewall and name service design” on page 327, depending on the level of security being sought, the system administrator needs to separate each individual type of services on different DMZ zones so that if one service has a security hole, it will not allow an intruder to exploit other services in another DMZ. DMZs that might be considered are shown in the following figure:

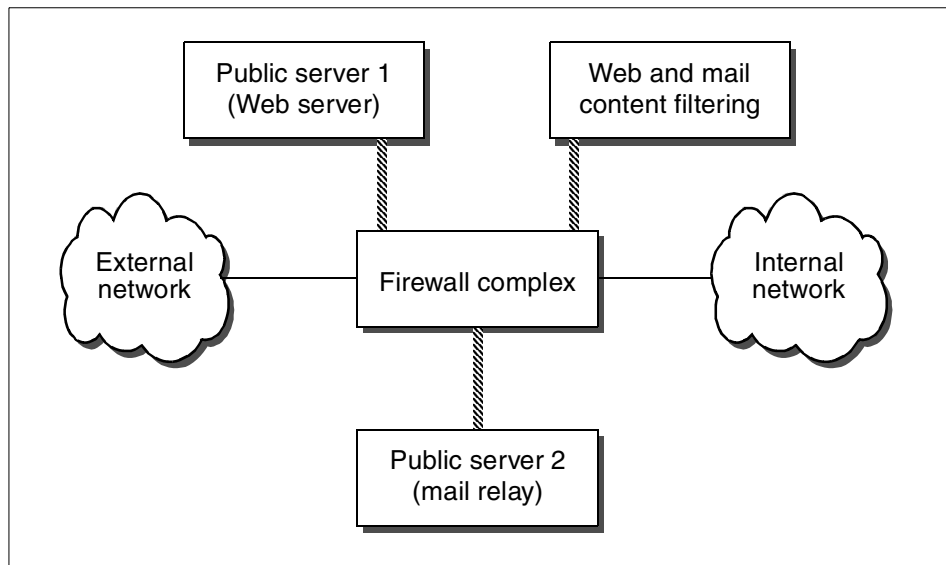


Figure 82. Modern firewall environment

The DMZ being shown in Figure 82 are as follows:

- Web servers and Web proxies.

- Mail relays where very simple SMTP relay software might operate, thus, allowing for sending and receiving of SMTP mail, with little or no processing of the mail itself.
- Mail/Web/FTP content filtering software. In this DMZ, WEBSweeper, MAILsweeper, and SurfinGate can be installed. SMTP, HTTP, and HTTPS traffic between this DMZ and the two other DMZs mentioned above should be allowed to pass through the firewall. From this DMZ, filtered e-mail can then be delivered to the internal e-mail servers, and the HTTP flow can be delivered to the clients' machines or to some other HTTP cache proxy down the chain.

Installing an HTTP cache proxy down the chain can be considered, so that when a cache hit is done on the HTTP cache server, the data on it has already been inspected and does not need to be analyzed again. In general, such chains are built so that they provide for better performance if the HTTP cache servers are as close to the client machines as possible.

10.2.1 A simple scenario

Let us consider the following scenario with only one DMZ (Figure 83):

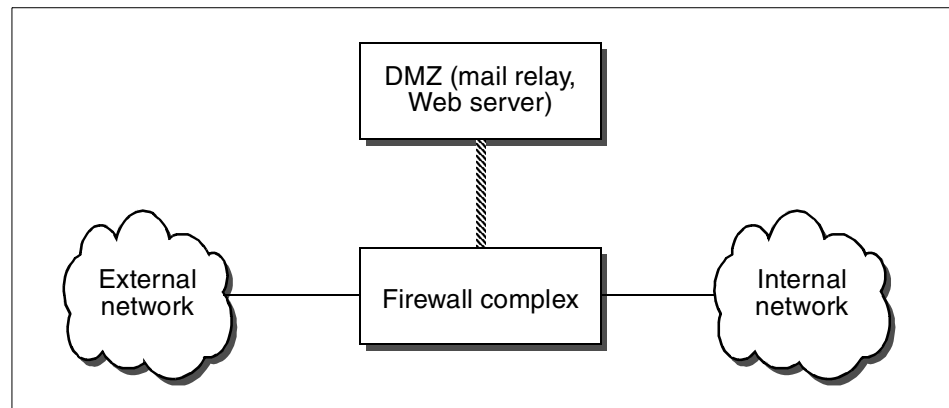


Figure 83. A simple firewall scenario

Having only one DMZ means that all the Internet-related servers, such as mail relays, Web servers, FTP servers, and the like, are very likely to be in that DMZ. SMTP relays will need inbound and outbound traffic on TCP port 25; Web servers will need to be contacted on TCP port 80 and 443 if HTTPS is going to be used, and FTP uses TCP ports 20 and 21. DNS queries from these servers should be resolved either by a local DNS server or the ISP's DNS server; so, the necessary DNS flow should also be allowed.

You can install MAILsweeper in that DMZ or even make MAILsweeper your inbound SMTP relay. Usually, the final mail server, where the mail is finally stored for users for later retrieval, is installed on the secure internal network.

WEBSweeper and SurfinGate can also be installed in the single DMZ, along with any other HTTP proxy, if you plan to chain them. The IBM Firewall, for example, includes an HTTP proxy.

There are some security issues in such a simple scenario as also discussed in Appendix C, “Basic firewall and name service design” on page 327. If an outsider gets unauthorized access to any of those servers, for example, by exploiting a bug on either server, then all the other servers in this DMZ will also be exposed. This is because the attacker will potentially be able, from that machine, to listen to all the network traffic in the DMZ, look for data, retrieve passwords sent in clear text, change parameters of the server, and so forth.

If you choose to install WEBSweeper, MAILsweeper, and SurfinGate on the internal network, then they are at risk from attacks from the internal network. An attacker might then switch, for example, MAILsweeper for an application that actually infects e-mails, or WEBSweeper and/or SurfinGate for applications that infect files being downloaded using HTTP.

Be aware that once a vulnerability on a certain product is found, it does not take long until a program is developed that makes it very easy, even for the normal users, to exploit a bug on a given server. Further information can be found, for example, at:

<http://www.insecure.org>
<http://www.rootshell.com>.

If you plan to use IBM SecureWay Directory to define the firewall’s proxy users, then you should allow access to LDAP services between the firewall and the IBM SecureWay Directory server.

10.2.2 A more sophisticated scenario

A safer scenario can be created by separating the servers that are being used for different functions as shown in Figure 84. This scenario assumes that if an attacker taps into one Web server, he or she might be able to break into all the other Web servers without having to “listen” on the network in order to get passwords to access the other Web servers. With this setup, if an outsider breaks into a Web server, the damage will not get to a mail server, or other servers, since they are connected to different physical networks that are protected by the firewall.

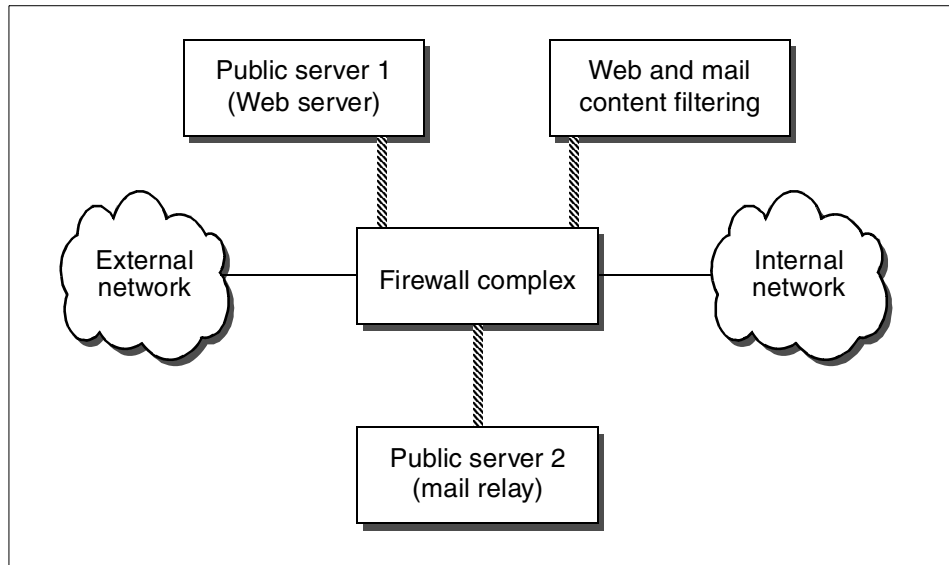


Figure 84. Sophisticated firewall environment

Between the simple scenario and this latest one, there are several mixed solutions. The design should group together servers with similar functionality within one DMZ. The good points and bad points should be analyzed and evaluated for each solution.

10.2.3 Systems administration

Another important topic is how system administration can be done in different DMZs. There are security issues involved, as client/server tools that do not encrypt their data stream using secure protocols should not be used over non-secure networks. Whenever possible, a directly attached console for each server should be used. Consider also those tools that allow for secure communications over non-secure networks (tools that use SSL, SSH). If necessary, the firewall needs to be configured to add some rules for certain IP communications to be allowed between the servers to the systems administrator(s) workstations.

There is also a need for planning based on the kind of users and services the network is going to serve. An example of a possible scenario is shown in Figure 85:

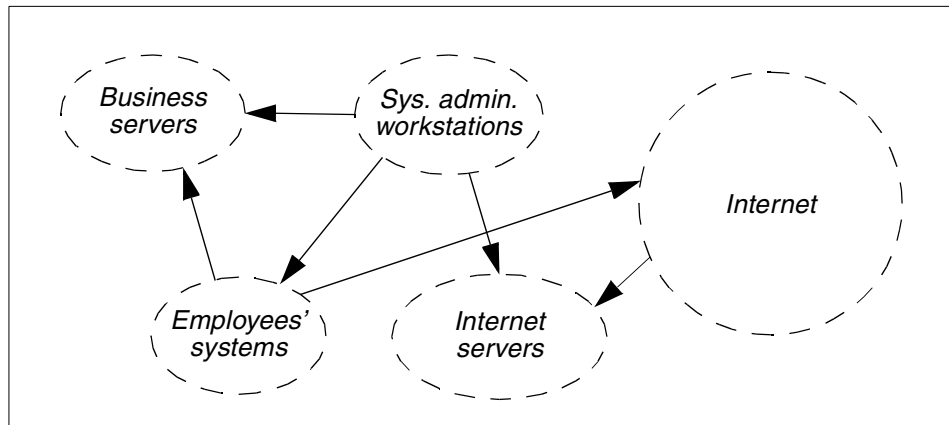


Figure 85. Required interconnections

At the network planning phase, considerations about the boundaries of the different networks involved and the connections that the system administrators would use to perform their work should be taken into account. In some cases, when maximum security is required, the system administrator network might even be a network zone of its own.

10.2.4 Boundary services on the internal network

So far, we have always considered the network boundary between an organization's secure intranet and the external, non-secure Internet. A second firewall complex on the intranet might be considered to separate groups of machines with different levels of security as shown in Figure 86.

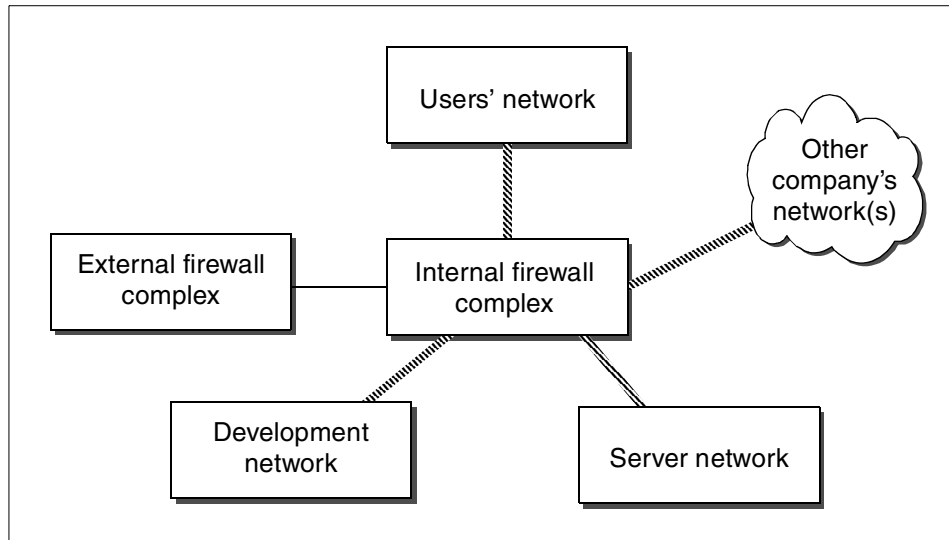


Figure 86. Intranet with an internal firewall complex

In Figure 86, the internal firewall complex separates different networks, each with different levels of security. Let us assume that an organization is running some ERP software (Enterprise Relationship Management, see <http://www.ibm.com/erp>). The firewall setup could be such that only users situated on the users' network are allowed to connect to the ERP servers located on the servers' network but only using the IP ports and protocols that this application uses. The firewall, at the same time, should deny all other traffic, such as NFS, SNMP, ICMP, SMTP, FTP, Telnet, and so forth. System administrators can then continue to use less secure protocols, such as NFS or plain Telnet sessions (without encryption), inside the servers' network without fear as long as the servers' network can be physically secured against outsiders tapping into it.

One good reason for such a multi-DMZ setup is that, in general, encryption takes time and CPU cycles because of the required processing. For large data transfers that may be required between servers, or for remote backup operations, the use of encryption might take too much time or might require specialized hardware to achieve a certain operation in a given period of time. If such a servers' network is secured by a firewall, non-secure protocols could safely be used within that network.

On an organization's internal network, the following utilities are candidates to be deployed as well:

- HTTP analyzers, such as WEBSweeper and SurfinGate, to check for potential malicious code on internal Web servers, or if users on the network are able to browse other companies internal Web sites by means of business-to-business connections.
- SMTP analyzers, such as MAILsweeper, if SMTP is being used as the standard e-mail protocol.
- Specialized mail analyzers for internal e-mail server. If the electronic mail system that the organization uses does not use SMTP by default, such as Lotus Notes, then you need some software other than MAILsweeper for SMTP to analyze that traffic. This kind of software, such as MAILsweeper for Domino, will prevent users from sending electronic mail to others with infected files attached.

Keep in mind that many of the security issues involved in networking also come from attacks originating from the internal network. As an example, there are many packages on the Web that allow a user to put a network adapter of his or her workstation in what is called *promiscuous mode*. By doing so, the user can “listen” on this workstation to all the network traffic that passes through the physical network segment to which the workstation is attached. One way to avoid such exposures is to use switched networks, which may involve a major financial investment in networking equipment. Another way to avoid such problems is to use adequate encryption and secure protocols on the applications. This is not always possible; so, it might be a good idea to build different physical networks such that network traffic can be held locally. Such networks can be connected through routers, or, if security is a concern, through firewalls.

In general, keep your network tight towards the outside but also keep a watchful eye on the inside because a considerable number of attacks are launched from inside a corporate network.

10.3 Intrusion Immunity

Intrusion Immunity (II) encompasses two components: *Norton AntiVirus Suite* and *Tivoli Cross-Site for Security*, as introduced in Chapter 6, “Intrusion Immunity (II)” on page 151. The Norton AnitVirus (NAV) Suite is a comprehensive set of desktop, server, gateway, and administrative software that has its place on literally any Windows-based computer in an organization (UNIX and other operating systems have, in comparison, not been the target of successful attacks). In collaboration with gateway (firewall) components and administrative services, NAV Suite provides for maximum virus protection as long as it is used on every possible system, and the virus signature files

are kept up-to-date. The NAV media package included with IBM SecureWay FirstSecure contains decent documentation about the various components and how to deploy them in real life scenarios. Support for NAV is provided exclusively through Symantec Corp. at:

<http://www.symantec.com>

In the following sections, we discuss the options of intrusion detection as provided by Tivoli Cross-Site for Security.

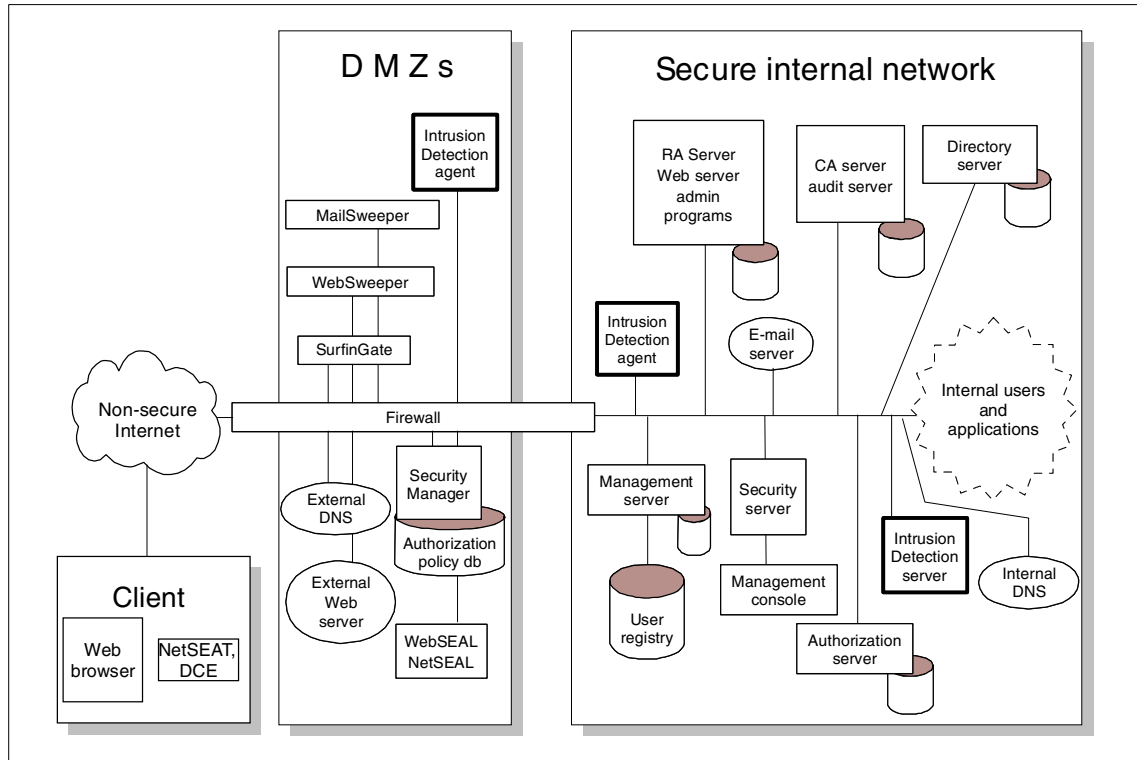


Figure 87. Intrusion Immunity within FirstSecure

The diagram above (Figure 87) shows two Cross-Site agents (highlighted) located in the secure area and in the DMZ that are monitored by a Cross-Site server located within a secure area.

10.3.1 Locating intrusion detection components

Although simplified, the layout shown in Figure 87 represents some typical principles, which are:

- The Cross-Site Server is located within the secure area. Security-related servers and functions, such as the Cross-Site server, should never be exposed to external traffic, thus, exposing the risk of getting compromised.
- The Cross-Site Management Console, not shown in Figure 87, is typically located anywhere in the secure area, typically on administrators' workstations. Note that there can be more than one Management Console. It could, however, also be located in the DMZ or even on the external network. The latter, however, poses several risks and should only be considered when absolutely necessary, for example, when systems administrators need to conduct some work over the Internet.
- Cross-Site Agents should be placed wherever intrusion detection is desired, which is definitely the case in the DMZ. If multiple DMZs are used, each DMZ should have a Cross-Site Agent. If DMZs, or dedicated server networks, are deployed within the secure networks (as discussed in 10.2.4, "Boundary services on the internal network" on page 292), these should be considered good candidates for Cross-Site Agents.
- As discussed in Chapter 6, "Intrusion Immunity (II)" on page 151, the scope of detection is largely reduced if switched networks are being used. If this is the case, Cross-Site Agents may need to be placed very specifically on those systems that need to be monitored. Most, if not all, servers in the DMZs are primary candidates for such monitoring. But also servers within the secure network might be considered to be monitored individually if they are connected to a switched network.

The various options as listed above, have some technical implications, which are discussed next.

10.3.2 Technical implications

The Tivoli Cross-Site for Security runs as a WebSphere application, that is, some of its functionality, namely the communication components, is implemented as Web server servlets. This has tremendous advantages because the network does not need to support and deal with additional protocols other than HTTP and HTTPS, respectively. In fact, the Cross-Site Agents and the Management Console only use HTTP/HTTPS to connect to the Management Server. Furthermore, the design is such that only the agents initiate connections to the server, much the same as Web browsers access Web servers.

Having this in mind, it is fairly straight forward to set up any firewalls between Cross-Site Agents and the Management Server. If only the secure HTTPS is used, which is recommended, then a firewall needs to only allow traffic to the

Management Server on one single port initiated from a known set of clients (agents and/or consoles).

Thus, if we consider a sample scenario as shown in Figure 88, where two agents reside in different DMZs, the firewall rules between these DMZs and the secure internal network need only be:

Allow traffic initiated by 'A' or 'B' to port 443 of the Management Server.

This example assumes that HTTPS uses the default port 443, but any other available port can be configured.

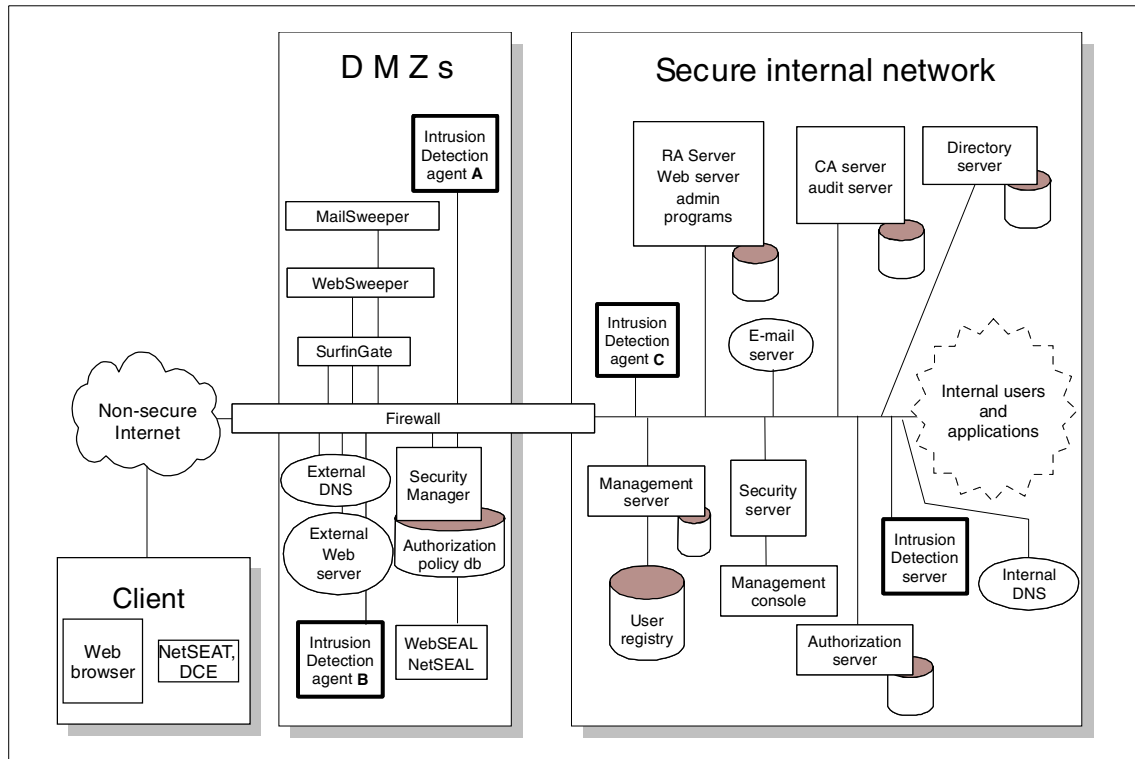


Figure 88. Cross-Site Agents and firewalls

Bear in mind that the Cross-Site Agents in the DMZs need to be protected from any attacks. The firewall (and the agent systems themselves) should not allow any access from other systems within the DMZ or from the external Internet. For maximum security, any remote access should be disabled as long as local administration on the attached console is feasible.

If there is a requirement that a system administrator can access the Management Server using the console connected to the external Internet, the firewall must be configured to allow traffic initiated from that client to a single port on the Management Server. For better security and to further limit the risk of attacks, the (external) client should have a permanently assigned IP address. Such a setup is, however, in violation with some basic firewall design rules and should only be considered in low-risk environments.

If configured as such, the Cross-Site server may generate e-mail and/or TEC messages. These will most likely stay within the secure network; so, no further consideration is necessary. Only when such messages are to be forwarded through the non-secure Internet, must the necessary path be enabled.

10.4 Trust Authority

A typical Trust Authority scenario within an organization's network infrastructure is shown in Figure 89.

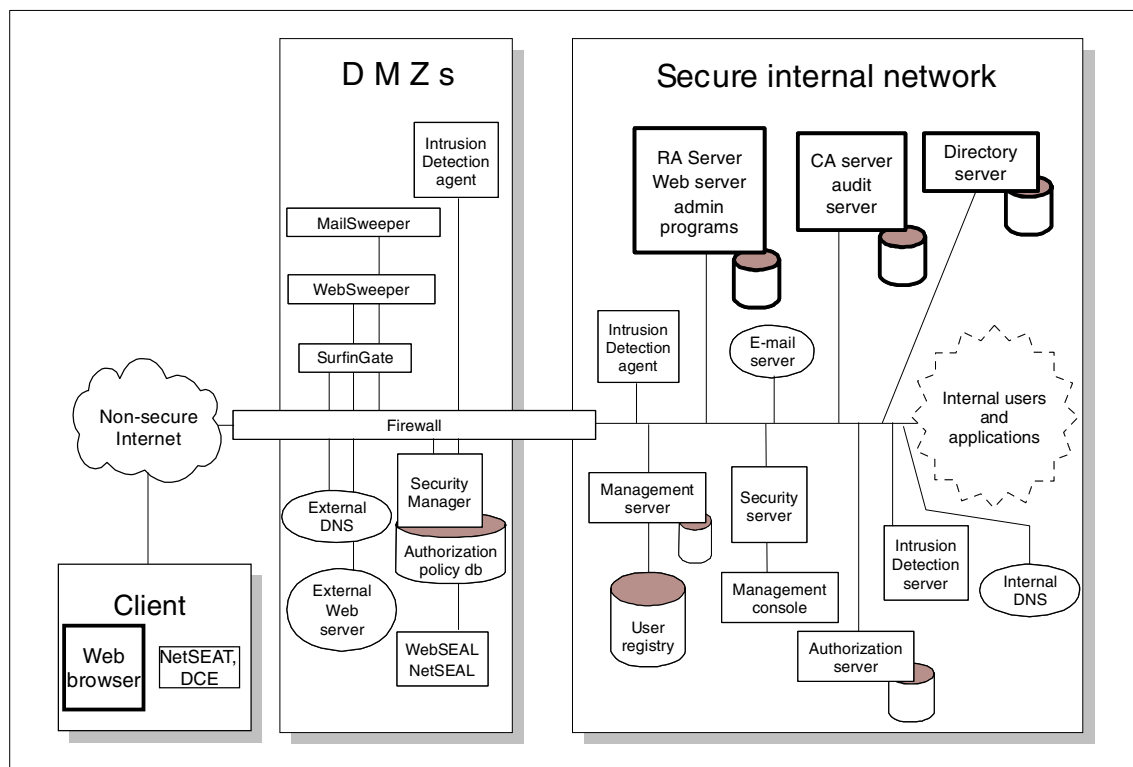


Figure 89. Trust Authority within FirstSecure

10.4.1 Locating Trust Authority components

As can be seen in Figure 89, Trust Authority is installed on the secure intranet because, no doubt, it is a critical element for security. The installation in the example scenario includes a *CA server*, an *RA server*, and an *LDAP directory server*. Although all Trust Authority functions could reside on one single system, the example layout provides for best scalability. Please refer to 9.5.1, “System environment” on page 260 for more details on how Trust Authority can be set up on more than one machine.

Administrators using the *RA desktop* could work anywhere on the organization’s intranet. If required (though for security reasons not recommended), administrators could also use the RA desktop from the external Internet.

TA clients, much the same as RA desktops, can be deployed anywhere on the organization's intranet as long as connectivity to the Trust Authority RA server is provided.

Since a Trust Authority installation is a critical link in an organization's security chain, it might be a good idea to locate them on a separate network that is connected to the organization's intranet through a firewall. This is discussed in 5.2, "Defining network boundaries" on page 122 and 10.2.4, "Boundary services on the internal network" on page 292. By doing this with a proper firewall setup, the risk of system intrusion can be lowered significantly. This also lowers the burden of hardening the operating system (see 9.1.1, "Hardening the operating systems" on page 218) down to a level where normal system administration becomes cumbersome because most commonly used services are disabled.

If a separate network is not used, it is certainly wise to apply some minimal guidelines for hardening the operating systems of the Trust Authority servers.

10.4.2 Connectivity implications

Connectivity between the components of Trust Authority is not normally of any concerns. Only when the TA servers are installed on an isolated network that connects to the organization's internal network through a firewall (see previous section), the firewall must be set up such that it allows Trust Authority traffic to pass. For the configuration of the firewall, it is important to know the destination ports on which connections are established.

In a Trust Authority environment, the following ports are used between the clients (Web browsers, RA desktop, and TA client) and the servers:

- 80** HTTP (non-secure) used by Web clients when connecting to the RA server for certificate requests.
- 81** HTTPS (secure) temporarily used by the Setup Wizard during installation and configuration to connect to the RA server.
- 443** HTTPS (secure, server authentication) used by Web clients to connect to the RA server for certificate enrollment status inquiries and certificate downloads.
- 829** Secure connection used by the TA client to connect to the RA server.
- 1443** HTTPS (secure, server and client authentication) used by the RA desktop to connect to the RA server.

Note

The port numbers listed above are the *default* numbers assigned during the installation. The installer has the option to change these numbers.

In addition to the ports listed above, but likely less important for the setup of a firewall, the CA, RA, Audit, and LDAP servers use other ports among themselves to communicate with each other. Their default numbers are:

1830 For the CA server

59998 For the Audit server

389 For the LDAP server (not using SSL)

636 For the LDAP directory server (using SSL for secure communication)

If it becomes important that external users on the Internet must have access to the Trust Authority systems, then there are two options:

Transparent firewall – The firewall between the intranet and the Internet could be configured to transparently pass traffic from the Internet to the RA server. Note that this is in violation with some security guidelines, but it might be considered as an option. It should only be considered when the external system has a fixed IP address (to allow tighter firewall rules) and when Trust Authority is not considered a high risk for the organization's security. This solution can be used, for example, to provide access to the RA for an administrator to manage the Trust Authority remotely. It should not be used (for security reasons) for a large number of users.

RA server in a DMZ – The RA server could be placed into a DMZ between the intranet and the Internet. Because a Web server is always considered at risk, the RA server should not be on the same DMZ as any Web server. This could be a solution when a relatively large number of clients need to connect to the RA server as, for example, when an organization offers the creation of certificates to their clients over the Internet. This is depicted in Figure 90.

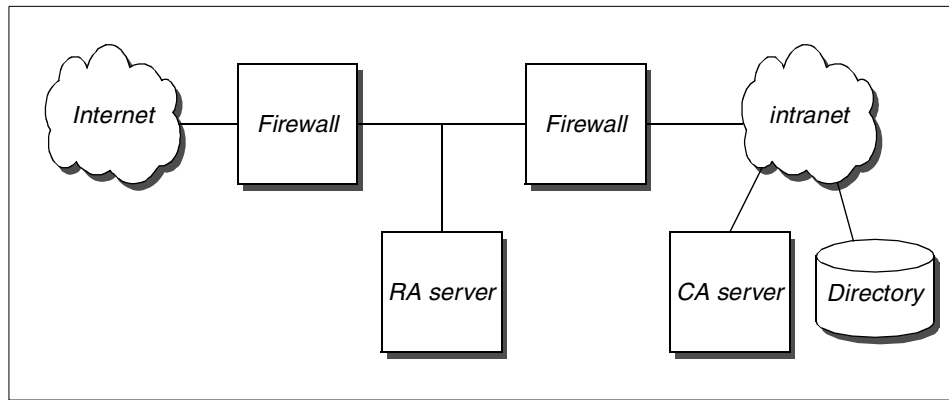


Figure 90. Trust Authority for external users

For an installation as shown in Figure 90, the firewall on the left must be transparent for traffic originating on the Internet to the well-known ports on the RA server (80, 443, and 1443), while the firewall on the right must allow traffic from the RA server to the CA server (port 1830), the Audit server (59998), and the LDAP directory server (390 or 636). For administration and user access from the intranet, the firewall on the right must also be transparent for traffic originating from the intranet to the RA server on ports 80, 443, and 1443.

For reasons of administration and separation of business processes, such an installation as discussed above will not likely be used at the same time for an organization's internal security domain (although it technically could). Also, e-business application may provide a similar service to external clients and then handle the certificate management internally on the intranet. In this case, an external client would not have to connect to the RA server directly.

10.4.3 More integration

Looking at Figure 89 on page 299 again, there are points of integration with other building blocks of IBM SecureWay FirstSecure. This section discusses points of integration and possible considerations.

10.4.3.1 Monitoring for security

Since the TA servers are critical for security, proper monitoring should be considered. Cross-Site Agents (see 6.2, "Tivoli Cross-Site for Security" on page 153) can passively monitor the network segments where the Trust Authority servers are installed. Possible intrusion attacks can be detected and counteractions be initiated. Alternatively, Cross-Site Agents could be installed

on each CA and RA server to specifically monitor these machines. If the TA servers are installed behind a firewall and the Cross-Site Server is located on the other side of that firewall, HTTP/HTTPS traffic (ports 80 or 443) must be allowed to pass (remember that the Cross-Site Agents use HTTP/HTTPS to communicate with the server).

10.4.3.2 Certificate authentication with Policy Director

Policy Director supports certificate based authentication (4.2, “Authentication and authorization” on page 61). This is done through SSL certificates that can be created with Trust Authority. With the proper customization of the CAS (Credentials Acquisition Server), Policy Director extracts a user’s information from a certificate and bases authentication and authorization on this information.

If Policy Director is being used for authentication and authorization, Trust Authority is an ideal supplement to provide convenient certificate based authentication. Two basic scenarios are briefly discussed:

Direct user access to Trust Authority – In this scenario, users apply for certificates by connecting directly to the RA server. If the users are external to an organization’s network, the RA server could be located in the DMZ as discussed in 10.4.2, “Connectivity implications” on page 300.

Indirect user access to Trust Authority – Through a separate process, either manual or by means of an application integration, users can get their certificates. Trust Authority is involved, but not by a direct connection from the user. An advantage of this solution may be that the RA server does not need to be exposed in a DMZ but can be installed more securely on the secure intranet.

10.4.3.3 SSL certificate management

SSL connections are commonly used to connect to secure Web sites when sensitive information is involved. While using server certificates signed by well-known CAs offers convenient advantages for the users, they have some disadvantages for the server administrator, such as cost and (sometimes) unnecessary delay for issuing.

But server certificates may not only be used for a small number of large-scale Web servers. It becomes more and more common that secure HTTPS (or other SSL-based) connections are used in client/server applications. Tivoli Cross-Site is such an example where the clients (called *agents*) communicate with their servers by using secure HTTPS. It is only a small step further when client authentication among such services is implemented by default. This requires a significant number of certificates, and it might not be appropriate

any more to request them from external providers. In fact, if an organization has a Trust Authority environment in place, for example, to provide its employees with certificates for e-mail and authentication purposes, it adds little to also issue SSL certificates for the various non-human communication (assuming the employees use their personal certificates for their SSL initiation).

10.4.3.4 Virtual Private Networks (VPN)

IPSec certificates are among the certificates that are supported by Trust Authority by default. If an organization uses IPSec connections, TA certificates offer an easy, inexpensive, and quick way to get certificates for these connections.

10.4.3.5 Cross-certification

Rather than having each participating party install the root certificate from a new Trust Authority installation, it might be a desirable option to cross-certify the Trust Authority CA with other CAs. An organization might, therefore, consider to cross-certify its (proprietary) Trust Authority CA with a well-known CA, such as Verisign or Entrust. Also, multiple Trust Authority security domains (CAs) can be cross-certified within an organization (or any business-dependent units) to simplify administration.

10.4.3.6 Directory exploitation

The use of an LDAP directory to publish certificates and/or certificate revocation lists (CRLs) is a very powerful use of a directory. An LDAP directory does not need to be exclusively used and updated by Trust Authority; it can (and shall) serve many other purposes as well. Applications can retrieve user and certificate and/or revocation information from a directory to get timely information about the status of users and their certificates. This way, applications have an independent source of information and can reliably check the validity of certificates.

Appendix A. Introduction to cryptosystem

A *cryptosystem* is a combination of three elements: An encryption-decryption algorithm (also called *cipher* or *encryption engine*), *keying information*, and *operational procedures* for their secure use. This appendix gives you an overview of cryptosystems explaining the services and mechanisms they provide and how they are related to a Public Key Infrastructure (PKI).

A.1 Basic terminology

Cryptography is the science of encryption, the transformation of data into a form unreadable by anyone without a secret decryption key. A *cryptosystem* or *cipher system* is a method of deliberately scrambling messages so that only certain people can descramble and read them. Cryptography can also be described as the art of creating and using cryptosystems. *Cryptanalysis* is the opposite of cryptography – it is the science of cracking code, decoding secrets, violating authentication schemes, and in general, breaking cryptographic protocols. The various techniques in cryptanalysis attempting to compromise cryptosystems are generally referred to as *attacks*. Attacks may originate from real attackers (or hackers), but also from scientists that try to prove or disprove the strength of an algorithm. While modern cryptography is growing increasingly diverse, cryptography is fundamentally based on problems that are difficult to solve. A problem may be difficult because its solution requires some secret knowledge, such as decrypting an encrypted message or signing some digital document. The problem may also be difficult because it is intrinsically difficult to complete, such as finding a message that produces a given hash value.

In cryptographic terminology, an unencrypted message is called *plaintext* or *cleartext*. Encoding the contents of the message in such a way that hides its contents from outsiders is called *encryption*. The encrypted message is called the *ciphertext*. The process of retrieving the plaintext from the ciphertext is called *decryption*. Encryption and decryption usually make use of a *key*, and the coding method is such that decryption can be performed only by knowing the proper key.

Modern cryptography consists of two categories of protection: Secret key (or symmetric key) and public key (or asymmetric key) cryptography. *Public key systems* allow a cryptosystem to be maintained by a group of users, each with a different key. These keys are combined to decrypt the data so that if one key is misplaced or compromised, the data are not lost. *Secret key cryptosystems* use a single key to encrypt *and* decrypt data. It is very secure

for an individual or small organization, but ineffective for a larger organization because the number of keys to be maintained increases exponentially with the number of parties involved. Cryptography deals with all aspects of secure messaging, authentication, digital signatures, electronic money, and other applications. *Cryptology* is the branch of mathematics that studies the mathematical foundations of cryptographic methods.

A.2 The goal of a cryptosystem

A cryptosystem, in general, provides four services: Confidentiality, integrity, authentication, and non-repudiation. In order to implement these services, different techniques are being used (see Table 44).

Table 44. *Cryptosystem matrix*

Service	Mechanisms
Confidentiality	Encryption and decryption
Integrity	Hashing and digital signatures
Authentication	Digital signatures
Non-repudiation	Digital signatures

The purpose of *confidentiality* is to hold sensitive data in confidence, limited only to an appropriate set of individuals or organizations. Information is not disclosed to anyone who is not authorized to use it. Confidentiality requires an individual to be authenticated (identified) and authorized to have access to a given piece of data. *Integrity* is to protect data against corruption, modification, or forgetting. *Authentication* is the process of reliably determining the identity of a communicating party, while *non-repudiation* is the scheme in which there is proof of who sent a message such that the author of a message cannot deny having sent this message.

As stated earlier, these four basic services can be implemented by using specific technical mechanisms: Encryption and decryption functions, hashing functions, and digital signature.

Encryption is the mechanism that provides confidentiality protection. Several methods (algorithms) can be used to encrypt and decrypt data streams. Encryption can be relatively easy to implement, while decryption without the proper key(s) is practically impossible.

Encryption and decryption allow two communicating parties to disguise the information they send to each other. Encryption is the process of transforming information so that it is unintelligible to anyone but the intended recipient(s).

This intermediate data stream is often referred to as *ciphertext*. Decryption is the process of transforming encrypted information so that it is intelligible again. A cryptographic algorithm (cipher) is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption. With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is usually widely known, but on a number, called a *key*, that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. The key must be known in order to decrypt encrypted text. Decryption without the correct key is very difficult and, in some cases, impossible for all practical purposes. This is due mainly to the amount of work and computational resources required by the *cryptanalyst* in order to obtain a successful *attack*.

The integrity service, known also as *tamper detection*, allows the recipient of information to verify and ensure that it has not been modified in transit. Any attempt to modify data or substitute a false message for a legitimate one can be detected. The mechanism used most to provide this service relies on a mathematical function called a one-way hash (also called a *message digest*).

The other two security services are *authentication* and *non-repudiation*. Authentication allows the recipient of information to determine its origin, that is, to confirm the sender's identity. Non-repudiation prevents the sender of information from claiming, at a later date, that the information was never sent.

These security services are provided by the *digital signature* mechanism. Digital Signature Algorithm (*DSA*) is a popular public-key technique, though it can only be used for signatures, not encryption.

The need for more than one cryptographic service can be easily pursued by merging the mechanisms. The next sections describe the commonly used mechanisms in more detail.

A.3 Basic cryptographic algorithms

A method of encryption and decryption is called a cipher. Some cryptographic methods rely on the secrecy of the algorithms. Such algorithms are only of historical interest and are not adequate for real-world needs. All modern algorithms use a key to control encryption and decryption; a message can be decrypted only if the key matches the encryption key. The key used for decryption can be different from the encryption key, but for most algorithms, they are the same.

There are two classes of key-based algorithms, symmetric (or *secret-key encryption*) and asymmetric (or *public-key encryption*) algorithms. The difference is that symmetric algorithms use the same key for encryption and decryption (or the decryption key is easily derived from the encryption key); whereas, asymmetric algorithms use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key.

A.3.1 Secret-key encryption algorithm

With symmetric-key encryption, the same key is used for both encryption and decryption as illustrated in Figure 91.

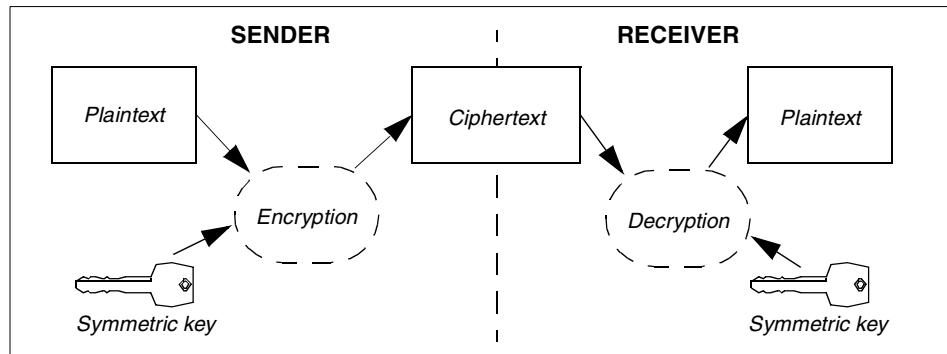


Figure 91. Symmetric-key encryption

The same key is used by both the sender and the receiver. The main challenge in symmetric-key cryptosystems is getting the sender and the receiver to agree on the secret key without it being disclosed to anyone else. This needs a secure transmission of the key before the secure path can be established using this key. A method is required by which the two parties can communicate without fear of eavesdropping. Another disadvantage of symmetric-key encryption is that each participating partner has to securely maintain a list of keys for each communication partner, which easily turns into a management nightmare as individuals might have hundreds of communication partners. However, the advantage of symmetric-key encryption is that it is relatively fast because it does not involve much computing power.

Symmetric-key encryption plays an important role in the widely used Secure Sockets Layer (SSL) protocol, which is used for authentication, tamper detection, and encryption over TCP/IP networks. But, SSL also uses techniques of public-key encryption, which is described next.

A.3.2 Public-key encryption algorithm

Asymmetric encryption involves a pair of keys (a public key and a private key) associated with an entity (for example, a person) that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published and, therefore, widely known, and the corresponding private key is kept secret. Data encrypted with a public key can only be decrypted with the corresponding private key. Data signed with a private key can only be verified with the corresponding public key (the terms *sign* and *verify* are used in this context instead of *encryption* and *decryption* as it will be explained in A.6, “Digital signatures” on page 316). In general, when the only service needed is confidentiality, the sender encrypts the data with the receiver's public key, and the receiver decrypts it with his or her corresponding private key (see Figure 92). For the digital signature mechanism, see Figure 93 on page 314.

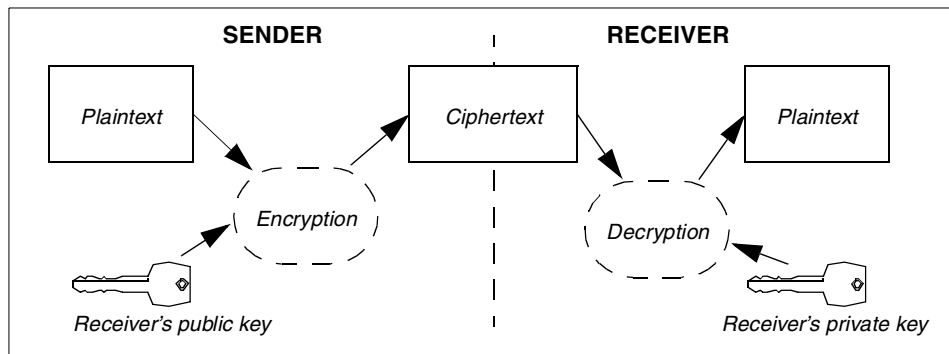


Figure 92. Asymmetric-key encryption

The popular *Pretty Good Privacy* (PGP) public key encryption is outlined at the following URL:

<http://web.mit.edu/network/pgp.html>

To read more about the *Rivest Shamir and Adleman* (RSA) encryption algorithm, see the URL:

<http://www.rsa.com>

This method employs an *asymmetrical* key. The name arises because the encryption key, or *public key*, is significantly different from the decryption key, that is, the *private key*. In this case, attempting to derive the private key from the public key involves a large computational effort.

Compared with symmetric-key encryption, public-key encryption requires considerably more computation and is, therefore, not always appropriate for large amounts of data. However, it is possible to use public-key encryption to encrypt and, thereafter, exchange a symmetric key that can subsequently be used to encrypt additional data. This scheme is called *Digital Envelope*, and it is the approach used by the popular SSL protocol.

A.3.3 Strength of cryptographic algorithms

Good cryptographic systems should always be designed so that they are as difficult to break as possible. It is possible to build systems that cannot be broken in practice, though this cannot be proved. This does not significantly increase system implementation effort; however, some care and expertise is required. There is no excuse for a system designer to leave the system breakable. Any mechanisms that can be used to circumvent security must be made explicit, documented, and brought to the attention of the end users.

In theory, any cryptographic method with a key can be broken by trying all possible keys in sequence. If using brute force to try all keys is the only option, the required computing power increases exponentially with the length of the key. A 32 bit symmetric key takes 2^{32} (about 10^9) steps to try all possibilities, but there is a (very) low chance that the correct key is found only after the first few iterations. This can be done in reasonable time using only a Personal Computer. A system with 40 bit keys (for example, the US-exportable version of RC4) takes 2^{40} steps, requiring a level of computing power readily available at most universities and even small companies. A system with 56 bit keys, such as DES, takes a substantial effort, but is breakable in reasonable time with special hardware. The cost of the special hardware is substantial but within reach of organized criminals, major companies, and governments. Keys with 64 bits are probably breakable now by major governments and will be within reach of organized criminals, major companies, and lesser governments in a few years. Eighty-bit keys may become breakable as available computing power increases, but 128-bit keys will probably remain unbreakable by brute force for the foreseeable future. Even larger keys are, of course, possible. However, key length is not the only relevant issue. Some ciphers can be broken without trying all possible keys. It must also be ensured that the algorithm that is used to generate such a random key is really capable of creating any possible key, not just a subset.

In general, it is very difficult to design ciphers that cannot be broken into. Designing new ciphers may be fun, but it is not recommended in real applications, as it has been proven that they are generally easy to break. One should generally be very wary of unpublished or secret algorithms. Quite often, the designer is then not sure of the security of the algorithm, or its

security depends on the secrecy of the algorithm. Generally, no algorithm that depends on the secrecy of the algorithm is secure.

It is relatively easy to hire someone who can disassemble and reverse-engineer software algorithms. Experience has shown that a vast majority of the secret algorithms that were broken and have become public knowledge later turned out to be pitifully weak in reality.

The key lengths used in public-key cryptography are usually much longer than those used in symmetric ciphers. There the problem is not that of guessing the right key, but deriving the matching secret key from the public key. In the case of RSA, this is equivalent to factoring a large integer that has two large prime factors. Other cryptosystems are based on other problems. To give some idea of the complexity, for the RSA cryptosystem, a 256 bit modulus is easily factored by people with limited computer resources. 384 bit keys can be broken by university research groups or companies. 512 bit keys are within reach of major institutions. Keys with 768 bits are probably not secure in the long term. Keys with 1024 bits and more should be safe for now unless major algorithmic advances are made in factoring; keys of 2048 bits are considered by many to be secure for decades. Most systems on the market nowadays support either 512 or 1024 bit key lengths.

No matter how complex the key is, it should be emphasized that the strength of a cryptographic system is usually equal to its weakest point. Therefore, no aspect of the system design, from the choice of algorithms to the key distribution and usage policies, should be overlooked.

A.4 The hashing function

Cryptography, as discussed above, provides for confidentiality of information. To guarantee integrity, hashing is used. A hash function compresses the bits of a message to a fixed-size hash value in a way that distributes the possible messages evenly among the possible hash values. A cryptographic hash function does this in a way that makes it extremely difficult to come up with a message that would hash to a particular hash value.

Cryptographic hash functions typically produce hash values of 128 or more bits. This number is vastly larger than the number of different messages likely to ever be exchanged in the world.

A *hash function* H (see Figure 93) is a transformation that takes an input m and returns a fixed-size string that is called the hash value h (that is, $h = H(m)$). The basic requirements for a one-way hash are:

- The value of the hash is unique for the hashed data (the input can be of any length, and the output has a fixed length).
- $H(x)$ is relatively easy to compute for any given x .
- $H(x)$ is collision free: It is infeasible to find using computation any two messages x and y such that $H(x) = H(y)$.
- Any change in the data, even deleting or altering a single character, results in a different hash value.
- The content of the hashed data cannot, for all practical purposes, be deduced from the hash. This is called “one-way-transformation”.

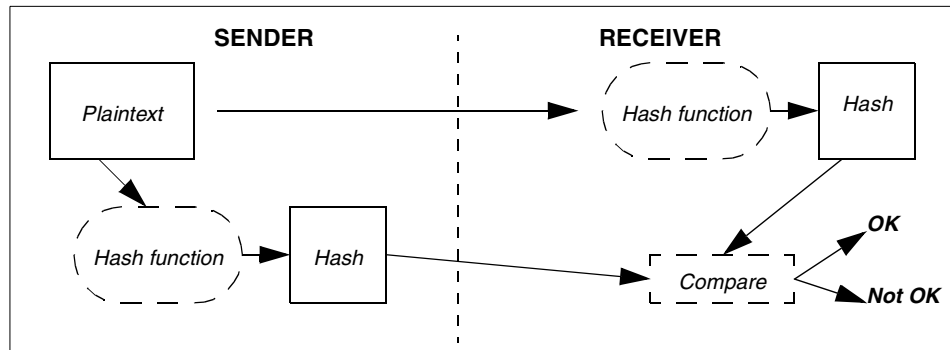


Figure 93. Hashing function - Integrity check

Cryptographic hash functions are typically used to compute the message digest when making a digital signature (see A.6, “Digital signatures” on page 316 below).

A.5 Merging confidentiality and integrity functions

The needs of the two cryptographic services described up until now (confidentiality and integrity), can be easily pursued by merging the two mechanisms: Encryption and hashing together (as shown in Figure 94).

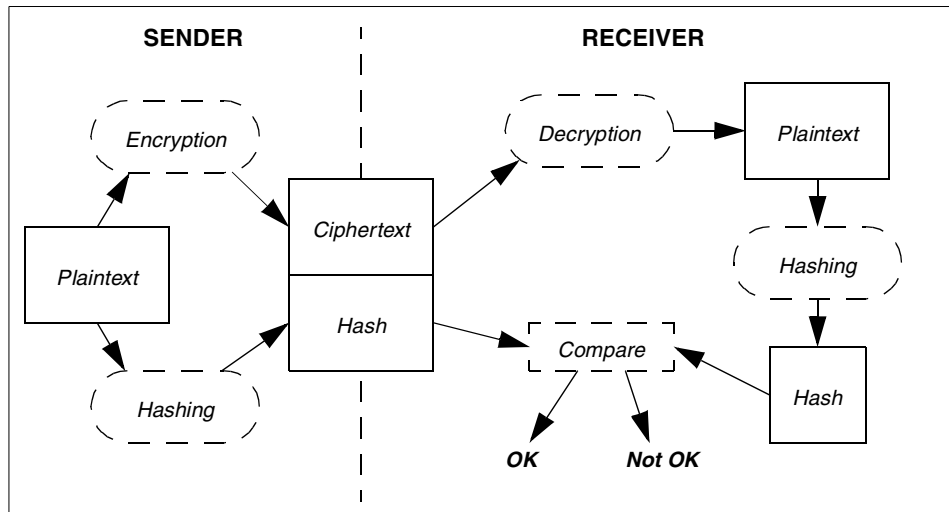


Figure 94. Combining confidentiality with integrity services

This way, confidentiality and integrity are guaranteed as long as the sender and receiver are in possession of their partner's public keys. But how do they know that the public keys they use are really the ones from their partners? If this cannot be guaranteed, an intruder could step in between and publish two fake public keys, one for the sender and another for the receiver, respectively. This attack is called masquerading. The sender would then use a false public key to encrypt the message, and the receiver would use a false public key to decrypt the hash. This makes both parties believe that they have securely exchanged their messages, while, in fact, the third party intercepted the transmission in between, decrypted it (with its own private key), read or even change its contents, encrypted it again (with the real receiver's public key), and forwarded it to the receiver. Similarly, the hash could have been regenerated to make the receiver believe that the message has not been altered.

Taking this threat into consideration, it is most essential for public key cryptography that the public keys are distributed (published) in such a way that there is a mechanism that allows verification that a public key belongs to the correct owner. This is one of the primary purposes of *certificates*, which are explained in A.7, "Digital certificates" on page 318.

A.6 Digital signatures

Public-key algorithms are frequently used to generate *digital signatures*. A digital signature is a block of data that was created using the private key of the signer, and there is a public key that can be used to verify that the signature was really generated using the corresponding private key. The algorithm used to generate the signature must be such that, without knowing the secret key, it is not possible to create a signature that would verify as valid.

Digital signatures are used to verify that a message really comes from the claimed sender (assuming only the sender knows the secret key corresponding to his/her public key). They can also be used to timestamp documents: A trusted third party signs the document and its timestamp with its secret key, thus, testifying that the document existed at the stated time.

Digital signatures can also be used to testify (or certify) that a public key belongs to a particular person. This is done by signing the combination of the public key and some relevant information about its owner by the private key of a trusted third party. The reason for trusting that key may again be that it was signed by another trusted key (see A.7, “Digital certificates” on page 318).

A digital signature of an arbitrary document is typically created by computing a message digest from the document and concatenating it with information, such as the name of the signer and a timestamp. The resulting string is then encrypted using the private key of the signer using a suitable algorithm. The resulting encrypted block of bits is the signature. It is often distributed together with information about the public key that was used to sign it. To verify a signature, the recipient first determines whether it trusts that the public key belongs to the person it is supposed to belong to (using the web of trust or a prior knowledge) and then decrypts the signature using the public key of the person. If the signature decrypts properly, and the information matches that of the message (proper message digest, and so on), the signature is accepted as valid.

Several methods for making and verifying digital signatures are freely available. The most widely known algorithm is RSA.

The following, Figure 95 on page 317, explains the digital signature mechanism.

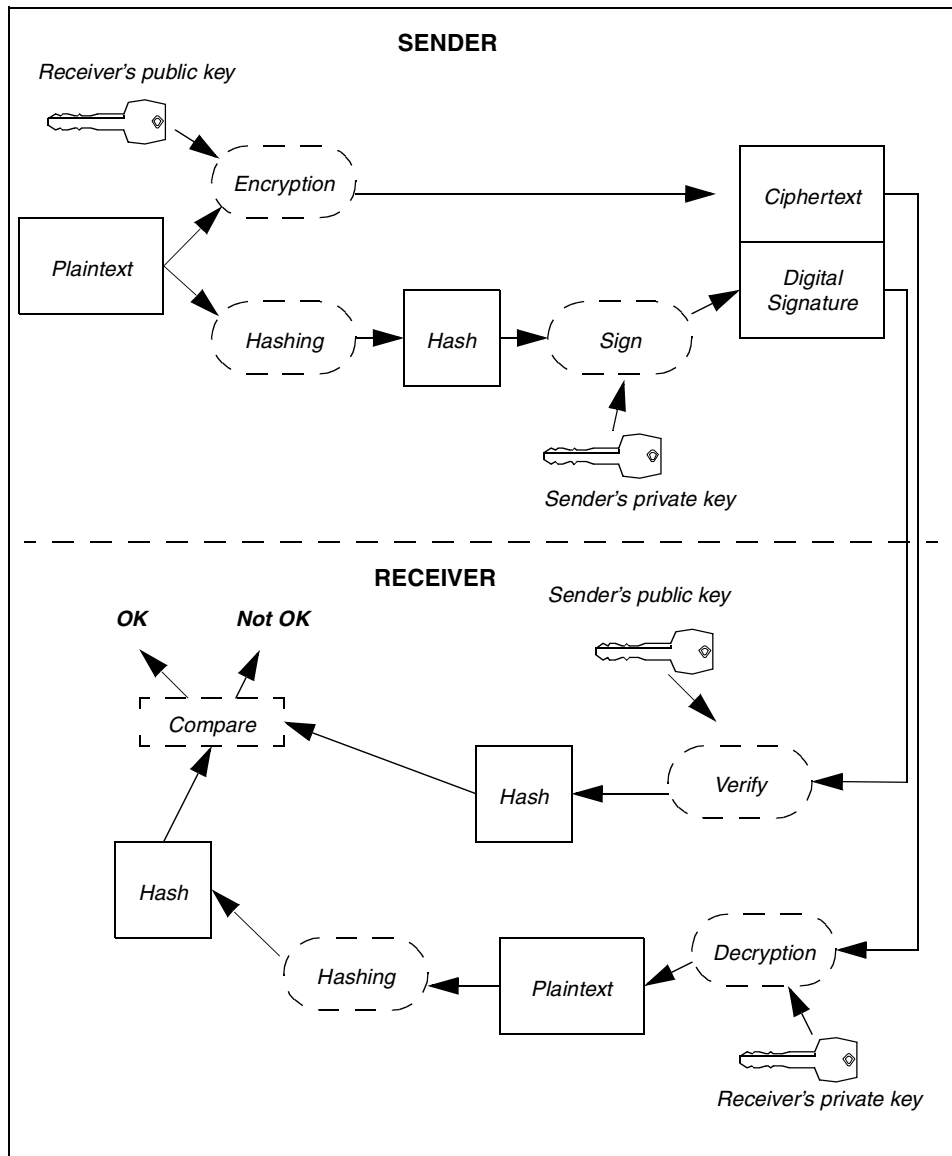


Figure 95. Digital signature mechanism

The significance of a digital signature is comparable to the significance of a handwritten signature. Once the data has been signed, it is difficult for the sender to deny doing so later, assuming that the private key has not been compromised or out of the owner's control. This quality of digital signatures

provides a high degree of non-repudiation. In other words, digital signatures make it difficult for the signer to deny having signed the data.

A.7 Digital certificates

Public keys are generally published in the form of a *certificate*, which is an electronic document used to identify an individual, a server, a company, or some other entity and to associate that identity with a public key. Certificates are required because they validate public keys. They are typically used to generate confidence in the legitimacy of a public key. They are issued by a Certificate Authority (CA, see 2.4.2.1, “Certificate Authority (CA)” on page 28), which can be any trusted party that assures the identity of those to whom it issues certificates and their association with a given key.

In its simplest form, a certificate may contain only a public key, the name and the digital signature of the certificate owner. It may also contain information about the key expiration date, the name of the certifying authority that issued the certificate, a serial number, and other optional information (called optional extensions).

The emerging standard for certificates is the X.509v3 certificate format (see Figure 96), which has been recommended by the International Telecommunications Union (ITU), an international standards body since 1988 and which is part of the OSI group of standards.

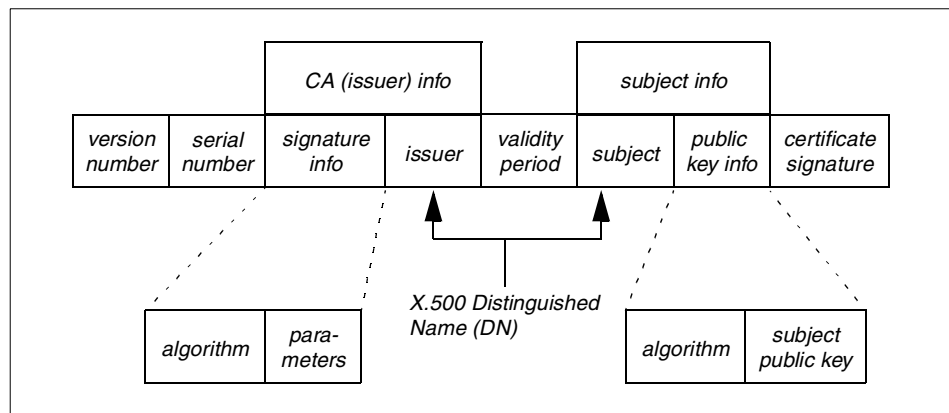


Figure 96. X.509v3 certificate format

X.509v3 certificates are defined using a notation called ASN.1 (Abstract Syntax Notation 1), which specifies the precise kinds of binary data that make up the certificate. ASN.1 can be encoded in many ways, but the emerging

standard is a very simple encoding called DER (Distinguished Encoding Rules), which results in a compact binary certificate (see Figure 97).

```
Certificate ::= SEQUENCE {
  tbsCertificate TBSCertificate,
  signatureAlgorithm AlgorithIdentifier,
  signatureValue BIT STRING }
TBSCertificate ::= SEQUENCE {
  version [0] EXPLICIT Version Default v1
  serialNumber CertificateSerialNumber,
  signature AlgorithmIdentifier,
  issuer Name
  validity Validity,
  subject Name
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
  (If present, version shall be v2 or v3)
  subjectUniqID [2] IMPLICIT UniqueIdentifier OPTIONAL,
  (If present, version shall be v2 or v3)
  extensions [3] EXPLICIT Extensions OPTIONAL,
  (If present, version shall be v3)
}
Version ::= INTEGER { v1(0), v2(1), v3(2)
```

Figure 97. X.509v3 certificate definition in ASN.1 notation

Every X.509v3 certificate consists of two sections, a data section and a signature section:

- The *data section* includes the following information:
 - The version number of the X.509 standard supported by the certificate.
 - The certificate's serial number. Every certificate issued by a CA has a serial number that is unique among the certificates issued by that CA.
 - Information about the user's public key including the algorithm used and a representation of the key itself.
 - The Distinguished Name (DN) of the CA that issued the certificate.
 - The period during which the certificate is valid (for example, between 1:00 p.m. on November 15, 1999 and 1:00 p.m. November 15, 2000).
 - The DN of the certificate subject (for example, in a client SSL certificate, this would be the user's DN), also called the subject name.
 - Optional certificate extensions, which may provide additional data used by the client or server. For example, the certificate type extension indicates the type of certificate, that is, whether it is a client SSL certificate, a server SSL certificate, a certificate for signing e-mail, and so on. Certificate extensions can also be used for a variety of other purposes.

- The *signature* section includes the following information:
 - Information about the cryptographic algorithm, or cipher, used by the issuing CA to create its own digital signature.
 - The CA's digital signature, obtained by hashing all of the data in the certificate together and encrypting it with the CA's private key.

If the key specified in the certificate has been compromised, or the user specified in the certificate no longer has the authority to use the key, a certificate might need to be revoked and be put in a Certificate Revocation List (CRL). A CRL consists of a list of certificates that have been revoked before their scheduled expiration time. This list is maintained by a CA, and it is usually published at well-publicized distribution points, such as an entry in a public directory or a Web site.

X.509v3 processes create time-stamped CRLs for all certificates. Each time a certificate is used, X.509v3 capabilities allow the application to check the validity of certificate. It also allows the application to determine that the certificate is not on that CRL. X.509v3 CRLs can be constructed on the basis of a specific validity period.

A.8 Cryptosystem as a base for a PKI

Public key cryptography requires a Public Key Infrastructure (PKI), essential services for managing digital certificates and encryption keys for people, programs, and systems.

PKI is the term generally used to describe the mechanisms, entities, policies, and relationships that are employed to retrieve cryptographic keys and to reliably associate public cryptographic keys with their owners.

In its most simple form, a PKI is a system for publishing the public-key values used in public-key cryptography. There are two basic operations common to all PKIs, *certification* and *validation*:

- *Certification* is the process of binding a public-key value to an individual, organization or other entity, or even to some other piece of information, such as a permission or credential.
- *Validation* is the process of verifying that a certificate is still valid.

A PKI, in addition to the bare cryptographic algorithms and mechanisms, supports these important processes.

Appendix B. Introduction to LDAP

The term *LDAP* (which stands for Lightweight Directory Access Protocol) has been frequently used throughout this book. For example, the Policy Director stores its user information in an LDAP directory. This appendix gives you a short introduction to LDAP, if you are not familiar with its concepts and what it can provide. An LDAP server ships with IBM SecureWay FirstSecure. It is also available from IBM free (or at no additional cost) for various IBM and non-IBM operating system platforms.

B.1 The history

In 1988, the CCITT (Consultative International Telephonique et Telegraphique, which is now ITU-T, International Telecommunications Union - Telecommunication Standardization Sector) created a standard for directory services, the X.500 standard. It became ISO standard 9594, *Data Communications Network Directory, Recommendations X.500-X.521*, in 1990. This set of standards is still commonly referred to as *X.500*.

X.500 defines a directory that can universally be used for large amounts of data, and today it is typically used by national telephone organizations for their huge, online telephone directories.

To access an X.500 directory, a client uses the *Directory Access Protocol* (DAP) that was defined along with the X.500 standard. Unfortunately, DAP is a rather complex protocol, which cannot be easily supported on thin clients, such as desktop computers. X.500 was, therefore, limited to powerful computers and large-scale implementations. The requirement to access centralized directories from slim clients, however, became apparent as the value of centralized directories became obvious for various reasons, such as cost-effectiveness for management.

Work being done at the University of Michigan (UMICH) and at Netscape Communications Corp. has led to a simplified version of DAP, which was then called *Lightweight Directory Access Protocol* (LDAP). LDAP supports most of the features of DAP, but lacks some of the complex, seldom used functions. Its implementation is relatively simple, and it can, therefore, be used by desktop applications.

B.2 Protocol or directory?

As the name implies, LDAP—strictly speaking—is only a protocol that runs over TCP/IP. The LDAP protocol standard not only includes low-level network protocol definitions, but also data representation and handling functionality. A directory that is accessible through LDAP is, therefore, commonly referred to as an LDAP directory. It must be noted and understood, however, that the LDAP standard does, by no means, define how the data are actually stored in the directory.

Initially, LDAP was designed such that thin clients could access an X.500 directory through a gateway server that did some translation between LDAP and DAP (see Figure 98).

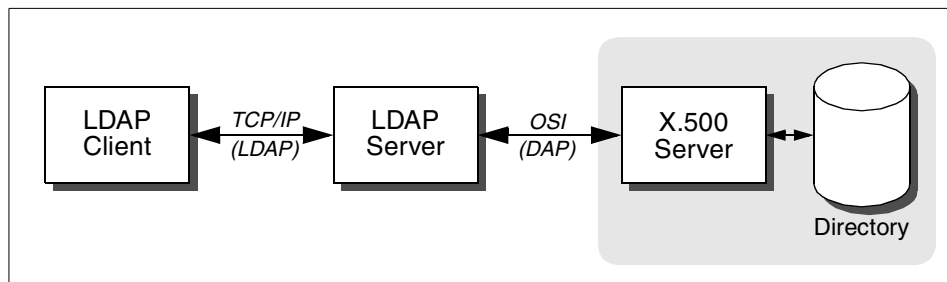


Figure 98. LDAP access to X.500

It did not take long before vendors developed directories that could handle the LDAP protocol natively rather than performing a translation between LDAP and DAP (Figure 99). The IBM implementation of an LDAP directory is the *SecureWay Directory*, which is available on AIX, Windows NT, Sun Solaris, OS/400, and OS/390.

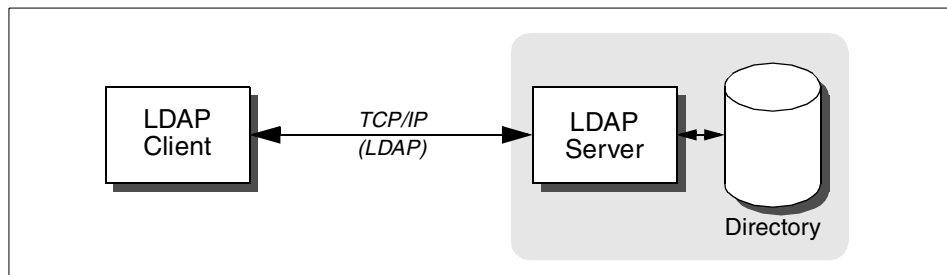


Figure 99. Stand-alone LDAP server

Vendors choose different implementations for the storage of the directory data. While most vendor implementations use flat file databases, the IBM SecureWay Directory uses the high-performance, highly scalable DB2 relational database as a backing store for its LDAP directory.

B.3 What is a directory?

The question of what a directory is has not been answered so far. To look at a simple example, most directories store information about people much the same way as a printed phone directory (white pages) does. The entries are usually organized in a hierarchical way that makes management and searching easy and intuitive.

LDAP directories are much more powerful and are, by no means, limited to name, phone number, and address entries. In fact, an LDAP directory can store (and subsequently retrieve) almost any kind of data. The type of data that can be stored in an LDAP directory is defined by the directory schema, which can be extended and adapted to meet certain requirements. The task of defining a directory schema and the directory information tree (the hierarchy in which entries are to be stored in the directory) can be compared to the design of a relational database. Thorough analysis of application requirements, corporate standards, and data definitions is necessary to design a directory schema and the directory information tree (DIT). Fortunately, LDAP server products, such as the IBM SecureWay Directory, come with a comprehensive schema that can (and should) be used unless specific requirements dictate alterations and/or additions.

Standards bodies work on standards and proposals for a large variety of possible data definitions and their hierarchical relationship. IBM supports these standards and proposals by actively participating in the respective organizations and by implementing the results in the IBM SecureWay Directory. The most important standards body for LDAP is the Internet Engineering Task Force (IETF) where representatives of IBM and other key industry leaders are actively supporting these activities.

Micro and macro directories are typically present in large numbers in every organization. For example, most modern operating systems, such as UNIX or Windows 9X/NT, store user account information either locally or on departmental servers. Network operating systems, such as NetWare (Novell), carry some sort of user databases, too. Departments may have their own “private” employee database, while at the same time, corporations have large human resource databases. Note that these examples only included information about people. In addition to this, operating systems store large

amounts of data about their system configuration and about other network resources, such as printers and servers.

Frequently, such information is even stored across multiple locations with multiple micro or macro directories, therefore, making administration and maintenance unnecessarily difficult.

A major reason why LDAP has quickly gathered so much interest is the potential for a single, standards based directory for distributed information. Most directory-related vendor products, including Novell NDS and Microsoft Active Directory, either already support, or have announced support for, LDAP. Even many widespread clients, such as Netscape Navigator or Microsoft Internet Explorer, have been providing support for LDAP since several product releases. Such client support of a Web browser ranges from the simple ability to access public LDAP directories for people lookup to the advanced ability to store personal preferences, including bookmarks, in a centralized LDAP directory to support travelling users.

Many public directory providers, such as Four11, that can be found at:

<http://www.four11.com>

offer public LDAP services available for use with any LDAP-capable client, such as Netscape Communicator.

B.4 The LDAP information model

The LDAP information model is based on a subset of the X.500 information model. Information in an LDAP directory is stored in *entries* that contain *attributes*. Attributes are typed in the form of *type=value* pairs in which the *type* is defined by an object identifier (OID), and the *value* has a defined syntax. Attributes can be *single-valued* (for example, a person can only have one date-of-birth) or *multi-valued* (a person can have multiple phone numbers).

Each entry in an LDAP directory has a unique *distinguished name* (DN). The directory schema defines rules for DNs and what attributes an entry must, and/or may, contain. To organize the information stored in directory entries, the schema defines object classes. An object class consists of *mandatory* and *optional* attributes.

Object classes can be inherited from other object classes, which provides a method for easy extensibility (for example, new object classes can be defined by just adding new attributes to existing object classes). The concept of

object classes, their different declaration and use, is beyond the scope of this short overview. See Appendix B.6, “Where to get more information” on page 326 for a list of selected LDAP reference documentation.

B.5 More LDAP features

This section lists and explains some additional features of current LDAP implementations that might not be obvious from the descriptions in the previous sections.

Scalability

LDAP directories, particularly when they are backed up by a relational database as in the IBM SecureWay Directory, are highly scalable. Large directories with millions of entries are possible with excellent performance. Due to the common standard base, another scalability factor is the easy step-up possibility to more powerful hardware and software. LDAP does not rely on a specific operating system and is vendor-independent.

Availability

LDAP supports replication and splitting of namespaces. Replication allows multiple LDAP servers to store the same directory contents, and, thus, clients have additional servers available in case one fails. Splitting allows parts of the whole directory (namespace) to be stored in different servers at different locations. This not only increases availability (no single point of failure) but also offers an easy way for distributed management.

Security

Security features are provided such that unauthorized access to data can be prevented. Secure communication protocols, such as SSL and authentication mechanisms, along with access control lists (ACLs) for data entries, guarantee a maximum level of security. This even allows for storage of sensitive data in an LDAP directory.

Manageability

Early versions of LDAP directory products came with command line tools, and configuration had to be done by editing configuration and schema files. Current versions, such as the IBM SecureWay Directory Version 3.1, facilitate a graphical user interface for both system administration and directory data administration. Dynamically extensible schema allows to extend the directory schema without interrupting the service.

Standardization

The LDAP protocol and many related client/server capabilities, application programming interfaces (APIs), and data definitions are defined by either

official standards or corresponding RFCs (Request for Comments). *Lightweight Directory Access Protocol (v3)*, RFC 2251, for example, defines the basic LDAP protocol. Other features are, although widely accepted and implemented, defined in Internet Drafts. Much of this work is done by the IETF (Internet Engineering Task Force) and the DMTF (Distributed Management Task Force).

B.6 Where to get more information

If you need more information about LDAP, consider these two IBM Redbooks:

- *Understanding LDAP*, SG24-4986
- *LDAP Implementation Cookbook*, SG24-5110

The IBM directory Web site can be accessed at:

- <http://www.ibm.com/software/enetwork/directory>

Other pertinent information can be found at:

- <http://www.ietf.org>
- <http://www.dmtf.org>
- <http://www.umich.edu/~dirsvcs/ldap>

Appendix C. Basic firewall and name service design

The SecureWay Boundary Server (SBS) takes a central place in an organization's network. It guards secure networks from potential threats on non-secure networks. The level of security provided by SBS depends largely on its implementation. Part of that implementation is the network layout and the naming service. This appendix explains some basic firewall and naming services as well as planning and design principles.

C.1 Introduction

The most basic firewall system is one that separates two different IP networks, for example, the Internet and an organization's internal network. All traffic between the two security zones must pass through the firewall system for it to be effective, as illustrated in Figure 100. The configuration of the firewall specifies which connections are permitted and which are not.

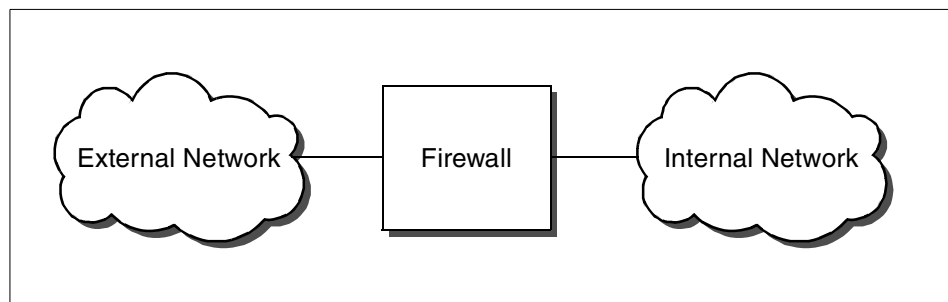


Figure 100. Simplest classic firewall

Different technologies can be used for controlling the traffic flow between the networks. Packet filtering checks individual IP packets, and proxies work on the level of connections and application byte streams. In modern firewall products, these techniques are often combined in a hybrid design that supports both techniques in some way.

It is important to keep in mind that a firewall is only able to check the traffic between the different attached networks. It cannot prohibit unwanted connections within one security zone. This fact can lead to major security risks. For example, if a company's public Web server is placed within the internal network (which is certainly not a good idea at all), the firewall needs to be configured to allow HTTP connections to this system so that everyone can get to the Web pages. If the Web server has security holes (for example, due to software bugs, configuration errors, non-secure dynamic content, or

any one of many other possible causes), an attacker may gain full access to the Web server system. The firewall cannot prevent the attacker from leveraging this to access other systems within one security zone.

Experience shows that it is not realistic to expect complex server software (such as Web servers) to be free of security holes. Major companies and government institutions have frequently been victims to these kinds of attacks. Everyday, new security holes are found and shared in the underground by hackers, and knowledge of this is delayed on public Internet sites, which can cause unknown security breaches. A public source for such information, for example, is:

<http://www.hackernews.com>

Placing important servers outside the firewall in the external network is not recommended either since they then cannot be protected by the firewall against attacks.

More security can be gained by introducing a perimeter network in which servers can be placed. This is known as a Demilitarized Zone (DMZ). The classical DMZ setup has two firewalls and a DMZ server network between them as shown in Figure 101.

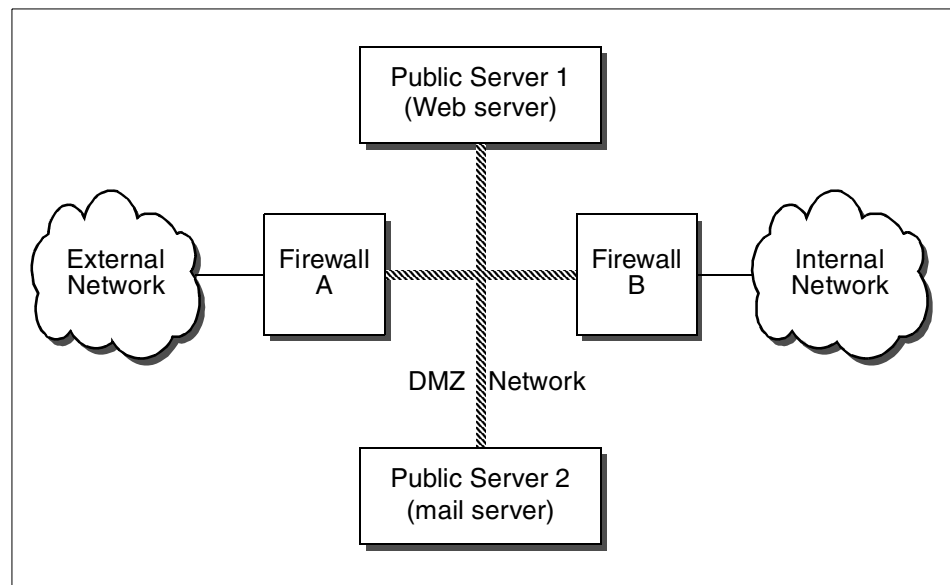


Figure 101. Classic DMZ and firewall design

The advantage of this setup is that the publicly accessible servers are now protected from the external network and also separated from the internal network.

The obvious disadvantage of this setup is that two firewalls are needed, which increases the cost, the complexity, and the administrative overhead, especially if different technologies are used for the two firewalls.

More importantly, in the worst case scenario, when Public Server 1 is broken into, more security is lost than necessary. For example:

- The intruder that broke into Public Server 1 can now freely attack Public Server 2 because there is no firewall between them.
- The intruder on Public Server 1 may be able to monitor all network traffic (including company e-mail and other possibly sensitive information when collected systematically) between Firewall A and Firewall B on the DMZ network. This technique is known as network sniffing.

The most frequently suggested approach to separate the systems in the DMZ is to use manageable switches or routers. A switch or router can be used to prevent network sniffing since packets are not sent to all attached systems by default. Access lists installed in switches or routers can also somewhat limit the kind of connections allowed between the computer systems attached to them.

However, as active and intelligent network devices, switches and routers may themselves be the target of attacks. An additional hardening process must be in place to improve and maintain the security of the switches and routers.

Switches and routers usually contain preloaded combinations of users and passwords that are defined at the factory, therefore, allowing for easy reconfiguration by a knowledgeable attacker if the local system administrator has not changed the passwords for these default users. Switches and routers might be configured by sending plain text (not encrypted) passwords over the network. These passwords can be easily captured, or even guessed, and are reusable. Many switches provide the ability to be configured using a dedicated RS-232 console, which should always be used in this case. The ability of the switch to be configured over the network should be disabled. Also, you can disable features, such as SNMP and remote management facilities, to further improve the security of these two components. Switches and routers can be used to provide additional filtering and alarming but, in few cases, should be relied on as a primary and dependable means of providing security to the business.

C.2 Compartmentalized firewall environment design

Another approach suitable for highly-complex environments is the compartmentalized firewall environment in which a single firewall system is equipped with more than two network interfaces and which can, therefore, mutually protect several different compartments (for example, DMZs or security zones) from each other.

Compartment is a new name for security zones that are protected from each other by one firewall. We chose it to differentiate this approach from the single two-firewall DMZ or secure server network.

The design that emerged in recent years, and may be considered state-of-the-art, looks similar to what is shown in Figure 102.

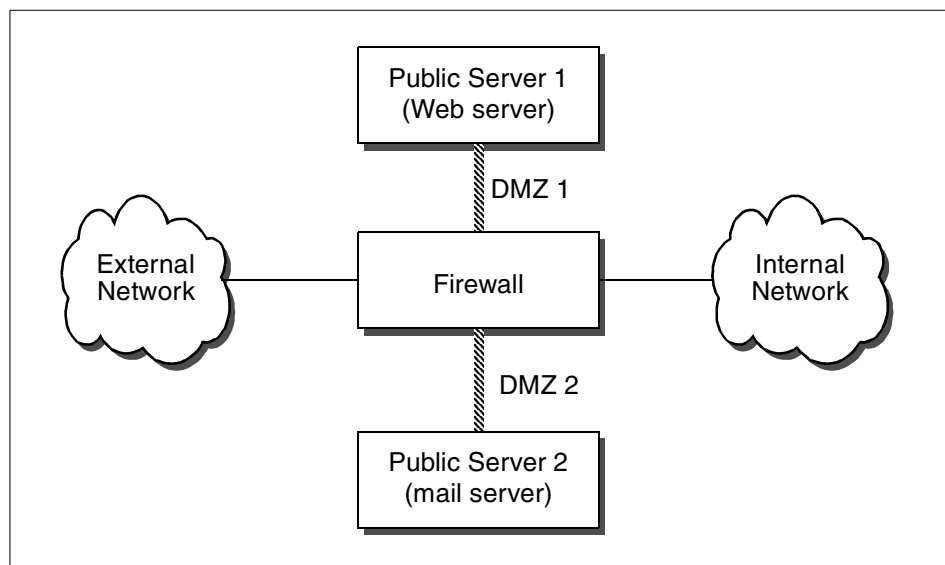


Figure 102. Modern firewall environment

The different compartments (Web, mail, and external and internal networks) each have their own physical network connected to the firewall through dedicated network adapters. The firewall is now able to control all the traffic between these compartments, and IP sniffing is also almost useless to an attacker because they can only see the traffic within the one compartment network they are able to break into. Since the compartments are independent, a security breach in one of the attached systems (for example, the Web server) does not lead to a total compromise of the environment. The damage is restricted within the network compartment of the affected server.

It is important to plan for this case whenever you install externally accessible servers because partial security breaches (successful attacks against one of the externally accessible servers) have happened to many and will continue to happen. The firewall system cannot prevent this, but it can make sure that an intruder will not be able to read e-mail just because the Web server has a security hole.

A properly configured firewall should generate an alert if an intruder tries to leverage more access from the attacked system, for example, by having the Web server try to access the mail server. One of the main functions of a firewall is to generate alarms when suspicious activity is detected (for example, the Web server connecting to the mail server) because no security device will ever be able to protect against all possible threats. It should alarm when possible attacks are being observed.

One small disadvantage of the compartmentalized approach is the somewhat higher complexity (more network adapters in the firewall and more routing issues), but the additional security is well worth the cost of the slightly higher networking complexity.

The real problem in this setup (as well as all other firewall designs) is that if the firewall is broken into, security is seriously compromised. Therefore, it is extremely important to make the firewall itself as secure as possible.

The operating system should be hardened and always up-to-date (see also 9.1.1, “Hardening the operating systems” on page 218). Operating system insecurities due to low integrity and quality and incomplete hardening have been major factors in many security breaches. Additional information can also be found at:

<http://www.ibm.com/security>

<http://www.cert.org>

It is recommended to plan ahead carefully before installing additional software on a firewall system. Only software that was explicitly designed, tested, and audited for use in a firewall environment should be considered for use. Installing server applications on separate systems often prevents possible issues caused by improper management of them.

Always keep the worst-case scenario in mind. A single software bug in the software usually enables the attacker to execute arbitrary binary code on the system that will eventually enable them to gain full control of the machine. Such a failure is reasonably harmless (because damage is limited to one compartment) if it happens on a separate server but disastrous if it happens on the main firewall system.

The firewall system described in Figure 102 should perform network traffic control and nothing else. Either IP filtering or secure proxies, or any combination of both, can be used for that purpose. Both have their own advantages.

Using IP filtering makes it very difficult to break into the firewall system because only IP packets are processed, and the task is carried by the kernel modules designed exclusively for that task.

Proxies that are designed exclusively for firewall use can protect against certain rare network-level attacks because new IP packets are generated by the operating system instead of forwarding the original IP packets, which could possibly be harmful.

The number of TCP or UDP server programs on the firewall should be kept to a minimum because those kind of programs are usually the weak spots that can be taken advantage of by a potential intruder.

The following sections list and explain good candidates for potentially separate compartments (server networks).

C.2.1 Mail servers

While most firewall systems contain SMTP gateways, separating the SMTP processes to another system can provide better flexibility and performance.

The *MAILsweeper* component of the SecureWay Boundary Server can be installed on these mail servers to receive mail from the Internet and then forward them to the corresponding internal mail servers after being checked. Anyway, it would be safer here to install a simple SMTP gateway that only receives the incoming mails and then forwards them to the MAILsweeper machines.

If there is a need for a normal SMTP gateway, be sure to choose securely designed mail products for incoming mail; outgoing mail might not be as critical.

C.2.2 Web proxies and servers

HTTP proxies can be used on a firewall system to supplement it. However, if you are more concerned about the performance of the Web proxy, separating the Web proxies, such as the IBM Web Traffic Express, on one or more dedicated servers on a separate network compartment may improve performance considerably. Flexibility is improved as the dedicated Web proxy products offer more functionality (caching to avoid repeated downloads,

filtering, authorization, and so on), and scalability is also improved since it is much easier to replace or upgrade a dedicated Web proxy.

It also improves administrative processes to have the server separate; for example, if you want to restrict outbound Web access, you might want to use the authentication mechanisms provided by the proxy software instead of the firewalls features because this is usually not so much a security as a internal control issue.

The administration of the Web proxy accounts should then be delegated away from the security administrator since those tasks are not really security related. The same principle applies to Web server pages that are protected with simple passwords. This is definitely a task for the Web server and not for the firewall, and the accounts should not be administrated by the security person either.

You might also consider installing more than just one Web server here if performance consideration indicate so.

C.2.3 Mail/Web/FTP content filtering and anti-virus proxies

Virus, content, and code checking of transferred files using *MAILsweeper*, *WEBsweeper*, *SurfinGate*, *Norton AntiVirus*, and other data laundering had better be conducted by servers in separate compartments (dedicated server networks). There is no good reason to integrate anti-virus proxies into the firewall. They can be treated just like standard mail/Web/FTP proxies.

If you have both anti-virus and standard proxies, you should set them up in such a way that the client talks to the normal HTTP proxy, which, in turn, gets data through proxy chaining from the anti-virus HTTP proxy. In this way, all pages get virus scanned only once before being cached, not every time they are requested.

C.2.4 Encryption devices

Encryption is getting more popular, and its function is different from a firewall because it ensures privacy and not necessarily security. For example, an e-mail encryption program typically will not prevent an encrypted e-mail from containing a macro virus.

Hardware encryption (for example, in encryption routers) is becoming more popular because it is faster than software encryption and can improve security by separating encryption from other security functions, which can be useful to extend separation of duties. An example of separation of duties would be if the firewall administrator did not know the encryption keys and

was not responsible for the support and maintenance of the encryption system since that would be the job of a separate person.

C.2.5 Remote access servers

It is probably a good idea to have the people that dial-in to the internal network be authenticated and monitored by the corporate firewall instead of allowing anyone, who might steal the right laptop, to have total, unaudited access to all internal resources.

C.3 The need for highly available firewalls

As we have discussed so far, the network configuration of a firewall system is getting more complicated than ever. A single firewall system can protect many systems, such as multiple Web servers, mail servers, and so on. The continuous availability of a firewall is becoming a critical factor for companies doing e-business on the Internet. If a firewall is down for any reason, customers lose access to business applications. Business assets could also be exposed to attacks by hackers if somebody disconnects the firewall and connects the organization's internal network directly to the Internet without any protection because the firewall is inoperable. Numerous business opportunities can be thrown away. Private data from customers might get exposed. Because a firewall system must be available to keep a company's business going 24 hours a day, 7 days a week, a high availability solution for the firewall system is more than a nice-to-have item.

However, we always need to remember that keeping network security is the area of most concern. Any high availability solution must be robust, dependable, and proven. One solution is the IBM HACMP program, and the other is the IBM SecureWay Network Dispatcher. To find out more about this item, please refer to *Highly Available IBM eNetwork Firewall Using HACMP or eNetwork Dispatcher*, SG24-5136, or search the ITSO Web site at:

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

C.4 Network address management

As mentioned previously, the distribution and management of network-layer addresses is very important. Addresses for networks and subnets must be well planned, administered, and documented. Because network and subnet addresses cannot be dynamically assigned, an unplanned or undocumented network will be difficult to debug and will not be scalable.

As opposed to the network itself, devices attached to the network can generally be configured for dynamic address allocation. This allows for easier administration and a more robust solution. The following section deals with the issues faced by technologies used in address management.

C.4.1 The Domain Name Service

The Domain Name System (DNS) is the method by which Internet addresses in mnemonic form that can be remembered more easily. Addresses, such as *www.ibm.com*, are converted into the equivalent numeric IP address that computers handle much better, such as *204.146.81.99*. To the user, and applications in general, this translation is a service provided either by the local host, another host on the internal network, or from a remote host via the Internet.

C.4.2 DNS name structure

DNS names are constructed hierarchically; the highest level of the hierarchy being the last component or *label* of the DNS name. If you consider the name *www.ibm.com*, then *com* will be the highest level hierarchy involved, telling the user that this name relates to an entity called IBM and that IBM is related to commercial activities.

There are standards that indicate what these highest level labels can be, and as the DNS was originally introduced in the United States of America, a three letter code indicated what type of organization that particular computer belonged to, as follows:

com	A commercial entity
edu	An educational institution
gov	A government entity
mil	A military entity
net	A network related organization
int	An international organization
org	Miscellaneous organization

Now, with the expansion of the Internet to the entire world, a two letter code that defines each country has been added to some of the network names of the machines, such as in *www.yourcompany.com.br*, if that company happens to be in Brazil, for example.

Because the average person cannot easily remember a multi-digit (in decimal form) IP address, some form of directory service is required in the network design to translate the more intuitive names into network addresses.

Increasing numbers of new applications also require host names, further reinforcing some form of name management in an IP network.

C.4.3 Static files

The simplest form of name resolution is through the use of static files on each host system where a mapping from a name to an IP address (and vice-versa) is made. This is specified in RFCs 606, 810, and 952. These RFCs defined the hosts.txt file used for the ARPANET. RFC 952 obsoleted the previous two. It specified the structure of the host names as they would be used in the ARPANET's host table.

An example that is often seen is the UNIX /etc/hosts file, although this file differs in its structure from that of the ARPANET's hosts.txt file. If this file exists, it will contain a listing of all the hosts the system requires to communicate with using the other host's host name. This listing supplies the host with each other host's host name and associated IP address.

The size of this file is directly related to the number of hosts a system requires name resolution for. In very small networks, this system works well, but as the network increases in size, this method becomes unmanageable.

C.4.4 The Domain Name System (DNS)

To solve the problems associated with the use of a static host file, the Domain Name System (DNS) was introduced. RFCs 1034 and 1035 are concerned with DNS.

The hierarchical approach of DNS allows for the delegation of authority and provides organizations with a level of control they required, while the distributed database eases the problems of the size of the database and the frequency of its updates.

DNS is made up of three major components:

- **The Domain Name Space and Resource Records** specify the hierarchical name space and the data associated with the resources held within it. Queries to the name space extract specific types of information from the records for the node in question.
- **Name Servers** are server programs that hold information about the name space structure and the individual sets of data associated with the resources within it.
- **Resolvers** are programs that extract information from the name servers in response to client requests.

We begin our discussion of DNS with a look at each of these elements.

C.4.5 The domain name space

The DNS name space is a distributed database holding hierarchical, domain-based information on hosts connected to a network. It is used for resolving IP addresses from host names. In addition to this service, it can also provide information on the resources available on that host, such as its hardware, operating system, and the protocols and services in use.

The name space is built in a hierarchical tree structure with a root at the top. This root is un-named and is delineated by a single period (.). The DNS tree has many branches. These branches originate from a point called a node. Each of these nodes corresponds to a network resource (a host or gateway).

We have called this structure the domain name space. But what exactly is a domain? A domain is identified by a domain name. It consists of the part of the name space structure that is at or below the domain name. Thus, a domain starts at a named node and encompasses all those nodes that emanate from below it. Let us look at an example (Figure 103).

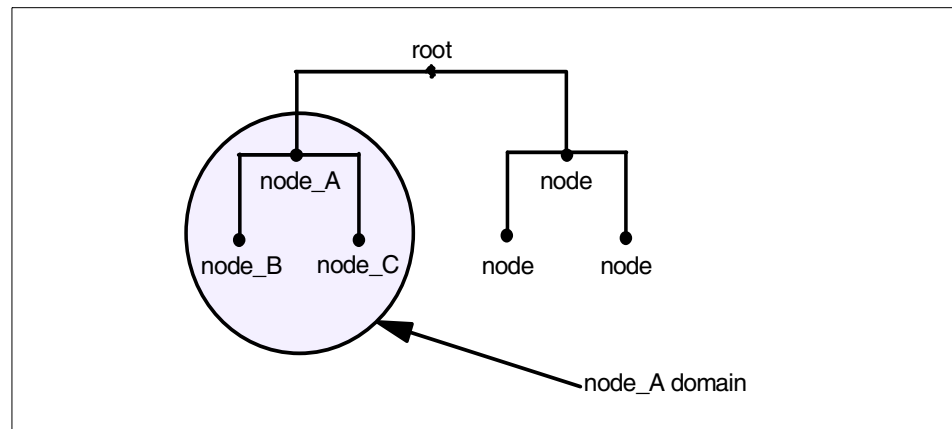


Figure 103. Example DNS domain

This figure shows a domain `node_A` that begins at `node_A`. It contains the nodes `node_A`, `node_B`, and `node_C`. This scheme may be taken a step further to show that, as we progress out from the root, we will create subdomains. Figure 104 illustrates this. A new domain, the `node_B` domain, contains `node_B`, `node_D`, and `node_E`. The original domain, `node_A`, now encompasses not only `node_A`, `node_B`, `node_C`, `node_D`, and `node_E`, but also the subdomain created by `node_B`.

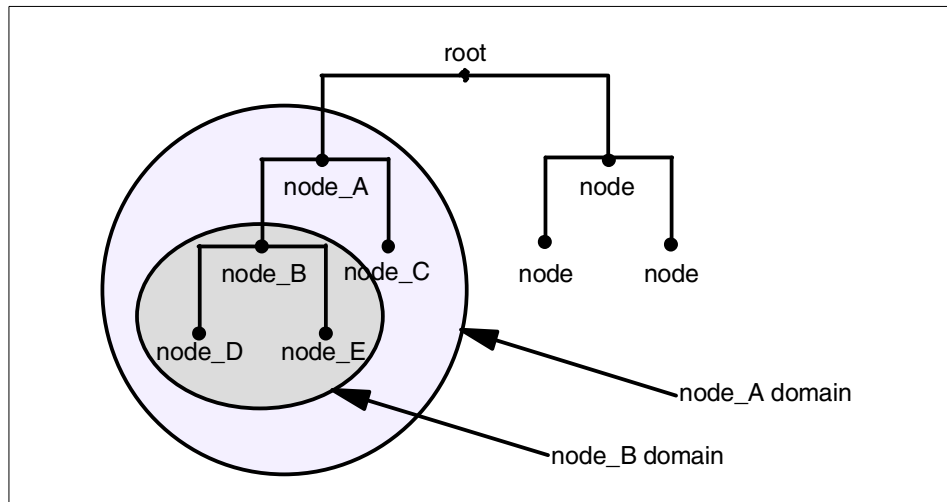


Figure 104. Example DNS subdomain

Domain names

Each domain node, in other words, each network resource, is labeled with a name of up to 63 characters in length. Currently, domain names are not case sensitive. A node may have a label AAA that can be referred to as either AAA or aaa. It is strongly recommended that you preserve the case of any names you use. Some operating systems, namely UNIX, are case-sensitive in many respects. Another reason for preserving case in your domain names is that future developments in DNS may possibly implement case-sensitive services.

The name does not have to be unique in itself. Some names appear many times in the name space. A good example of this are the names *mailserver* and *mail*. These names appear in almost every network connected to the Internet. However, to ensure that each node in the tree can be uniquely identified through its domain name, it is stipulated that sibling nodes (that is, those nodes with the same parent node) must not use the same name. This limitation applies only to the child nodes, and the name may appear in a node with a different parent.

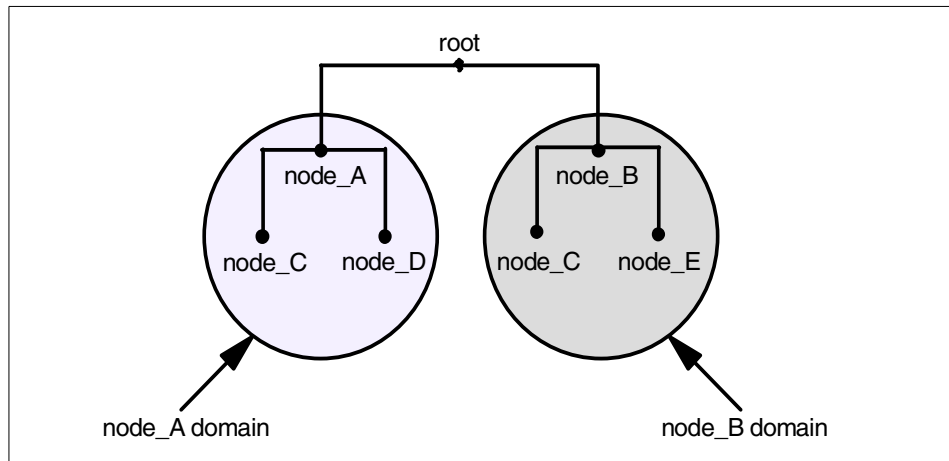


Figure 105. Domain names

Figure 105 illustrates how a name may appear more than once within the tree. The name `node_C` appears twice in the tree, once as part of the domain `node_A`, and again as part of the domain `node_B`. Both `node_A` and `node_B` are siblings (they have the same parent node - root); so, their names must be unique, otherwise, things can get confusing. Both `node_C` and `node_D`, in the `node_A` domain, are also siblings and must again be named uniquely. However, `node_C` in the `node_B` domain has a different parent node from `node_C` in the `node_A` domain, `node_B` and `node_A`, respectively. The unique identity of each node must be maintained. This is achieved through the use of the identity, the name, of its parent node whenever you reference the node outside of its own domain. This scheme fully qualifies the name and provides what is known as a *fully qualified domain name* (FQDN).

Reiterating, a domain name may be of two types:

- **Unqualified Name** – This type of name consists of only the host name given to a particular host. As can be appreciated, throughout the world there may be many hosts with the same unqualified name. It is impractical, if not impossible, to specify unique host names to every machine on the Internet such that no two machines have conflicting DNS entries.

Thus, a host's unqualified domain name alone does not enable it to be identified, except in the local domain. A 32-bit IP address still must be used to address hosts on the network.

- **Fully Qualified Domain Name (FQDN)** – The use of an unqualified name within a domain is the efficient way that names are used in preference to addresses and is perfectly valid. Referring to `USER1` is much easier (from

a human perspective) than using the 32-bit IP address 172.16.3.14, for example. However, the IP address is unique within the Internet, while the name node_C (as we have shown previously) may not be. The answer is the FQDN. To create the FQDN of a node, we must use the sequence of names on the path from the node back to the root with periods separating the names. These names are read from left to right, with the most specific name (the lowest and farthest from the root) being on the left. Thus, we see that the two hosts in our previous example now have completely unique FQDNs:

```
node_C.node_A.root and node_C.node_B.root
```

In practice, the name of the root domain is never shown; it has null length and is usually represented by a period (.). When the root appears in a domain name, the name is said to be absolute. For example:

```
node_C.node_A. (the root is represented by the trailing period)
```

This makes the FQDN totally unambiguous within the name space. However, domain names are usually written relative to a higher level domain rather than to the root itself. In the previous example, this would mean leaving off the trailing period and referring to node_C relative to the node_A domain. For example:

```
node_C.node_A
```

When you configure a TCP/IP host, you are requested to enter the host name of the host and the domain origin to which this host belongs. In the previous example, if we configured a host in the node_C.node_A domain, we would enter the host name as, for example, host-X, and the domain origin as node_C.node_A. Whenever a non-qualified name is entered at this host, the resolver will append the current domain origin to the name, resulting in a FQDN belonging to the same domain as our own host, which enables us to refer to hosts that belong to the same domain as this host by just entering the unqualified host name. If we enter host-Y, the resolver will append the domain origin building the fully qualified name host-Y.node_C.node_A before trying to resolve the name to an IP address. If we want to refer to hosts outside our own domain, we will enter the fully qualified name as, for example, host-Z.node_E.node.A.

Top-Level Domain (TLD)

There is seemingly no restriction on the names that you can create for each node, other than that of length and uniqueness among siblings. However, the NIC decided to provide some sort of order within the name space to ease the burden of administration. Below the root are a number of top-level domains or (TLDs). These TLDs consist of seven generic domains established originally in the U.S. to identify the types of organization represented by the particular

branch of the tree. These have been listed in C.4.2, “DNS name structure” on page 335.

DNS Zones

We have used the word zone on a number of occasions in the last section without explaining its meaning. Divisions in the domain name space can be made between any two adjacent nodes. The group of connected names between those divisions is called a zone. A zone is said to be authoritative for all the names in the connected region. Every zone has at least one node and, consequently, at least one domain name, and all the nodes in a zone are connected. This sounds very much like a domain.

However, there is a subtle difference between a zone and a domain. A zone may contain exactly the same domain names and data as a domain, and this is often the case. If a name server has authority for the whole domain, then the zone will, in fact, be the same as the domain. As networks grow, it is common that, for ease of administration, a domain may be divided into subdomains with the responsibility for these subdomains being delegated to separate parts of an organization, or indeed, to a different organization completely. When this happens, the authority for those subdomains is usually assigned to different name servers. At this point, the zone is no longer the same as the domain. The domain contains all the names and data for all of the subdomains, but the zone will contain only the names and data for which it has been delegated authority.

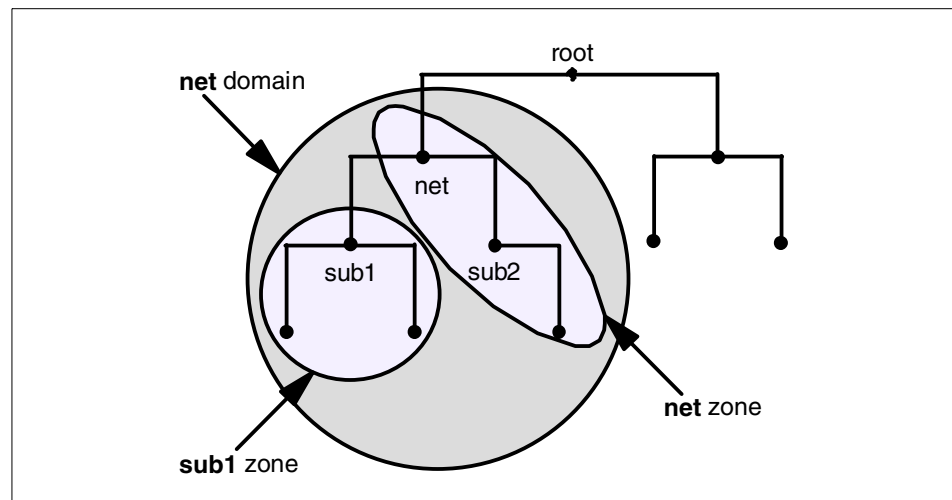


Figure 106. Domains and Zones

Figure 106 illustrates the difference between a zone and a domain. The *net* domain contains names and data for the *net* domain, the *sub1* domain and the *sub2* domain (*sub1* and *sub2* are both subdomains of the *net* domain). However, only domain *sub1* has been delegated the authority for its resources and, hence, has its own zone, the *sub1* zone. The *sub2* domain is still under the authority of the *net* zone.

Name Servers

The second component of the Domain Name System is the *name server*. Name servers are the repositories for all of the information that makes up the domain name space. Originally, there was a single name server, operated by the NIC, which held the single hosts.txt file. The concept of the hierarchical name space has meant that a single name server would be impractical. There are now nine root name servers with responsibility for the top-level domains. The name space is then divided into zones, as we have already discussed, and these zones are distributed among the name servers such that each name server has authority over just a section of the entire name space. This division is frequently based on organizational boundaries, with freedom to subdivide at will. A name server may, and often will, support more than one zone, and a single zone may be served by more than one name server.

Name servers come in the following three types:

- **Primary name server** – This maintains the zone data for the zones it has authority over. Queries for this data will be answered with information from files kept on this name server.
- **Secondary name server** – This has authority over a zone but does not maintain the data on its own disks. The zone data is copied from the primary name server database when the servers are started. This is known as a zone transfer. The secondary then contacts the primary at regular intervals for updates.
- **Caching-only name server** – This server has no authority over any zones and contains only records pointing to other (primary or secondary) name servers. Data is kept in a cache for future use and discarded after a time-to-live (TTL) value expires.

The main function of the name server is to answer standard queries from clients. These queries flow in DNS messages and identify the type of information that the client wants from the database and the host in question. The name server can answer queries in a number of ways depending on the mode of operation of the client and server.

- **Recursive mode (Figure 107)** – When a client makes a recursive query for information about a specified domain name, the name server responds

either with the required information or with an error, such as the domain name does not exist (name error) or there is no information of the requested type. If the name server does not have authority over the domain name in the query, it will send its own queries to other name servers to find the answer. These name servers are pointed to by the additional resource records in the database.

- Non-recursive or iterative mode (Figure 108) – In this case, when a client makes a query, the name server has an extra option. It will return the information if it has it. If not, rather than ask other name servers if they have the data, it will respond to the query with the names and addresses of other name servers for the client to try next.

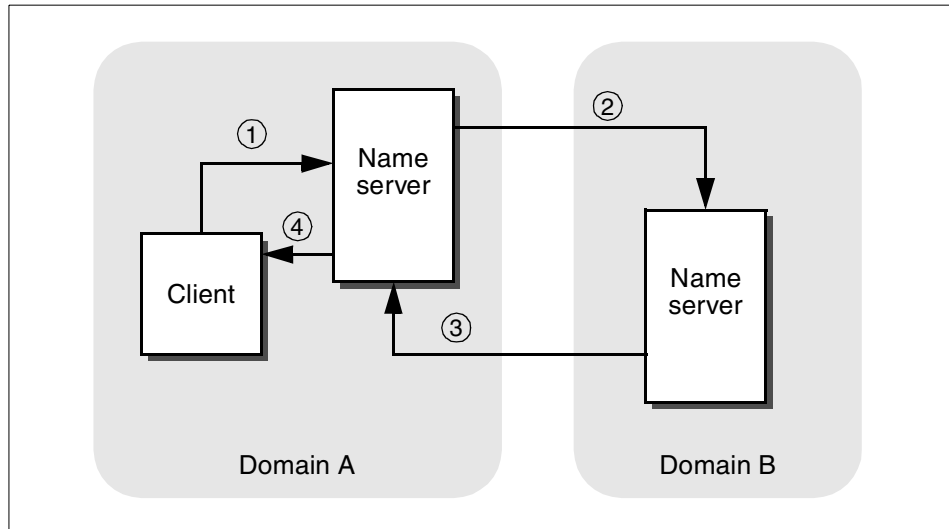


Figure 107. Recursive mode example

Legend:

- 1) The client in domain A sends a simple query to its name server asking for the address of a host in domain B.
- 2) The specified name server does not have authority over domain B and has no record of the host. The name server has an NS resource record pointing to an authoritative name server for domain B and, therefore, it sends a query to that name server asking for the address of the host.
- 3) The name server in domain B returns the address of the host to the name server in domain A.
- 4) The name server in domain A returns the address of the host to the client.

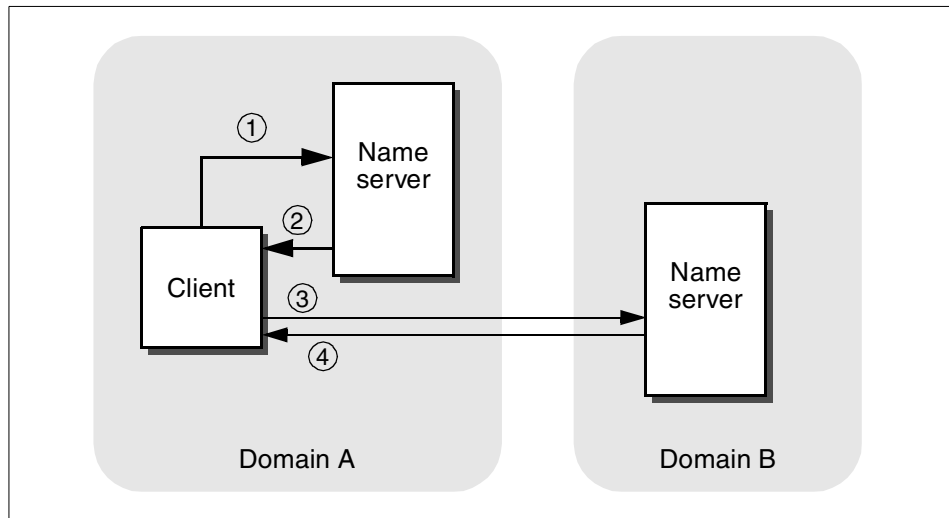


Figure 108. Non-Recursive mode example

Legend:

- 1) The client in domain A sends a simple query to its name server asking for the address of a host in domain B.
- 2) The specified name server does not have authority over domain B and has no record of the host. The name server has an NS resource record pointing to an authoritative name server for domain B. But, rather than send its own query to that name server, it responds negatively to the clients query and gives the client the address of the name server in domain B.
- 3) The client sends a second query, this time to the name server in domain B.
- 4) The name server in domain B returns the address of the host to the client.

Resolvers

The resolvers are the third component of the Domain Name System. These are the clients making queries to the name servers on behalf of programs running on the host. These user programs make system or subroutine calls to the resolver requesting information from the name server. The resolver, which runs on the same host as the user program, will transform the request into a search specification for resource records located (hopefully) somewhere in the domain name space. The request is then sent as a query to a name server that will respond with the desired information to the resolver. This

information is then returned to the user program in a format compatible with the local host's data formats.

C.4.6 DNS Security

A DNS name server poses a risk when exposed to an external network. As a basic security precaution, an organization's internal network names and addresses must not be exposed to external users. There are only a few exceptions to this, such as an organization's external Web server or e-mail server that need to be known to the external world. We must, therefore, adopt a special technique when installing a name server in relation to a firewall. Also, there have been attacks to some versions of DNS servers; so, it is always good practice to find out about any known vulnerabilities of the DNS server being used.

The goal of this scheme is to provide a full Domain Name System to hosts inside the secure network while only providing information about the firewall itself and a few other servers, such as Web servers, to the outside world. This is often provided as a feature of the firewall implementation. The firewall name server will respond to queries from the outside only with information about the firewall address itself. When a host in the secure network makes a query about a host in the non-secure network, the name server will forward the query to the firewall name server. The firewall name server will, in turn, refer the query to a name server in the non-secure network to resolve the name.

A similar process applies to electronic mail passing through the firewall. One way to overcome the problem is to employ a mail forwarding service on the firewall. This acts as a relay for the secure mail server inside the secure network. External hosts will direct their mail to *user@firewall.company.com* or *user@company.com* depending on where the domain begins. Both the secure mail server and the mail forwarder on the firewall must be configured as Relay Hosts (DR entry in *sendmail.cf* file) to allow mail headers to be re-written and mail not destined for the local host to be routed through the firewall.

C.4.7 Name server structures

The domain servers can be placed inside the network. This configuration is presented in Figure 109. As can be seen, the DNS server(s) are on the secure network inside the firewall.

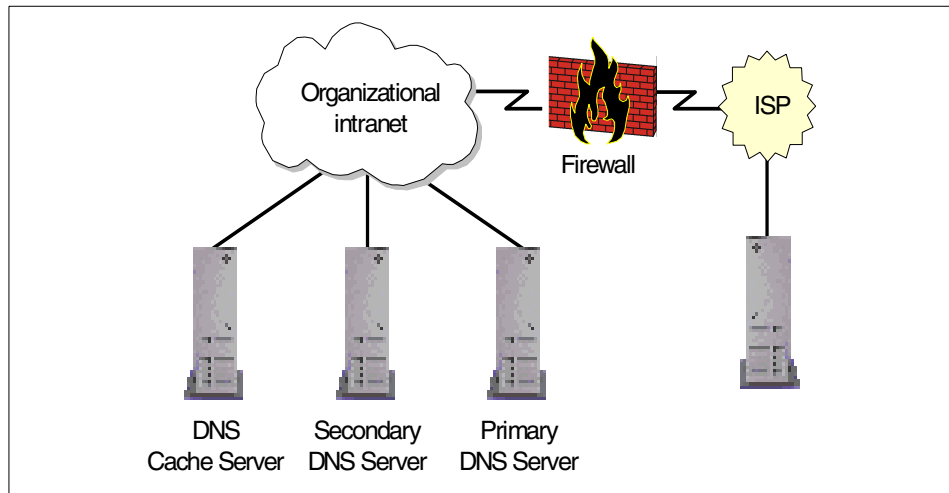


Figure 109. Internal Domain Server allocation

In this scenario (Figure 109), the outside world has access either to all or none of the DNS services. This depends on the configuration of the firewall. Allowing access to the DNS server(s) on the secure network is not a good idea; it leaves a security hole that can be attacked.

If an organization requires some addresses be advertised on the Internet, a better idea is to have two separate DNS services. One serves the organization's internal network, while the other is dedicated to the few addresses that need to be "visible" from the external network. This latter DNS server would be placed inside the demilitarized zone (DMZ), which is presented in Figure 110.

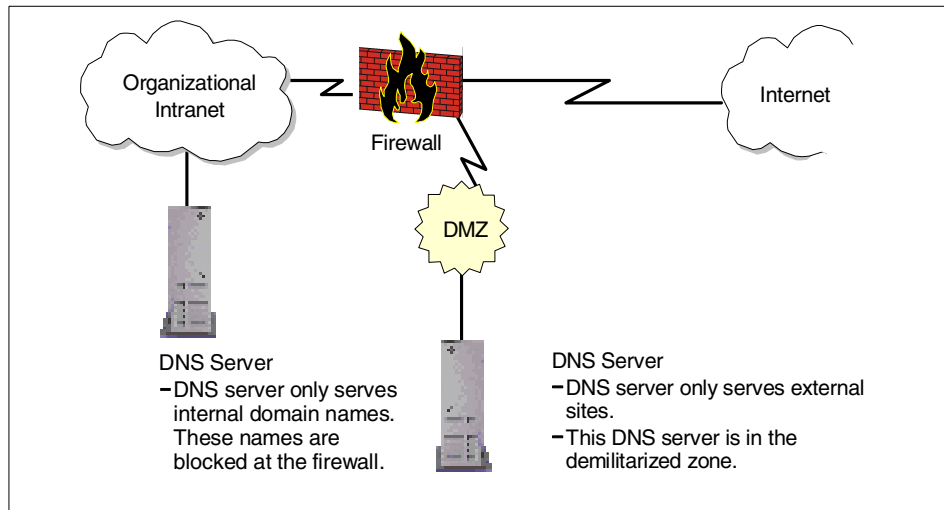


Figure 110. Implementing internal and external domain name spaces

This scheme enables the organization to have DNS services to all the internal hosts while limiting external DNS services to those machines listed on the external DNS server. For large IP networks, this scheme, coupled with a hierarchical domain name space, is recommended for the DNS implementation.

C.4.8 Final word on DNS

Always remember that whenever configuring a DNS systems, the goal of DNS is to enable people to easily identify and remember hosts without using cryptic IP addresses.

This is the theme for the design of DNS. DNS should be implemented in this manner. Computer systems do not require DNS; they are perfectly happy using IP addresses. It is people who require these systems to work efficiently; so, all designs should endeavor to be user-friendly.

As an example, an organization's main Web page could be accessible as:

`http://www.mycompany.com`

The organization's employees, however, could access the very same server as:

`http://w3.mycompany.com`

Then, one could move further on along this path, maybe allowing each division of the company have its internal Web site as follows:

<http://w3.marketing.mycompany.com>
<http://w3.countrycode.mycompany.com>

The entire series of internal w3 sites might not be available from the external Internet at all. Some extranet users can be allowed to access some of these servers and, of course, make them available to the internal network. This will allow for better, faster, safer, and easier communication and coordination for the people inside the organization as the business and network grows. After all, that what is networking is all about.

C.5 Where to get more information

More in-depth information on firewall and name services designs can be found in:

- *IBM SecureWay Firewall for AIX 4.1*, SG24-5855
- *Highly Available IBM eNetwork Firewall Using HACMP or eNetwork Dispatcher*, SG24-5136
- *Building Internet Firewalls*, ISBN 1-5659-2124-0
- *Firewalls and Internet Security: Repelling the Wily Hacker*, ISBN 0-2016-3357-4

Appendix D. Special notices

This publication is intended to help IBM customers, IBM Business Partners and IBM professionals to understand, plan and deploy the IBM SecureWay FirstSecure framework and product offerings. The information in this publication is not intended as the specification of any programming interfaces that are provided by the IBM SecureWay FirstSecure. See the PUBLICATIONS section of the IBM Programming Announcement for the IBM SecureWay FirstSecure for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain 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.

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

DB2 ®	eNetwork
IBM Global Network	IBM Registry
IBM ®	IMS
MQ	MQSeries
NetView	OS/390 ®
OS/400	RACF
RS/6000 ®	S/390 ®
SecureWay ®	SP
System/390	WebSphere
World Registry	400

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Appendix E. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization publications

For information on ordering these ITSO publications see “How to get ITSO Redbooks” on page 357.

- *Understanding LDAP*, SG24-4986
- *LDAP Implementation Cookbook*, SG24-5110
- *Protect and Survive Using IBM Firewall 3.1 for AIX*, SG24-2577
- *Guarding the Gates Using the IBM eNetwork Firewall for NT 3.2*, SG24-5209
- *Highly Available IBM eNetwork Firewall Using HACMP or eNetwork Dispatcher*, SG24-5136
- *IP Network Design Guide*, SG24-2580

E.2 Other publications

We recommend that you check out the IBM SecureWay FirstSecure Web site (see link in the next section) library. It is an excellent online reference for documentation available with IBM SecureWay FirstSecure. Some documents, especially the *Up and Running* guides, are available in multiple languages. This site contains relevant links to other pages for documentation and support.

It is also very important that you check for any latest information, such as *README* files, available on the product CD-ROMs or shipped as hardcopies with the product.

Other sources of related information include:

- *IBM SecureWay Firewall for AIX 4.1*, SG24-5855 (to be available at a later date)
- *Building Internet Firewalls*, ISBN 1-5659-2124-0
- *Firewalls and Internet Security: Repelling the Wily Hacker*, ISBN 0-2016-3357-4
- *Applied Cryptography*, ISBN 0-4711-1709-9

- *Digital Certificates, Applied Internet Security*, ISBN 0-2013-0980-7

E.3 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at <http://www.redbooks.ibm.com/> for information about all the CD-ROMs offered, updates, and formats.:

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

E.4 Information on the Web

Most of the product documentation that is referenced in this redbook can be found at the IBM SecureWay FirstSecure Web site, which includes links to the individual building blocks of FirstSecure and their respective support and documentation libraries:

<http://www.ibm.com/software/security/firstsecure>

The following Web sites are also useful resources:

IBM DCE documentation library:

<http://www.ibm.com/software/network/dce/library>

Tivoli Cross-Site product information and documentation library:

<http://www.cross-site.com>

<http://www.cross-site.com/support/docs>

The Content Technologies Ltd. Web site can be accessed at:

<http://www.contenttechnologies.com>

<http://www.mimesweeper.com>

Information about the Norton AntiVirus products and the Symantec AntiVirus Research Center (SARC) can be found at:

<http://www.av.ibm.com>
<http://www.symantec.com>
<http://www.sarc.com>

Information about the ACE/Server product line can be found at:

<http://www.securitydynamics.com>

The following links provide general information about security, standards, and related subjects:

The IETF: <http://www.ietf.org>
CERT: <http://www.cert.org>
The Open Group: <http://www.opengroup.org>
DMTF: <http://www.dmtf.org>
Insecure.Org: <http://www.insecure.org>

How to get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web site** <http://www.redbooks.ibm.com/>

Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail orders**

Send orders via e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl/

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl/

- **Fax orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information for customer may be found at <http://www.redbooks.ibm.com/> and for IBM employees at <http://w3.itso.ibm.com/>.

IBM Intranet for employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

List of abbreviations

ACL	Access Control List	HTTP	Hypertext Transfer Protocol
API	Application Programming Interface	HTTPS	HTTP over SSL
CA	Certificate Authority	IBM	International Business Machines Corporation
CAS	Credentials Acquisition Service	ICMP	Internet Control Message Protocol
CDE	Common Desktop Environment	ID	Intrusion Detection
CDSA	Common Data Security Architecture	IDE	Integrated Development Environment
CERT	Computer Emergency Response Team	IETF	Internet Engineering Task Force
CGI	Common Gateway Interface	IMAP	Internet Message Access Protocol
CLI	Command Line Interface	IP	Internet Protocol
CR	Certificate Repository	IRC	Internet Relay Chat
CRL	Certificate Revocation List	ISP	Internet Service Provider
CSSM	Common Security Services Manager	ISV	Independent Software Vendor
DCE	Distributed Computing Environment	ITSO	International Technical Support Organization
DES	Data Encryption Standard	ITU-T	International Telecommunications Union, Telecommunication Standardization Sector
DNS	Domain Name Service	JDK	Java Development Kit
DSA	Digital Signature Algorithm	JRAS	Java Reliability, Availability, and Serviceability
DSB	Directory Services Broker	JRE	Java Runtime Environment
EFM	Enterprise Firewall Manager	LAN	Local Area Network
ESMTP	Extended Simple Mail Transfer Protocol	LDAP	Lightweight Directory Access Protocol
FTP	File Transfer Protocol	LPR	Line Printer Protocol
GSO	Global Sign-On	NAT	Network Address Translation
GSSAPI	Generic Security Service API	NAV	Norton AntiVirus
GUI	Graphical User Interface		
HTML	Hypertext Markup Language		

NAVIEG	Norton AntiVirus for Internet e-mail Gateways	SMP	Symmetric Multiprocessor (or Shared Memory Multiprocessor)
NFS	Network File System	SNMP	Simple Network Management Protocol
NIS	Network Information System	SSL	Secure Sockets Layer
NNTP	Network News Transfer Protocol	TB	Toolbox
NTP	Network Time Protocol	TCB	Trusted Computing Base
ODS	On-Demand Server	TCP	Transmission Control Protocol
PD	Policy Director	TEC	Tivoli Enterprise Console
PDF	Portable Document Format	TFTP	Trivial File Transfer Protocol
PEM	Privacy Enhanced Mail	UDP	User Datagram Protocol
PICS	Platform for Internet Content Selection	URL	Universal Resource Locator
PKCS	Public Key Cryptography Standards	UUCP	Unix to Unix Copy
PKI	Public Key Infrastructure	VPN	Virtual Private Network
PKIX	Public Key Infrastructure for X.509 V3 certificates	WAN	Wide Area Network
POP	Post Office Protocol	WTE	WebTraffic Express
RA	Registration Authority	WWW	World Wide Web
RDBMS	Relational Database Management System		
RFC	Request for Comments		
RPC	Remote Procedure Call		
RSA	Rivest, Shamir, and Adleman algorithm		
SARC	Symantec AntiVirus Support Center		
SBS	Secure Boundary Server		
SDK	Software Development Kit		
SHTTP	Secure Hypertext Transfer Protocol		
SMB	Server Message Block Protocol		
S/MIME	Secure Multipurpose Internet Mail Extension		

Index

Symbols

%TEMP% 263
/etc/hosts 249, 336
/etc/inittab 263
/etc/passwd 249
/etc/security/limits 249
/etc/services 121, 249
/usr/lpp/iau/logs 264

Numerics

4758 PCI Cryptographic Coprocessor 175, 180, 188, 261
5-A's 6

A

Access Control List, see ACL
accountability 6
ACE/Agent 141
ACE/Client 141
ACE/Server 43, 130, 141, 249, 355
ACL 42, 54, 57, 66, 74, 139, 224, 276, 325, 359
 attributes 83
 evaluation 95
 ID 84
 inheritance 78, 91, 94
 namespace 111
 types 85
ActiveX 129, 139, 145, 288
add_rauser 267
admin server (Netscape) 255
administration (5-A's) 6
agent (Cross-Site) 154, 253, 257, 296, 302
AIX 52, 130, 141, 154, 156, 173, 192, 193
antirelaying 142
anti-spamming 133
anti-spoofing 133
AntiVirus (Norton) 151, 240, 294
anti-virus software 11, 23, 123, 136, 142, 144
API 48, 173, 176, 359
 Authorization 42, 112, 118, 193, 210, 230
 certificate-based 185
 CSSM 185
 GSSAPI 65
 LDAP 192, 198
 PKIX 168, 192, 203

SSL 197
 Toolbox 192
 WTE 139
applet 178
application gateway firewalls 127
ARPANET 336
assurance (5-A's) 6
attack 5, 11, 23, 41, 142, 153, 155, 307
audit.log 114
auditing 113
authentication 11, 27, 41, 53, 61, 308
 ACE/Server 130, 140, 249
 basic 63
 certificate-based 62, 64, 303
 forms-based 63
 IP address based 128
 Kerberos 60
 mutual 42
 NetSEAL and NetSEAT 71
 PKI 167
 policy-based 20, 42
 SecurID 140, 249
 SSL 167, 303
 user ID/password 62, 63
authorization 6, 27, 41, 53, 61, 66
 evaluator 67
 policy-based 20, 42
Authorization API 42, 112, 118, 193, 210, 230
Authorization Policy Database 56, 66, 96, 113, 115, 282
availability 6, 15, 115, 282, 325

B

backup 50
basic authentication 63
bloodhound heuristic technology 164
BMP 136
boundaries 124, 292
boundary server (see also SBS) 122
boundary services 11
British Standard Code of Practise 14
brute force 312
BS7799 14

C

CA 27, 168, 171, 173, 175, 203, 299, 318, 359

- caching-only name server 342
- CAS 54, 57, 62, 63, 64, 65, 221, 233, 303, 359
- CCITT 183, 321
- CDE 359
- CDMF 133
- CDSA 47, 167, 171, 184, 191, 359
- cell (DCE) 60, 222
- CERT 8, 49, 153, 355, 359
- certificate 26, 28, 180, 255
 - chaining 30
 - root (self-signed) 33, 257
 - server 177
 - types (Trust Authority) 179
- Certificate Authority, see CA
- Certificate Library (CL) 185
- Certificate Repository, see CR
- Certificate Request Syntax Standard 184
- Certificate Revocation List, see CRL
- certificate-based authentication 62, 64, 303
- CfgPostInstall 263
- CfgStart 264
- CGI 54, 57, 68, 359
- CGI script 96, 109
- checksum 120
- child node 338
- CICS 168
- cipher 307
- ciphertext 309
- circuit gateway firewalls 126
- CL 185
- cleartext 307
- CLI 359
- client (DCE) 231
- coarse-grained access control 43
- code analysis 129
- collection 153
- Common Data Security Architecture (CDSA) 47, 184
- Common Security Services Manager (CSSM) 185
- compartmentalized firewall 330
- computer virus xvii, 17, 24, 129, 162, 294
- confidentiality 15, 308
- console.properties 283
- content filtering 21
- content inspection 139
- Content Technologies Ltd. 44, 135, 137, 354
- coprocessor (cryptographic) 175, 180, 188, 261
- CORBA 52, 54
- CR 27, 34, 359

- Credentials Acquisition Server, see CAS
- CRL 33, 169, 171, 176, 179, 180, 304, 320, 359
- cross-certification 304
- Cross-Site
 - Agent 45, 154, 253, 257, 296, 302
 - Management Console 45, 156, 160, 253, 256, 258
 - Management Server 155, 253
- cryptanalysis 307, 309
- CryptoAPI 172
- cryptographic coprocessor 175, 180, 188, 261
- Cryptographic Message Syntax Standard 184
- Cryptographic Service Provider (CSP) 185
- cryptography 36, 307
- Cryptoki 184
- cryptology 308
- cryptosystem 307
- CSP 172, 185
- CSSM 171, 185, 359
- custom-built applications 11

D

- daemons 55
- damage 275
- DAP 321
- data backup 50
- Data Service Library (DL) 185
- DB2 155, 168, 180, 188, 224, 253, 261, 323
- DB2CODEPAGE 262
- DCE 222, 359
 - cell 60, 222
 - client 231
 - documentation library 354
 - DTS 220
 - registry 56
 - RPC 60
 - Security Server 58
 - user registry 56, 65, 235
- dce_login 106
- dcecp 114
- dced.log 114
- decryption 307
- Demilitarized Zone, see DMZ
- DES 133, 312, 359
- destination port 121
- DIB 183
- digital envelope 312
- digital signature 27, 35, 140, 169, 182, 186, 308,

316
Digital Signature Algorithm (DSA) 309
Directory Information Base (DIB) 183
Directory Management Tool (DMT) 224
Directory Services Broker (DSB) 221, 232
dispatcher 334
Distinguished Encoding Rules (DER) 319
Distinguished Name 80
Distributed Computing Environment, see DCE
Distributed Time Service (DTS) 220
DIT 323
DL 185
DMTF 326, 355
DMZ 22, 50, 233, 235, 239, 329, 330, 346
 multiple 275
DNS 49, 152, 155, 160, 239, 251, 336, 347, 359
 design 347
 message 342
 name space 337
 recursive mode 342
 resolver 336, 344
 security 345
 server 336, 342
 tree 337
 zones 341
domain 154
domain name 337, 338, 340, 342, 345
domain name space 336, 341, 344
Domain Name System, see DNS
domain node 338
domain origin 340
Dr. Solomon's Anti-Virus Toolkit 136
DSA 309, 359
DTS 220

E

eavesdropping 21, 25, 26, 140, 153, 159, 310
e-business 3, 41
EFM 135, 359
encryption 48, 84, 133, 184, 185, 307, 308
 hardware encryption 333
end entity 27, 34, 172, 204, 260
Enterprise Server (Netscape) 253
Entrust 255, 304
envelope (digital) 312
ePerson 80
ERP 293
ESMTP 359

Ethernet 154, 158, 159, 160
evaluating an ACL 95
event 154
Exchange (Microsoft) 46
external DNS server 347
extranet 122

F

fine-grained access control 42, 78
finger 155
Finjan Software Ltd. 44, 139
FIPS 36
firewall 21, 121, 122, 130, 220, 345, 346
 circuit gateway 126
 FTP proxy 132
 HTTP proxy 131, 148, 247, 332
 IP filter 125
 mail proxy 133
 name server 345
 proxy users 43, 235, 283
 stateful inspection 125
 Telnet proxy 132
flood attacks 155
forms-based authentication 63
F-Prot 137
fragmentation attacks 155
fsize 249
FTP 43, 71, 131, 147, 155, 186, 244, 248, 359
 proxy 132
fully qualified domain name (FQDN) 339

G

generic proxies 126
GIF 136
Global Security Kit (GSKit) 227, 261
gopher 155
government issues 4, 270
GSO 69, 359
GSSAPI 65, 71, 359
GUI 359
GZIP 136

H

H+B EDV 137
HACMP 282, 334
hardening the operating system 134
hardware encryption 333

- hashing 313
- heartbeat 154
- hierarchical domain name space 347
- host table 336
- host-based intrusion detection 23
- HTML 359
- HTTP 20, 43, 131, 155, 186, 244, 248, 252, 296, 359
 - cache proxy 289
 - proxy 131, 148, 247, 332
- HTTPS 248, 296, 359

I

- IBM Firewall (see also firewall) 130, 287
- IBM HTTP Server 54, 261
- IBM Policy Director (see also Policy Director) 53
- IBM World Registry 255
- ICMP 359
- ICMP redirect 155
- ID 359
- IDE 359
- ident 155
- ids.cfg 160
- ids.msg 160, 162
- ids.rules 160
- IETF 36, 47, 169, 187, 323, 355, 359
- IMAP 155, 359
- IMS 168
- incident 155
- Inprise 52
- InstallShield 256, 257, 263, 269
- integrity 15, 308
- internal network 121
- Internet 3, 11, 20, 44, 62, 119, 121, 218, 238, 271
- Internet Engineering Task Force, see IETF
- Internet Explorer (Microsoft) 135
- Internet Service Provider (ISP) 42, 148, 240, 289
- intranet 121
- intrusion detection 11, 23, 44
- Intrusion Immunity 23, 41, 151
- IONA 52
- IP 359
 - address 347
 - filtering 332
 - network 121, 347
 - protocol 119
- IP filter firewalls 125
- IPSec 133, 187, 304

- iptrace 242
- IRC 155, 359
- ISO 14, 321
- ISP 42, 148, 240, 289, 359
- ISV 359
- iterative mode 343
- ITSO 359
- ITU-T 183, 321, 359
- IV.Base 230, 231
- iv.conf 114, 230, 237
- IV.Net, IV.Web, IV.Trap 232
- ivacl.d.log 114
- ivadmin 111, 113, 116
- IVAuthADK 230
- ivmgrd 237
- ivmgrd.conf 100, 230, 237
- ivmgrd.log 114

J

- Java 139, 261, 288
- Java Development Kit, see JDK
- Java Runtime Environment (JRE) 225
- JavaScript 129, 139, 261, 288
- JDK 227, 228, 253, 256, 261, 270, 359
- JNH 171
- Jonah 47, 169, 172, 173
- JPEG 136
- JRAS 359
- JRE 225, 359
- junctioncp 106
- junctioned 43

K

- Kerberos 60, 62, 65, 155
- key 309
- key recovery service 35
- keyring file 257
- KeyStore 177
- KeyWorks 48, 173, 191, 192, 194, 261, 270

L

- label 335
- LAN 359
- law 4, 270
- LDAP 180, 187, 188, 222, 223, 304, 321, 359
 - API 192, 198
 - user registry 43, 54, 56, 64, 149, 223, 230,

234, 235
ldapmodify 284
legal practices 6
license key 254
lifecycle 188
LiveUpdate 163
load balancing 282
log files 114
logon attempts 219
Lotus Notes 46, 294
LPR 155, 359

M

mail proxy 133
mail relaying 142
MAILsweeper 44, 130, 135, 249, 287, 289, 294, 332, 333
manageability 325
Management Console (Cross-Site) 45, 156, 253, 256, 258
Management Console (Microsoft) 250
Management Console (Policy Director) 57, 221, 225, 229, 232, 236
Management Server (Cross-Site) 45, 155, 160, 253
Management Server (Policy Director) 56
MAPI32.dll 250
masquerading 315
McAfee VirusScan 137
MD5 133
message authentication code (MAC) 175
Microsoft Exchange 46
MIME 136
MIMESweeper 44, 130, 249, 287
MQSeries 52, 168
mutual authentication 42

N

name server (DNS) 336, 342
name space (DNS) 337
namespace 67, 96, 325
 management 110
NAT 125, 241, 359
NAV 359
NAVIEG 360
NETBIOS 241
Netscape 321
Netscape Enterprise Server 155, 253

NetSEAL 57, 71, 211, 221
NetSEAT 71
NetWare (Novell) 46
network address translation (NAT) 125
network boundaries 124, 292
Network Dispatcher 117, 282
Network File System, see NFS
Network Manager for Windows 164
network name resolution 49
Network Security Auditor (NSA) 133, 218, 252
Network Time Protocol (NTP) 220
NFS 155, 160, 360
NIS 155, 360
NNTP 155, 360
non-repudiation 308
normal mode FTP 147
Norton AntiVirus 44, 45, 136, 333, 355
Norton AntiVirus Suite 151, 162, 252, 294
Norton System Center 164
Notes (Lotus) 46
Novell NetWare 46
NSA 133, 218, 252
nsid.log 114
ns-install 255
NTP 220, 360

O

object namespace 67
Object Request Brokers (ORBs) 52
object space 67
ODS 360
operating system 218, 331
operating system hardening 49, 134
operational practices 6
Oracle 155, 253
OS/390 62, 322
OS/400 322

P

parent node 339
passive mode FTP 147
password 21, 62, 64, 219
pattern search 129
PD (see also Policy Director) 360
PDF 360
performance 251
Personal Computer 312
Personal Information Exchange Syntax Standard

- 184
- personnel security 6
- PGP 311
- physical design 50
- physical security 6, 220
- PICS 137, 360
- ping of death attack 155
- PKCS 168, 183, 360
 - #10 171, 184
 - #11 179, 184
 - #12 184
 - #7 184
- PKI 26, 46, 52, 167, 320, 360
- PKIX 168, 182, 360
 - API 192, 203
 - Toolkit 270
- pkmslogout 64
- plaintext 307
- Platform for Internet Content Selection (PICS) 137, 360
- policy 11, 154
- Policy Director 20, 41, 53
 - ADK 230
 - architecture 55
 - auditing 113
 - Authorization API 210
 - configuration files 236
 - DCE Security Server 58
 - log files 114
 - Management Console 57, 221, 225, 229, 232, 236
 - Management Server 56
 - scalability and availability 115
 - Security Manager 56
 - User Registry (see also User Registry) 56
- policy enforcer 67
- policy template 67
- policy-based authentication 20, 42
- policy-based authorization 20, 42
- POP 23, 57, 155, 360
- port 120, 121
- port scan 155
- portmapper 155
- Post Office Protocol, see POP
- Pretty Good Privacy (PGP) 311
- primary name server (DNS) 342
- Privacy Enhanced Mail (PEM) 169
- promiscuous mode 294
- protocol 120

- proxy 132
- proxy domain 286
- proxy users 43, 235, 283
- Public Key Cryptography Standards, see PKCS
- public key cryptosystems 307
- Public Key Infrastructure (PKI) 11, 167, 320
- public-key encryption 310
- puschema.def 284

Q

- query_contents 109

R

- RA 27, 169, 171, 173, 174, 203, 299, 360
- RACF 62
- RADInst.exe 267
- radio clock 220
- RC4 312
- RDBMS 253, 360
- recursive mode (DNS) 342
- Registration Authority, see RA
- registry (DCE) 56
- Relying Entity (PKI) 27, 34
- remote mode (Authorization API) 212
- replica mode (Authorization API) 212
- request ID 178, 266
- resolver (DNS) 336, 344
- resource 153
- resource records (RRs) 336
- revocation 188
- rexec 155
- RFC 326, 360
 - 1034 336
 - 1035 336
 - 1422 169
 - 1597 133
 - 2459 33
 - 606 336
 - 810 336
 - 952 336
- risk assessment 5, 7
- Rivest Shamir and Adleman (RSA) 311
- rlogin 155
- root CA 30
- root certificate 264
- RPC 360
- RPC_RESTRICTED_PORTS 278, 279, 280
- rpsccp 280

RSA 311, 360
rsh 155
rstatd 155

S

S/MIME 172, 186, 360
SARC 355, 360
SBS 41, 119, 238, 287, 360
scalability 56, 62, 115, 149, 177, 187, 231, 281, 282, 299, 325
scalable infrastructure 188
scalable system 188
scenarios 271
schema 323
SDK 11, 35, 48, 191, 360
sec_key 255
secd.log 114
secmgrd 237
secmgrd.conf 230, 237
secmgrd.log 114
secondary name server (DNS) 342
secret key cryptosystems 307
secret-key encryption 310
secure FTP proxy 132
secure HTTP proxy 131
secure server network 330
Secure Sockets Layer, see SSL
secure Telnet 251
secure Telnet proxy 132
SecureWay Boundary Server, see SBS
SecureWay Directory 56, 177, 180, 188, 198, 224, 236, 244, 261, 322
SecureWay Directory Client SDK 203
SecureWay Network Dispatcher 117, 282, 334
SecurID 43, 52, 140, 141, 249
security 325
 DNS 345
 holes 329
 policies 4, 11, 74
 token 179
 zones 330
security domain 177
Security Dynamics 43
Security Manager (PD) 56
Security Server (Cross-Site) 252
SecurityMaster 285
sendmail 133
server certificate 255

services 55, 121
servlet 57, 155, 255, 296
SET 187
SHA 133
SHTTP 360
sibling node 338
signature 129
Simple Public Key Infrastructure (SPKI) 184
Single Sign-On 52
Smart Card 179, 184, 269
smart junctions 102
SMB 155, 160, 360
SMIT 230
SMP 130, 360
SMTP 20, 43, 131, 155, 252
SMTP gateway 332
sniffing 330
SNMP 156, 360
Socks 126, 244
software development kit, see SDK
Solaris (Sun) 154, 156, 192, 193, 252, 322
Sophos Anti-Virus 137
source port 120
SPKI 184
SSH 251
SSL 71, 167, 186, 197, 227, 255, 281, 303, 310, 312, 360
SSLight 197
standards 7, 17, 26, 41, 133, 182, 192, 325
stateful inspection 125
statements of directions 51
strike technology 164
subdomain 337, 341
subscribers 27
SurfinConsole 140, 247
SurfinGate 44, 130, 139, 235, 246, 288, 289, 294, 333
swingall.jar 263
Symantec Corp. 163, 252, 295, 355
synchronizing time 220
system backup 50
system defaults 219

T

TACInst.exe 269
talk 155
tamper detection 309
tampering 25, 26

- tar 136
- task 154
- TB 360
- TCB 219, 360
- TCP 119, 360
- TCP/IP 322
- tcpdump 242
- TEC 156, 259, 360
- Telnet 20, 43, 57, 71, 131, 155, 186, 244
- Telnet proxy 132
- TFTP 155, 360
- The Internet Security Conference (TISC) 9
- The Open Group 36, 47, 59, 172, 184, 355
- threats 4
- Thunderbyte Anti-Virus 137
- TIF 136
- time synchronization 220
- time-to-live 342
- TISC 9
- Tivoli 52
- Tivoli Cross-Site for Security 45, 151, 153, 252, 294, 354
- Tivoli Enterprise Console, see TEC
- TLS 186
- token 179
- Token-Ring 154, 158, 159, 160
- Toolbox 41, 48, 270, 272
- top-level domain 340
- TP 185
- tree (DNS) 337
- triple DES 133
- trojan horse 129
- Trust Authority 41, 255, 259
 - additional customization 269
 - CA 175, 299
 - certificate types 179
 - client 269
 - crypto hardware 180, 261
 - installation and configuration 262
 - principles of operations 177
 - RA 174, 299
 - RA desktop 175, 267, 299
 - RA server 301
 - system configurations 177
- Trust Policy (TP) 185

U

- UDP 119, 360

- UMICH 321
- UNIX 336, 338
- URL 360
- user IDs 219
- User Registry 43, 54, 56, 65, 84, 149, 221, 273
- UTF-8 262
- UUCP 155, 360

V

- Verisign 255, 304
- VET Anti-Virus 137
- virus xvii, 17, 24, 129, 162
 - anti-virus proxy 333
 - protection 45, 252, 294
 - scanning programs 25
- VPN 43, 133, 167, 172, 187, 304, 360

W

- W3C 9
- WAN 360
- Web server 19, 42, 54, 57, 67, 102, 109, 138, 159, 224, 251, 252, 255, 327, 330, 345
- Web Traffic Express (WTE) 131, 139, 145, 246
- WebSEAL 57, 67, 211, 221
- WebSphere 54
- WebSphere Application Server 253, 261
- WEBSweeper 44, 130, 249, 287, 289, 294, 333
- well-known CA 257
- well-known port 121
- whois 152
- Windows NT 130, 141, 154, 156, 173, 192, 193, 218, 225, 228, 240, 246, 252, 322
- writesrv 155
- WTE (see also Web Traffic Express) 360
- WWW 360

X

- X.500 183, 321
- X.509 63, 168, 169, 180, 182
- X.509 v2 CRL 33
- X.509v3 183, 318
- xIC.rte 261
- XSite 153
- X-Windows 155

Z

- zip 136, 163

zone transfer 342
zones (DNS) 341

ITSO redbook evaluation

Understanding IBM SecureWay FirstSecure
SG24-5498-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?

Customer **Business Partner** **Solution Developer** **IBM employee**
 None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

SG24-5498-00
Printed in the U.S.A.

Understanding IBM SecureWay FirstSecure

SG24-5498-00

