



System Administration Guide: Security Services

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 806-4078-06
December 2001

Copyright 2001 Sun Microsystems, Inc. 901 San Antonio Road Palo Alto, CA 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, SUNOS, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2001 Sun Microsystems, Inc. 901 San Antonio Road Palo Alto, CA 94303-4900 U.S.A. Tous droits réservés

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, SunOS, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



011025@2471



Contents

Preface	17
1 Security Services Overview	23
Introduction to Security	23
Authentication	24
Access Control	24
Secure Communication	25
Auditing	25
2 Authentication Services Topics	27
3 Using Authentication Services (Tasks)	29
Overview of Secure RPC	29
NFS Services and Secure RPC	30
DES Encryption	30
Kerberos Authentication	30
Diffie-Hellman Authentication	31
Administering Diffie-Hellman Authentication	34
▼ How to Restart the Keyserver	34
▼ How to Set Up a Key in NIS+ Credentials for Diffie-Hellman Authentication	35
How to Set Up a New User Key Using NIS+ Credentials for Diffie-Hellman Authentication	36
▼ How to Set Up a root Key Using NIS Credentials With Diffie-Hellman Authentication	36

	How to Create a New User Key Using NIS Credentials with Diffie-Hellman Authentication	37
	▼ How to Share and Mount Files With Diffie-Hellman Authentication	38
	PAM (Overview)	38
	Benefits of Using PAM	39
	PAM Components	39
	Stacking Feature	40
	Password-Mapping Feature	40
	PAM (Tasks)	41
	PAM (Task Map)	41
	Planning for PAM	41
	▼ How to Add a PAM Module	42
	▼ How to Prevent Unauthorized Access From Remote Systems With PAM	43
	▼ How to Initiate PAM Error Reporting	43
	PAM (Reference)	44
	PAM Modules	44
	PAM Module Types	45
	PAM Configuration File	46
4	Using Secure Shell (Tasks)	51
	Introduction to Secure Shell	51
	Using Secure Shell (Task Map)	54
	Using Secure Shell	55
	▼ How to Create a Public/Private Key Pair	55
	▼ How to Log Into Another Host Using Secure Shell	56
	▼ How to Log in with No Password Using ssh-agent	56
	▼ How to Set ssh-agent to Run Automatically	58
	▼ How to Use Secure Shell Port Forwarding	58
	▼ How to Copy Files with Secure Shell	60
	▼ How to Transfer Files Remotely Using sftp	60
	▼ How to Set Up Default Connections to Hosts Outside a Firewall	61
	How to Connect to Hosts Outside a Firewall from the Command Line	62
5	Secure Shell Administration	63
	A Typical Secure Shell Session	63
	Session Characteristics	63
	Authentication	64

	Command Execution and Data Forwarding	64
	Configuring the Secure Shell	65
	Secure Shell Client Configuration	65
	Secure Shell Server Configuration	67
	Maintaining Known Hosts on a Site-Wide Basis	68
	Secure Shell Files	69
6	Introduction to SEAM	73
	What Is SEAM?	73
	How SEAM Works	74
	Initial Authentication: the Ticket-Granting Ticket	75
	Subsequent Authentications	77
	Principals	78
	Realms	78
	Security Services	80
	SEAM Releases	80
	SEAM 1.0 Components	81
	SEAM Components in the Solaris 9 Release	82
	SEAM Components in the Solaris 8 Release	82
	SEAM 1.0.1 Components	83
7	Planning for SEAM	85
	SEAM Configuration Decisions	85
	Realms	85
	Mapping Hostnames Onto Realms	87
	Client and Service Principal Names	87
	Ports for the KDC and Admin Services	87
	Slave KDCs	88
	Database Propagation	88
	Clock Synchronization	88
8	Configuring SEAM	91
	SEAM Configuration Task Map	91
	Configuring KDC Servers	92
	▼ How to Configure a Master KDC	93
	▼ How to Configure a Slave KDC	97

Configuring Cross-Realm Authentication	100
▼ How to Establish Hierarchical Cross-Realm Authentication	100
▼ How to Establish Direct Cross-Realm Authentication	101
Configuring SEAM NFS Servers	102
▼ How to Configure SEAM NFS Servers	103
▼ How to Create a Credential Table	104
▼ How to Add a Single Entry to the Credential Table	105
▼ How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes	105
Configuring SEAM Clients	107
▼ How to Configure a SEAM Client	107
Setting Up Root Authentication to Mount NFS File Systems	110
Synchronizing Clocks between KDCs and SEAM Clients	110
Swapping Master and Slave KDCs	112
▼ How to Configure a Swappable Slave KDC	112
▼ How to Swap a Master and Slave KDC	114
Administering the Kerberos Database	116
Backing Up and Propagating the Kerberos Database	116
▼ How to Back Up the Kerberos Database	118
▼ How to Restore the Kerberos Database	119
▼ How to Manually Propagate the Kerberos Database to the Slave KDCs	120
Setting Up Parallel Propagation	120
▼ How to Set Up Parallel Propagation	121
Administering the Stash File	122
▼ How to Remove a Stash File	122
Increasing Security	123
▼ How to Restrict Access for KDC servers	123
9 SEAM Error Messages and Troubleshooting	125
SEAM Error Messages	125
SEAM Administration Tool Error Messages	125
Common SEAM Error Messages (A-M)	126
Common SEAM Error Messages (N-Z)	132
SEAM Troubleshooting	135
Problems With the Format of the <code>krb5.conf</code> File	135
Problems Propagating the Kerberos Database	135
Problems Mounting a Kerberized NFS File System	136

Problems Authenticating as Root 137

10	Administering Principals and Policies	139
	Ways to Administer Principals and Policies	140
	SEAM Administration Tool	140
	Command-Line Equivalent of the SEAM Tool	141
	Files Modified by the SEAM Tool	142
	Print and Online Help Features of the SEAM Tool	142
	Working With Large Lists in the SEAM Tool	142
	▼ How to Start the SEAM Tool	143
	Administering Principals	145
	Administering Principals Task Map	146
	Automating the Creation of New Principals	147
	▼ How to View the List of Principals	147
	▼ How to View a Principal's Attributes	149
	▼ How to Create a New Principal	151
	▼ How to Duplicate a Principal	153
	▼ How to Modify a Principal	153
	▼ How to Delete a Principal	155
	▼ How to Set Up Defaults for Creating New Principals	155
	▼ How to Modify the Kerberos Administration Privileges	156
	Administering Policies	158
	Administering Policies Task Map	158
	▼ How to View the List of Policies	159
	▼ How to View a Policy's Attributes	161
	▼ How to Create a New Policy	163
	▼ How to Duplicate a Policy	165
	▼ How to Modify a Policy	165
	▼ How to Delete a Policy	166
	SEAM Tool Reference	167
	SEAM Tool Panel Descriptions	167
	Using the SEAM Tool With Limited Kerberos Administration Privileges	170
	Administering Keytabs	172
	Administering Keytabs Task Map	173
	▼ How to Add a Service Principal to a Keytab	173
	▼ How to Remove a Service Principal From a Keytab	174
	▼ How to Display the Keylist (Principals) in a Keytab	175

▼ How to Temporarily Disable Authentication for a Service on a Host 176

11 Using SEAM 179

Ticket Management 179

Do You Need to Worry About Tickets? 180

▼ How to Create a Ticket 180

▼ How to View Tickets 181

▼ How to Destroy Tickets 182

Password Management 183

Advice on Choosing a Password 183

Changing Your Password 184

12 SEAM Reference 187

SEAM Files 187

PAM Configuration File 188

SEAM Commands 189

SEAM Daemons 190

SEAM Terminology 190

Kerberos-Specific Terminology 190

Authentication-Specific Terminology 191

Types of Tickets 192

How the Authentication System Works 196

Gaining Access to a Service Using SEAM 196

Obtaining a Credential for the Ticket-Granting Service 196

Obtaining a Credential for a Server 197

Obtaining Access to a Specific Service 198

Using the `gsscred` Table 199

13 Managing System Security Topics 201

14 Managing System Security (Overview) 203

Where to Find System Security Tasks 203

Controlling Access to a Computer System 204

Maintaining Physical Site Security 204

Maintaining Login and Access Control 204

Restricting Access to Data in Files 205

	Maintaining Network Control	205
	Monitoring System Usage	205
	Setting the Correct Path	206
	Securing Files	206
	Installing a Firewall	206
	Reporting Security Problems	207
	File Security	207
	File Administration Commands	207
	File Encryption	208
	Access Control Lists (ACLs)	208
	System Security	208
	Login Access Restrictions	209
	Special Logins	209
	Managing Password Information	210
	Using the Restricted Shell	211
	Tracking Superuser (Root) Login	212
	Network Security	212
	Firewall Systems	213
	Authentication and Authorization	214
	Sharing Files	215
	Restricting Superuser (Root) Access	216
	Using Privileged Ports	216
	Using Automated Security Enhancement Tool (ASET)	216
15	Securing Files (Tasks)	217
	File Security Features	217
	User Classes	217
	File Permissions	218
	Directory Permissions	218
	Special File Permissions (setuid, setgid and Sticky Bit)	219
	Default umask	220
	Displaying File Information	221
	▼ How to Display File Information	221
	Changing File Ownership	223
	▼ How to Change the Owner of a File	223
	▼ How to Change Group Ownership of a File	224
	Changing File Permissions	225

▼ How to Change Permissions in Absolute Mode	227
▼ How to Change Special Permissions in Absolute Mode	228
▼ How to Change Permissions in Symbolic Mode	229
Searching for Special Permissions	230
▼ How to Find Files With <code>setuid</code> Permissions	230
Executable Stacks and Security	231
▼ How to Disable Programs From Using Executable Stacks	232
▼ How to Disable Executable Stack Message Logging	232
Using Access Control Lists (ACLs)	232
ACL Entries for Files	233
ACL Entries for Directories	234
▼ How to Set an ACL on a File	235
▼ How to Copy an ACL	237
▼ How to Check If a File Has an ACL	237
▼ How to Modify ACL Entries on a File	238
▼ How to Delete ACL Entries From a File	238
▼ How to Display ACL Entries for a File	239
16 Securing Systems (Tasks)	241
Displaying Security Information	241
▼ How to Display a User's Login Status	241
▼ How to Display Users Without Passwords	243
Temporarily Disabling User Logins	243
▼ How to Temporarily Disable User Logins	244
Saving Failed Login Attempts	244
▼ How to Save Failed Login Attempts	244
Password Protection Using Dial-up Passwords	245
▼ How to Create a Dial-up Password	247
▼ How to Temporarily Disable Dial-up Logins	248
Restricting Superuser (root) Access on the Console	249
▼ How to Restrict Superuser (root) Login to the Console	249
Monitoring Who Is Using the <code>su</code> Command	249
▼ How to Monitor Who Is Using the <code>su</code> Command	250
▼ How to Display Superuser (root) Access Attempts to the Console	250
Modifying a System's Abort Sequence	251
▼ How to Disable or Enable a System's Abort Sequence	251

17	Role-Based Access Control (Overview)	253
	RBAC: Replacing the Superuser Model	253
	Solaris RBAC Elements	254
	Privileged Applications	256
	Applications Checking UIDs and GIDs	257
	Applications Checking Authorizations	257
	Profile Shell	257
	Roles	257
	Authorizations	258
	Rights Profiles	258
	Management Scope	259
18	Role-Based Access Control (Tasks)	261
	Configuring RBAC (Task Map)	262
	Planning for RBAC	262
	▼ How to Plan Your RBAC Implementation	262
	First-Time Use of the User Tool Collection	264
	▼ How to Run the User Tool Collection	264
	Initial User Setup	266
	▼ How to Create Initial Users Using the User Accounts Tool	266
	Initial Role Setup	268
	▼ How to Create the First Role (Primary Administrator) Using the Administrative Roles Tool	268
	Making Root a Role	270
	▼ How to Make Root a Role	270
	Managing RBAC Information (Task Map)	271
	Using Privileged Applications	272
	▼ How to Assume a Role at the Command Line	272
	▼ How to Assume a Role in the Console Tools	273
	Creating Roles	274
	▼ How to Create a Role Using the Administrative Roles Tool	274
	▼ How to Create a Role From the Command Line	275
	Changing Role Properties	277
	▼ How to Change a Role Using the Administrative Roles Tool	277
	▼ How to Change a Role From the Command Line	278
	Creating or Changing a Rights Profile	279
	▼ How to Create or Change a Rights Profile Using the Rights Tool	279

	▼ How to Change Rights Profiles From the Command Line	283
	Modifying a User's RBAC Properties	283
	▼ How to Modify a User's RBAC Properties Using the User Accounts Tool	284
	▼ How to Modify a User's RBAC Properties From the Command Line	284
	Securing Legacy Applications	285
	▼ How to Add Security Attributes to a Legacy Application	285
	▼ How to Add Security Attributes to Commands in a Script	285
	▼ How to Check for Authorizations in a Script or Program	285
19	Role-Based Access Control (Reference)	287
	RBAC Elements: Reference Information	287
	Configuring Suggested Roles	287
	Contents of Rights Profiles	288
	Authorizations	292
	Databases Supporting RBAC	293
	Overview of Databases Supporting RBAC	293
	user_attr Database	296
	auth_attr Database	297
	prof_attr Database	299
	exec_attr Database	300
	policy.conf File	301
	RBAC Commands	301
	Command-Line Applications for Managing RBAC	301
	Commands Requiring Authorizations	303
20	Using Automated Security Enhancement Tool (Tasks)	305
	Automated Security Enhancement Tool (ASET)	305
	ASET Security Levels	306
	ASET Tasks	307
	ASET Execution Log	309
	ASET Reports	310
	ASET Master Files	313
	ASET Environment File (asetenv)	314
	Configuring ASET	314
	Restoring System Files Modified by ASET	317
	Network Operation Using the NFS System	318

	ASET Environment Variables	319
	ASET File Examples	322
	Running ASET	323
	▼ How to Run ASET Interactively	324
	▼ How to Run ASET Periodically	325
	▼ How to Stop Running ASET Periodically	325
	▼ How to Collect ASET Reports on a Server	326
	Troubleshooting ASET Problems	327
	ASET Error Messages	327
21	Auditing Topics	331
22	Auditing Overview	333
	What Is Auditing?	333
	Auditing Terminology	334
	Audit Events	335
	Audit Classes	336
	Audit Directory	337
	How Does Auditing Work?	337
	How is Auditing Related to Security?	338
	How Can I Configure Auditing?	338
	Audit Events, Classes, and Policies	338
	Audit Flags	339
	Audit File Storage Issues	339
	Stored Audit Files vs. Printed Audit Files	340
	Why is <code>/etc/security</code> Important?	340
	Audit Utilities	341
23	Audit Planning	343
	How Does Auditing Work?	343
	What is an Audit Trail?	344
	Audit File Names	344
	Deciding What to Audit	345
	What Files Can I Change?	346
	Determining Which Systems to Audit	349
	Determining Which Audit Policies to Use	350

Determining Audit Events and Classes	353
Determining Where to Store Audit Files	356
Primary and Secondary Audit Directories	357
Setting Up or Modifying Audit Directories	358
Determining Audit Space Usage	358
Device Allocation	360
Managing Device Allocation	360
How the Allocate Mechanism Works	360
Risks Associated with Device Use	362
Using Device Allocation	362
Setting Up an Archive Policy	363
Warnings	364
Determining the Warning Levels	364
The <code>audit_warn</code> Script	365
When Should I Change Audit Parameters?	366
Using the Audit Data	367
Controlling the Cost of Auditing	367
Determining When to Merge Audit Records	368
Changing Audit Trail File Locations	369
Preventing Audit Trail Overflow	370
Using the Results from Audit Trail Analysis	370
How the Audit Trail is Created	370
The Audit Daemon's Role	371
24 Managing Auditing (Tasks)	373
Configuring Audit Files	373
▼ How to Change Audit Flags	373
▼ How to Change User's Audit Characteristics	374
▼ How to Create Audit Classes	375
▼ How to Create Audit Events	375
Configuring an Audit Server	376
▼ How to Create Partitions for Auditing	376
Setting Up Auditing	378
▼ How to Enable Auditing	378
▼ How to Disable Auditing	378

25	Audit Reference	381
	Audit Files	381
	The /etc/system File	382
	The audit_class File	382
	The audit_control File	383
	The audit_data File	385
	The audit_event File	385
	The audit_startup File	385
	The audit_user File	386
	The audit_warn File	387
	The device_maps File	388
	Audit Programs	389
	The audit Program	389
	The auditd Daemon	389
	The auditconfig Program	390
	The auditreduce Program	390
	The praudit Program	391
	Using crontab and atjob	391
	Audit Record Structure	393
	Audit Token Structure	393
	acl token	395
	arbitrary Token	395
	arg Token	396
	attr Token	397
	exec_args Token	397
	exec_env Token	398
	exit Token	399
	file Token	399
	groups Token (Obsolete)	400
	header Token	401
	in_addr Token	401
	ip Token	402
	ipc Token	402
	ipc_perm Token	403
	iport Token	404
	newgroups Token	404
	opaque Token	405

path Token	405
process Token	406
return Token	407
seq Token	408
socket Token	408
subject Token	409
text Token	410
trailer Token	411
Utilities Summary	412

Glossary	415
-----------------	------------

Preface

System Administration Guide: Security Services is part of a six-volume set that covers a significant part of the Solaris™ system administration information. This book assumes that you have already installed the SunOS™ 5.9 operating system, and you have set up any networking software that you plan to use. The SunOS 5.9 operating system is part of the Solaris 9 product family, which also includes many features, including the Solaris Common Desktop Environment (CDE).

Note – The Solaris operating environment runs on two types of hardware, or platforms—SPARC™ and IA. The Solaris operating environment runs on both 64-bit and 32-bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems running the Solaris 9 release. To use this book, you should have 1-2 years of UNIX® system administration experience. Attending UNIX system administration training courses might be helpful.

How the System Administration Volumes Are Organized

Here is a list of the topics covered by the volumes of the System Administration Guides.

System Administration Guide: Basic Administration

- “Solaris Administration Tool Roadmap” in *System Administration Guide: Basic Administration*
- “Working With the Solaris Management Console Tools (Tasks)” in *System Administration Guide: Basic Administration*
- “Managing Users and Groups Topics” in *System Administration Guide: Basic Administration*
- “Managing Server and Client Support Topics” in *System Administration Guide: Basic Administration*
- “Shutting Down and Booting a System Topics” in *System Administration Guide: Basic Administration*
- “Managing Removable Media Topics” in *System Administration Guide: Basic Administration*
- “Managing Software Topics” in *System Administration Guide: Basic Administration*
- “Managing Devices Topics” in *System Administration Guide: Basic Administration*
- “Managing Disks Topics” in *System Administration Guide: Basic Administration*
- “Managing File Systems Topics” in *System Administration Guide: Basic Administration*
- “Backing Up and Restoring Data Topics” in *System Administration Guide: Basic Administration*

System Administration Guide: Advanced Administration

- “Managing Printing Services Topics” in *System Administration Guide: Advanced Administration*

- “Managing Terminals and Modems Topics” in *System Administration Guide: Advanced Administration*
- “Managing System Resources Topics” in *System Administration Guide: Advanced Administration*
- “Managing System Performance Topics” in *System Administration Guide: Advanced Administration*
- “Troubleshooting Solaris Software Topics” in *System Administration Guide: Advanced Administration*

System Administration Guide: IP Services

- “TCP/IP Topics” in *System Administration Guide: IP Services*
- “DHCP Topics” in *System Administration Guide: IP Services*
- “IPv6 Topics” in *System Administration Guide: IP Services*
- “IP Security Topics” in *System Administration Guide: IP Services*
- “Mobile IP Topics” in *System Administration Guide: IP Services*
- “IP Network Multipathing Topics” in *System Administration Guide: IP Services*

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

- “Overview of Naming and Directory Services” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- “The Name Service Switch” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- “Introduction to DNS” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- “Network Information Service (NIS): An Overview” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- “NIS+: An Introduction” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- “LDAP: An Overview” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- “Administering FNS: Attributes Overview” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*

System Administration Guide: Resource Management and Network Services

- “Solaris 9 Resource Manager Topics” in *System Administration Guide: Resource Management and Network Services*
- “SLP Topics” in *System Administration Guide: Resource Management and Network Services*
- “Mail Services Topics” in *System Administration Guide: Resource Management and Network Services*
- “Accessing Remote File Systems Topics” in *System Administration Guide: Resource Management and Network Services*
- “Modem-Related Network Services Topics” in *System Administration Guide: Resource Management and Network Services*
- “Working With Remote Systems Topics” in *System Administration Guide: Resource Management and Network Services*

System Administration Guide: Security Services

- “Security Services Overview” in *System Administration Guide: Security Services*
- “Using Authentication Services (Tasks)” in *System Administration Guide: Security Services*
- “Using Secure Shell (Tasks)” in *System Administration Guide: Security Services*
- “Introduction to SEAM” in *System Administration Guide: Security Services*
- “Managing System Security Topics” in *System Administration Guide: Security Services*
- “Role-Based Access Control (Overview)” in *System Administration Guide: Security Services*
- “Using Automated Security Enhancement Tool (Tasks)” in *System Administration Guide: Security Services*
- “Auditing Overview” in *System Administration Guide: Security Services*

System Administration Guide: Naming and Directory Services (FNS and NIS+)

- “NIS+: An Introduction” in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
- “Federated Naming Service (FNS)” in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*

Related Books

This is a list of related documentation that is referred to in this book.

- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security*. Addison Wesley, 1994.

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at <http://www1.fatbrain.com/documentation/sun>.

Accessing Sun Documentation Online

The docs.sun.comSM Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is <http://docs.sun.com>.

Typographic Conventions

The following table describes the typographic changes used in this book.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	<code>machine_name%</code> su Password:
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type rm <i>filename</i> .
<i>AaBbCc123</i>	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You must be <i>root</i> to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-2 Shell Prompts

Shell	Prompt
C shell prompt	<code>machine_name%</code>
C shell superuser prompt	<code>machine_name#</code>
Bourne shell and Korn shell prompt	<code>\$</code>
Bourne shell and Korn shell superuser prompt	<code>#</code>

Security Services Overview

This book focuses on the Solaris™ Operating Environment features that can help make a site more secure. It is intended for system administrators and users of these security products. It discusses these topics:

- “Introduction to Security” on page 23
- “Authentication” on page 24
- “Access Control” on page 24
- “Secure Communication” on page 25
- “Auditing” on page 25

Introduction to Security

To help an organization secure its computing environment, the Solaris Operating Environment software provides:

- **Authentication**—The ability to securely identify a user, requiring the user’s name and some form of proof (typically a password) .
- **Access Control**—Restricting users to only those parts of the system necessary for their job.
- **Secure Communication**—Ensuring that authenticated parties can communicate without interception, modification, or spoofing.
- **Auditing**—The ability to identify the source of security changes to the system, including file access, security-related system calls, and authentication failures.

For a general discussion of system security, see Chapter 14.

Authentication

Authentication is a mechanism that identifies a user or service based on predefined criteria. Authentication systems range from simple name-password pairs to more elaborate challenge-response systems, such as smart cards and biometrics. Strong authentication mechanisms rely on a user supplying information that only that person knows, such as a username, and something that can be verified, such as a smart card or fingerprint. The Solaris Operating Environment features for authentication include:

- **Secure RPC**—An authentication technique based on the Diffie-Hellman method. This is covered in “Overview of Secure RPC” on page 29.
- **Pluggable Authentication Module (PAM)**—A framework that enables various authentication technologies to be plugged in without disturbing system entry services, such as `login` or `ftp`. See “PAM (Overview)” on page 38.
- **Sun Enterprise Authentication Module (SEAM)**—A client/server architecture that provides authentication with encryption. See Chapter 6.
- **Smart Card**—A plastic card with a microprocessor and memory that can be used with a card reader to access systems. See *Solaris Smartcard Administration Guide*.
- **Login Administration Tools**—Various commands for administering a user’s ability to log in or to abort a session. See Chapter 16.

Access Control

Access control enables users or administrators to restrict the users permitted to access resources on the system. The Solaris Operating Environment features for access control include:

- **UNIX® permissions**—Attributes of a file or directory that control the users and groups permitted to read, write, or execute a file, or search a directory. See Chapter 15.
- **Role-Based Access Control (RBAC)**—An architecture for creating special, restricted user accounts permitted to perform specific security-related tasks. See Chapter 17.
- **Device Allocation**—A facility that enables restriction on who can use a device, such as a floppy or CD-ROM drive. It ensures that a device is used by only one qualified user at a time. See `allocate(1)`.
- **Security Enhancement**—Through the use of scripts, system files and parameters can be adjusted to reduce security risks. See Chapter 20.

Secure Communication

The basis of secure communication is requiring authentication with encryption. Authentication helps ensure that the source and destination are the intended parties. Encryption codes the communication at the source and decodes it at the target to prevent intruders from reading any transmissions they might manage to intercept. The Solaris Operating Environment features for secure communication include:

- **Sun Enterprise Authentication Module (SEAM)**—A client/server architecture that provides encryption with authentication. See Chapter 6.
- **Internet Protocol Security Architecture (IPsec)**—An architecture providing IP datagram protection including confidentiality, strong integrity of the data, partial sequence integrity (replay protection), and data authentication. See “IPsec (Overview)” in *System Administration Guide: IP Services*.
- **Solaris Secure Shell**—A protocol for protecting data transfers and interactive user network sessions from eavesdropping, session hijacking, and man-in-the-middle attacks. Strong authentication is provided through public key cryptography. X-windows and other network services can be tunneled safely over secure shell connections for additional protection. See Chapter 4.

Auditing

Auditing is a fundamental concept of security and maintainability. It is the process of examining the history of actions and events on a system to find out what happened. Auditing entails keeping a log of what was done, by whom, when it was done, and what was affected. For more information on Solaris Operating Environment auditing, see Chapter 22.

Authentication Services Topics

Chapter 3	Provides information about Diffie-Hellman authentication and the Pluggable Authentication Module (PAM) framework.
Chapter 4	Provides a introduction to Solaris secure shell, as well as step-by-step instructions.
Chapter 5	Provides a description of the files that are used to configure Solaris secure shell.
Chapter 6	Provides overview information about SEAM.
Chapter 7	Provides a list of information or issues that need to be resolved before configuring SEAM.
Chapter 8	Provides step-by-step configuration instructions for SEAM.
Chapter 9	Provides a list of SEAM error messages, how to fix the condition that generates the messages and how to troubleshoot some error conditions.
Chapter 10	Provides step-by-step instructions for administering principles and policies with <code>gkadmin</code> and at the command line.
Chapter 11	Provides user instructions for SEAM.
Chapter 12	Provides additional information about SEAM.

Using Authentication Services (Tasks)

The first section of this chapter provides information about the Diffie-Hellman authentication mechanism that may be used with Secure RPC. The second section covers the Pluggable Authentication Module (PAM) framework. PAM provides a method to “plug-in” authentication services and provides support for multiple authentication services.

This is a list of the step-by-step instructions in this chapter.

- “How to Restart the Keyserver” on page 34
- “How to Set Up a Key in NIS+ Credentials for Diffie-Hellman Authentication” on page 35
- “How to Set Up a root Key Using NIS Credentials With Diffie-Hellman Authentication” on page 36
- “How to Share and Mount Files With Diffie-Hellman Authentication” on page 38
- “How to Add a PAM Module” on page 42
- “How to Prevent Unauthorized Access From Remote Systems With PAM” on page 43
- “How to Initiate PAM Error Reporting” on page 43

Overview of Secure RPC

Secure RPC is a method of authentication that authenticates both the host and the user making a request. Secure RPC uses Diffie-Hellman. This authentication mechanisms use DES encryption. Applications that use Secure RPC include NFS and the NIS+ name service.

NFS Services and Secure RPC

The NFS software enables several hosts to share files over the network. Under the NFS system, a server holds the data and resources for several clients. The clients have access to the file systems that the server shares with the clients. Users logged in to the client machine can access the file systems by mounting them from the server. To the user on the client machine, it appears as if the files are local to the client. One of the most common uses of the NFS environment is to allow systems to be installed in offices, while keeping all user files in a central location. Some features of the NFS system, such as the `mount -nosuid` option, can be used to prohibit the opening of devices as well as file systems by unauthorized users.

The NFS environment uses Secure RPC to authenticate users who make requests over the network. This is known as Secure NFS. The authentication mechanism, `AUTH_DH`, uses DES encryption with Diffie-Hellman authentication to ensure authorized access. The `AUTH_DH` mechanism has also been called `AUTH_DES`.

“Administering the Secure NFS System” in *System Administration Guide: Resource Management and Network Services* describes how to set up and administer Secure NFS. Setting up the NIS+ tables and entering names in the `cred` table are discussed in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. See “Implementation of Diffie-Hellman Authentication” on page 31 for an outline of the steps involved in RPC authentication.

DES Encryption

The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt data. If two credential users (or principals) know the same DES key, they can communicate in private, using the key to encipher and decipher text. DES is a relatively fast encryption mechanism. A DES chip makes the encryption even faster; but if the chip is not present, a software implementation is substituted.

The risk of using just the DES key is that an intruder can collect enough cipher-text messages encrypted with the same key to be able to discover the key and decipher the messages. For this reason, security systems such as Secure NFS change the keys frequently.

Kerberos Authentication

Kerberos is an authentication system developed at MIT. Encryption in Kerberos is based on DES. Kerberos V4 support is no longer supplied as part of Secure RPC, but a client-side implementation of Kerberos V5, which uses `RPCSEC_GSS`, is included with the Solaris 8 release. For more information see [**link to security book—SEAM overview](#).

Diffie-Hellman Authentication

The Diffie-Hellman method of authenticating a user is non-trivial for an intruder to crack. The client and the server each has its own private key (sometimes called a secret key) which they use together with the public key to devise a common key. They use the common key to communicate with each other, using an agreed-upon encryption/decryption function (such as DES). This method was identified as DES authentication in previous Solaris releases.

Authentication is based on the ability of the sending system to use the common key to encrypt the current time, which the receiving system can decrypt and check against its current time. Make sure you synchronize the time on the client and the server.

The public and private keys are stored in an NIS or NIS+ database. NIS stores the keys in the `publickey` map, and NIS+ stores the keys in the `cred` table. These files contain the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS or NIS+ tables and generating a public key and a private key for each user. The private key is stored encrypted with the user's password. This makes the private key known only to the user.

Implementation of Diffie-Hellman Authentication

This section describes the series of transactions in a client-server session using DH authorization (`AUTH_DH`).

Generating the Public and Secret Keys

Sometime prior to a transaction, the administrator runs either the `newkey` or `nisaddcred` commands that generates a public key and a secret key. (Each user has a unique public key and secret key.) The public key is stored in a public database; the secret key is stored in encrypted form in the same database. To change the key pair, use the `chkey` command.

Running the keylogin Command

Normally, the login password is identical to the secure RPC password. In this case, a `keylogin` is not required. If the passwords are different, the users have to log in, and then do a `keylogin` explicitly.

The `keylogin` program prompts the user for a secure RPC password and uses the password to decrypt the secret key. The `keylogin` program then passes the decrypted secret key to a program called the `keyserver`. (The `keyserver` is an RPC service with a

local instance on every computer.) The keyserver saves the decrypted secret key and waits for the user to initiate a secure RPC transaction with a server.

If the passwords are the same, the login process passes the secret key to the keyserver. If the passwords are required to be different and the user must always run `keylogin`, then the `keylogin` program may be included in the user's environment configuration file, such as `~/.login`, `~/.cshrc`, or `~/.profile`, so that it runs automatically whenever the user logs in.

Generating the Conversation Key

When the user initiates a transaction with a server:

1. The keyserver randomly generates a conversation key.
2. The kernel uses the conversation key to encrypt the client's time stamp (among other things).
3. The keyserver looks up the server's public key in the public-key database (see the `publickey(4)` man page).
4. The keyserver uses the client's secret key and the server's public key to create a common key.
5. The keyserver encrypts the conversation key with the common key.

First Contact With the Server

The transmission including the encrypted time stamp and the encrypted conversation key is then sent to the server. The transmission includes a credential and a verifier. The credential contains three components:

- The client's net name
- The conversation key, encrypted with the common key
- A "window," encrypted with the conversation key

The window is the difference the client says should be allowed between the server's clock and the client's time stamp. If the difference between the server's clock and the time stamp is greater than the window, the server would reject the client's request. Under normal circumstances this will not happen, because the client first synchronizes with the server before starting the RPC session.

The client's verifier contains:

- The encrypted time stamp
- An encrypted verifier of the specified window, decremented by 1

The window verifier is needed in case somebody wants to impersonate a user and writes a program that, instead of filling in the encrypted fields of the credential and verifier, just stuffs in random bits. The server will decrypt the conversation key into

some random key and use it to try to decrypt the window and the time stamp. The result will be random numbers. After a few thousand trials, however, there is a good chance that the random window/time stamp pair will pass the authentication system. The window verifier makes guessing the right credential much more difficult.

Decrypting the Conversation Key

When the server receives the transmission from the client:

1. The keyserver local to the server looks up the client's public key in the publickey database.
2. The keyserver uses the client's public key and the server's secret key to deduce the common key—the same common key computed by the client. (Only the server and the client can calculate the common key because doing so requires knowing one secret key or the other.)
3. The kernel uses the common key to decrypt the conversation key.
4. The kernel calls the keyserver to decrypt the client's time stamp with the decrypted conversation key.

Storing Information on the Server

After the server decrypts the client's time stamp, it stores four items of information in a credential table:

- The client's computer name
- The conversation key
- The window
- The client's time stamp

The server stores the first three items for future use. It stores the time stamp to protect against replays. The server accepts only time stamps that are chronologically greater than the last one seen, so any replayed transactions are guaranteed to be rejected.

Note – Implicit in these procedures is the name of the caller, who must be authenticated in some manner. The keyserver cannot use DES authentication to do this because it would create a deadlock. To solve this problem, the keyserver stores the secret keys by UID and grants requests only to local root processes.

Verifier Returned to the Client

The server returns a verifier to the client, which includes:

- The index ID, which the server records in its credential cache
- The client's time stamp minus 1, encrypted by conversation key

The reason for subtracting 1 from the time stamp is to ensure that the time stamp is invalid and cannot be reused as a client verifier.

Client Authenticates the Server

The client receives the verifier and authenticates the server. The client knows that only the server could have sent the verifier because only the server knows what time stamp the client sent.

Additional Transactions

With every transaction after the first, the client returns the index ID to the server in its second transaction and sends another encrypted time stamp. The server sends back the client's time stamp minus 1, encrypted by the conversation key.

Administering Diffie-Hellman Authentication

A system administrator can implement policies that help secure the network. The level of security required will differ with each site. This section provides instructions for some tasks associated with network security.

▼ How to Restart the Keyserver

1. **Become superuser.**
2. **Verify whether the `keyserv` daemon (the keyserver) is running.**

```
# ps -ef | grep keyserv
root  100      1 16  Apr 11  ?                0:00 /usr/sbin/keyserv
root   2215    2211  5  09:57:28 pts/0        0:00 grep keyserv
```

3. **Start the keyserver if it isn't running.**

```
# /usr/sbin/keyserv
```

▼ How to Set Up a Key in NIS+ Credentials for Diffie-Hellman Authentication

For detailed description of NIS+ security, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

1. Become superuser.

2. Edit the `/etc/nsswitch.conf` file and add the following line:

```
publickey: nisplus
```

3. Initialize the NIS+ client.

```
# nisinit -cH hostname
```

hostname is the name of a trusted NIS+ server that contains an entry in its tables for the client machine.

4. Add the client to the `cred` table by typing the following commands.

```
# nisaddcred local
# nisaddcred des
```

5. Verify the setup by using the `keylogin` command.

If you are prompted for a password, the procedure has succeeded.

Example—Setting Up a New Key for root on a NIS+ Client

The following example uses the host `pluto` to set up `earth` as an NIS+ client. You can ignore the warnings. The `keylogin` command is accepted, verifying that `earth` is correctly set up as a secure NIS+ client.

```
# nisinit -cH pluto
NIS Server/Client setup utility.
This machine is in the North.Abc.COM. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@North.Abc.COM
Adding new key for unix.earth@North.Abc.Com (earth.North.Abc.COM.)

Network password: xxx <Press Return>
Warning, password differs from login password.
Retype password: xxx <Press Return>

# keylogin
Password:
#
```

How to Set Up a New User Key Using NIS+ Credentials for Diffie-Hellman Authentication

1. Add the user to the `cred` table on the root master server by typing the following command:

```
# nisaddcred -p unix.UID@domainname -P username.domainname. des
```

Note that, in this case, the *username-domainname* must end with a dot (.)

2. Verify the setup by logging in as the client and typing the `keylogin` command.

Example—Setting Up a New Key for an NIS+ User

The following example gives DES security authorization to user `george`.

```
# nisaddcred -p unix.1234@North.Abc.com -P george.North.Abc.COM. des
DES principal name : unix.1234@North.Abc.COM
Adding new key for unix.1234@North.Abc.COM (george.North.Abc.COM.)
```

```
Password:
```

```
Retype password:
```

```
# rlogin rootmaster -l george
```

```
# keylogin
```

```
Password:
```

```
#
```

▼ How to Set Up a root Key Using NIS Credentials With Diffie-Hellman Authentication

1. Become superuser on the client.
2. Edit the `/etc/nsswitch.conf` file and add the following line:

```
publickey: nis
```

3. Create a new key pair by using the `newkey` command.

```
# newkey -h hostname
```

hostname is the name of the client.

Example—Setting Up an NIS+ Client to Use Diffie-Hellman Security

The following example sets up earth as a secure NIS client.

```
# newkey -h earth
Adding new key for unix.earth@North.Abc.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

How to Create a New User Key Using NIS Credentials with Diffie-Hellman Authentication

1. Log in to the server as superuser.

Only the system administrator, logged in to the NIS+ server, can generate a new key for a user.

2. Create a new key for a user.

```
# newkey -u username

username is the name of the user. The system prompts for a password. The system administrator can type a generic password. The private key is stored encrypted with the generic password.

# newkey -u george
Adding new key for unix.12345@Abc.North.Acme.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

3. Tell the user to log in and type the `chkey -p` command.

This allows the user to re-encrypt their private key with a password known only to the user.

```
earth% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@Abc.North.Acme.COM
Please enter the Secure-RPC password for george:
Please enter the login password for george:
Sending key change request to pluto...
#
```

Note – The `chkey` command can be used to create a new key-pair for a user.

▼ How to Share and Mount Files With Diffie-Hellman Authentication

Prerequisite

The Diffie-Hellman `publickey` authentication must be enabled on the network. See “How to Set Up a Key in NIS+ Credentials for Diffie-Hellman Authentication” on page 35 and “How to Set Up a root Key Using NIS Credentials With Diffie-Hellman Authentication” on page 36.

To share a file system with Diffie-Hellman authentication:

1. **Become superuser.**
2. **Share the file system with Diffie-Hellman authentication.**

```
# share -F nfs -o sec=dh /filesystem
```

To mount a file system with Diffie-Hellman authentication:

1. **Become superuser.**
2. **Mount the file system with Diffie-Hellman authentication.**

```
# mount -F nfs -o sec=dh server:resource mountpoint
```

The `-o sec=dh` option mounts the file system with `AUTH_DH` authentication.

PAM (Overview)

The Pluggable Authentication Module (PAM) framework lets you “plug in” new authentication technologies without changing system entry services such as `login`, `ftp`, `telnet`, and so on. You can also use PAM to integrate UNIX login with other security mechanisms like DCE or Kerberos. Mechanisms for account, session, and password management can also be “plugged in” using this framework.

Benefits of Using PAM

The PAM framework allows a system administrator to choose any combination of system entry services (`ftp`, `login`, `telnet`, or `rsh`, for example) for user authentication. Some of the benefits PAM provides are:

- Flexible configuration policy
 - Per application authentication policy
 - The ability to choose a default authentication mechanism
 - Multiple passwords on high-security systems
- Ease of use for the end user
 - No retyping of passwords if they are the same for different mechanisms
 - The ability to use a single password for multiple authentication methods with the password mapping feature, even if the passwords associated with each authentication method are different
 - The ability to prompt the user for passwords for multiple authentication methods without having the user enter multiple commands
- The ability to pass optional parameters to the user authentication services

PAM Components

The PAM software consists of a library, several modules, and a configuration file. New versions of several system entry commands or daemons which take advantage of the PAM interfaces are also included.

The figure below illustrates the relationship between the applications, the PAM library, the `pam.conf` file, and the PAM modules.

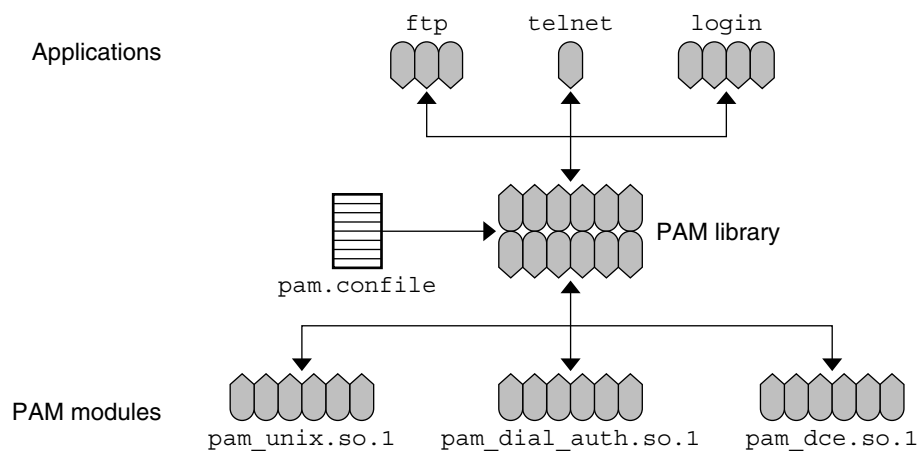


FIGURE 3-1 How PAM Works

The applications (*ftp*, *telnet*, and *login*) use the PAM library to access the appropriate module. The *pam.conf* file defines which modules to use, and in what order they are to be used with each application. Responses from the modules are passed back through the library to the application.

The following sections describe this relationship.

PAM Library

The PAM library, */usr/lib/libpam*, provides the framework to load the appropriate modules and manage the stacking process. It provides a generic structure to which all of the modules can plug in.

Stacking Feature

The PAM framework provides a method for authenticating users with multiple services using *stacking*. Depending on the configuration, the user can be prompted for passwords for each authentication method. The order in which the authentication services are used is determined through the PAM configuration file.

Password-Mapping Feature

The stacking method can require that a user remember several passwords. With the *password-mapping* feature, the primary password is used to decrypt the other

passwords, so the user doesn't need to remember or enter multiple passwords. The other option is to synchronize the passwords across each authentication mechanism. Note that this could increase the security risk, since the security of each mechanism is limited by the least secure password method used in the stack.

PAM (Tasks)

The section below discusses some of the tasks that may be required to make the PAM framework fully functional. In particular, you should be aware of some of the security issues associated with the PAM configuration file.

PAM (Task Map)

TABLE 3-1 PAM Task Map

Task	Description	For Instructions, Go To ...
Plan for your PAM Installation	Consider configuration issues and make decisions about them before starting the software configuration process.	"Planning for PAM" on page 41
(Optional) Add new PAM modules	Sometimes site-specific modules must be written and installed to cover requirements that are not part of the generic software. This procedure covers the installation process.	"How to Add a PAM Module" on page 42
(Optional) Block access through <code>~/ .rhosts</code>	Steps to further increase security by preventing access through <code>~/ .rhosts</code> .	"How to Prevent Unauthorized Access From Remote Systems With PAM" on page 43
(Optional) Initiate error reporting	Steps to start reporting of PAM error messages through <code>syslog</code> .	"How to Initiate PAM Error Reporting" on page 43

Planning for PAM

When deciding how best to employ PAM in your environment, start by focusing on these issues:

- Determine what your needs are, especially which modules you should select.
- Identify the services that need special attention; use `OTHER` if appropriate.
- Decide on the order in which the modules should be run.

- Select the control flag for that module.
- Choose any options necessary for the module.
Here are some suggestions to consider before changing the configuration file:
 - Use the `OTHER` entry for each module type so that every application does not have to be included.
 - Make sure to consider the security implications of the `sufficient` and `optional` control flags.
 - Review the man pages associated with the modules to understand how each module will function, what options are available, and the interactions between stacked modules.



Caution – If the PAM configuration file is misconfigured or gets corrupted, it is possible that even the superuser would be unable to log in. Since `sulogin` does not use PAM, the superuser would then be required to boot the machine into single user mode and fix the problem.

After changing the `/etc/pam.conf` file, review it as much as possible while still logged in as superuser. Test all of the commands that might have been affected by your changes. For example, if you added a new module to the `telnet` service, use the `telnet` command and verify that the changes you made behave as expected.

▼ How to Add a PAM Module

1. **Become superuser.**
2. **Determine which control flags and other options should be used.**
Refer to “PAM Modules” on page 44 information on the module.
3. **Copy the new module to `/usr/lib/security/sparcv9`.**
In the Solaris 8 release, the module should be moved to `/usr/lib/security`.
4. **Set the permissions so that the module file is owned by `root` and permissions are `555`.**
5. **Edit the PAM configuration file, `/etc/pam.conf`, and add this module to the appropriate services.**

Verification

It is very important to do some testing *before* the system is rebooted in case the configuration file is misconfigured. Run `rlogin`, `su`, and `telnet` before rebooting the

system. If the service is a daemon spawned only once when the system is booted, it may be necessary to reboot the system before you can verify that the module has been added.

▼ How to Prevent Unauthorized Access From Remote Systems With PAM

Remove the `rlogin auth rhosts_auth.so.1` entry from the PAM configuration file. This prevents reading the `~/.rhosts` files during an `rlogin` session and therefore prevents unauthenticated access to the local system from remote systems. All `rlogin` access requires a password, regardless of the presence or contents of any `~/.rhosts` or `/etc/hosts.equiv` files.

Note – To prevent other unauthenticated access to the `~/.rhosts` files, remember to disable the `rsh` service. The best way to disable a service is to remove the service entry from `/etc/inetd.conf`. Changing the PAM configuration file does not prevent the service from being started.

▼ How to Initiate PAM Error Reporting

1. **Edit the `/etc/syslog.conf` to add any of the following PAM error reporting entries:**
 - `auth.alert` — messages about conditions that should be fixed immediately
 - `auth.crit` — critical messages
 - `auth.err` — error messages
 - `auth.info` — informational messages
 - `auth.debug` — debugging messages
2. **Restart the `syslog` daemon or send a `SIGHUP` signal to it to activate the PAM error reporting.**

Example—Initiating PAM Error Reporting

The example below displays all alert messages on the console. Critical messages are mailed to root. Informational and debug messages are added to the `/var/log/pamlog` file.

```
auth.alert    /dev/console
auth.crit     'root'
auth.info;auth.debug  /var/log/pamlog
```

Each line in the log contains a time stamp, the name of the system that generated the message, and the message itself. The `pamlog` file is capable of logging a large amount of information.

PAM (Reference)

PAM employs run-time pluggable modules to provide authentication for system entry services. These modules are broken into four different types based on their function: authentication, account management, session management, and password management. A stacking feature is provided to let you authenticate users through multiple services, as well as a password-mapping feature to not require that users remember multiple passwords.

PAM Modules

Each PAM module implements a specific mechanism. When setting up PAM authentication, you need to specify both the module and the module type, which defines what the module will do. More than one module type (auth, account, session, or password) may be associated with each module.

The following list describes each of the PAM modules. The path each module is determined by the instruction set that is available in the Solaris release that is installed. For instance, the value for `$ISA` could be `sparc` or `i386`. See the `isalist(5)` man page for more information.

TABLE 3-2 PAM Modules

Module Name and Location	Description
<code>dial_auth</code> <code>/usr/lib/security/\$ISA/pam_dial_auth.so.1</code>	Can only be used for authentication. It uses data stored in the <code>/etc/dialups</code> and <code>/etc/d_passwd</code> files for authentication. This is mainly used by <code>login</code> . See <code>pam_dial_auth(5)</code> for more information.
<code>krb5</code> <code>/usr/lib/security/\$ISA/pam_krb5_auth.so.1</code>	Provides support for authentication, account management, session management, and password management. Kerberos credentials are used for authentication. See <code>pam_krb5(5)</code> for more information.
<code>ldap</code> <code>/usr/lib/security/\$ISA/pam_ldap.so.1</code>	Provides support for authentication and password management. Data from an LDAP server are used for authentication. See <code>pam_ldap(5)</code> for more information.

TABLE 3-2 PAM Modules (Continued)

Module Name and Location	Description
projects /usr/lib/security/\$ISA/pam_projects.so.1	Provides support for account management. See <code>pam_projects(5)</code> for more information.
rhosts_auth /usr/lib/security/\$ISA/pam_rhosts_auth.so.1	Can only be used for authentication. It uses data stored in the <code>~/.rhosts</code> and <code>/etc/host.equiv</code> files through <code>ruserok()</code> . This is mainly used by the <code>rlogin</code> and <code>rsh</code> commands. See <code>pam_rhosts_auth(5)</code> for more information.
roles /usr/lib/security/\$ISA/pam_roles.so.1	Provides support for account management only. The RBAC <code>user_attr</code> database is to determine the roles a user can assume. See <code>pam_roles(5)</code> for more information.
sample /usr/lib/security/\$ISA/pam_sample.so.1	Provides support for authentication, account management, session management, and password management. Used for testing. See <code>pam_sample(5)</code> for more information.
smartcard /usr/lib/security/\$ISA/pam_smartcard.so.1	Provide support for authentication only. See <code>pam_smartcard(5)</code> for more information.
unix /usr/lib/security/\$ISA/pam_unix.so.1	Provides support for authentication, account management, session management, and password management. Any of the four module type definitions can be used with this module. It uses UNIX passwords for authentication. In the Solaris environment, the selection of appropriate name services to get password records is controlled through the <code>/etc/nsswitch.conf</code> file. See <code>pam_unix(5)</code> for more information.

For security reasons, these module files must be owned by `root` and must not be writable through `group` or `other` permissions. If the file is not owned by `root`, PAM will not load the module.

PAM Module Types

It is important to understand the PAM module types because the module type defines the interface to the module. These are the four types of run-time PAM modules:

- The *authentication modules* provide authentication for the users and allow for credentials to be set, refreshed, or destroyed. They provide a valuable administration tool for user identification.
- The *account modules* check for password aging, account expiration, and access hour restrictions. After the user is identified through the authentication modules, the

account modules determine if the user should be given access.

- The *session modules* manage the opening and closing of an authentication session. They can log activity or provide for clean-up after the session is over.
- The *password modules* allow for changes to the actual password.

PAM Configuration File

The PAM configuration file, `/etc/pam.conf`, determines the authentication services to be used, and in what order they are used. This file can be edited to select authentication mechanisms for each system-entry application.

Configuration File Syntax

The PAM configuration file consists of entries with the following syntax:

service_name module_type control_flag module_path module_options

<i>service_name</i>	Name of the service (for example, <code>ftp</code> , <code>login</code> , <code>telnet</code>).
<i>module_type</i>	Module type for the service.
<i>control_flag</i>	Determines the continuation or failure semantics for the module.
<i>module_path</i>	Path to the library object that implements the service functionality.
<i>module_options</i>	Specific options that are passed to the service modules.

You can add comments to the `pam.conf` file by starting the line with a # (pound sign). Use white spaces or tabs to delimit the fields.

Note – An entry in the PAM configuration file is ignored if one of the following conditions exist: the line has less than four fields, an invalid value is given for *module_type* or *control_flag*, or the named module is not found.

Valid Service Names

The table below lists some of the valid service names, the module types that can be used with that service, and the daemon or command associated with the service name.

There are several module types that are not appropriate for each service. For example, the `password` module type is only specified to go with the `passwd` command. There is no `auth` module type associated with this command since it is not concerned with authentication.

TABLE 3-3 Valid Service Names for `/etc/pam.conf`

Service Name	Daemon or Command	Module Type
<code>cron</code>	<code>/usr/sbin/cron</code>	<code>auth, account</code>
<code>dtlogin</code>	<code>/usr/dt/bin/dtlogin</code>	<code>auth, account, session</code>
<code>dtsession</code>	<code>/usr/dt/bin/dtsession</code>	<code>auth</code>
<code>ftp</code>	<code>/usr/sbin/in.ftpd</code>	<code>auth, account, session</code>
<code>init</code>	<code>/usr/sbin/init</code>	<code>session</code>
<code>login</code>	<code>/usr/bin/login</code>	<code>auth, account, session</code>
<code>passwd</code>	<code>/usr/bin/passwd</code>	<code>password</code>
<code>ppp</code>	<code>/usr/bin/ppp</code>	<code>auth, account, session</code>
<code>rexcd</code>	<code>/usr/sbin/rpc.rexd</code>	<code>account, session</code>
<code>rlogin</code>	<code>/usr/sbin/in.rlogind</code>	<code>auth, account, session</code>
<code>rsh</code>	<code>/usr/sbin/in.rshd</code>	<code>auth, account, session</code>
<code>sac</code>	<code>/usr/lib/saf/sac</code>	<code>session</code>
<code>ssh</code>	<code>/usr/bin/ssh</code>	<code>auth, account, session</code>
<code>su</code>	<code>/usr/bin/su</code>	<code>auth, account</code>
<code>telnet</code>	<code>/usr/sbin/in.telnetd</code>	<code>auth, account, session</code>
<code>ttymon</code>	<code>/usr/lib/saf/ttymon</code>	<code>session</code>
<code>uucp</code>	<code>/usr/sbin/in.uucpd</code>	<code>auth, account, session</code>

Control Flags

To determine continuation or failure behavior from a module during the authentication process, you must select one of four *control flags* for each entry. The control flags indicate how a successful or a failed attempt through each module is handled. Even though these flags apply to all module types, the following explanation assumes that these flags are being used for authentication modules. The control flags are as follows:

- `required` - This module must return success in order to have the overall result be successful.

If all of the modules are labeled as `required`, then authentication through all modules must succeed for the user to be authenticated.

If some of the modules fail, then an error value from the first failed module is reported.

If a failure occurs for a module flagged as `required`, all modules in the stack are still tried but failure is returned.

If none of the modules are flagged as `required`, then at least one of the entries for that service must succeed for the user to be authenticated.

- `requisite` - This module must return success for additional authentication to occur.

If a failure occurs for a module flagged as `requisite`, an error is immediately returned to the application and no additional authentication is done. If the stack does not include prior modules labeled as `required` that failed, then the error from this module is returned. If a earlier module labeled as `required` has failed, the error message from the `required` module is returned.

- `optional` - If this module fails, the overall result can be successful if another module in this stack returns success.

The `optional` flag should be used when one success in the stack is enough for a user to be authenticated. This flag should only be used if it is not important for this particular mechanism to succeed.

If your users need to have permission associated with a specific mechanism to get their work done, then you should not label it as `optional`.

- `sufficient` - If this module is successful, skip the remaining modules in the stack, even if they are labeled as `required`.

The `sufficient` flag indicates that one successful authentication will be enough for the user to be granted access.

More information about these flags is provided in the section below, which describes the default `/etc/pam.conf` file.

Generic `pam.conf` File

The generic `/etc/pam.conf` file specifies:

1. When running `login`, authentication must succeed for both the `pam_unix` and the `pam_dial_auth` modules.
2. For `rlogin`, authentication through the `pam_unix` module must succeed, if authentication through `pam_rhost_auth` fails.
3. The `sufficient` control flag indicates that for `rlogin` the successful authentication provided by the `pam_rhost_auth` module is sufficient and the next entry will be ignored.

4. Most of the other commands requiring authentication require successful authentication through the `pam_unix` module.
5. Authentication for `rsh` must succeed through the `pam_rhost_auth` module.

The `OTHER` service name allows a default to be set for any other commands requiring authentication that are not included in the file. The `OTHER` option makes it easier to administer the file, since many commands that are using the same module can be covered using only one entry. Also, the `OTHER` service name, when used as a “catch-all,” can ensure that each access is covered by one module. By convention, the `OTHER` entry is included at the bottom of the section for each module type.

Normally, the entry for the `module_path` is “root-relative.” If the file name you enter for `module_path` does not begin with a slash (/), the path `/usr/lib/security/$ISA` is prepended to the file name. A full path name must be used for modules located in other directories. The values for the `module_options` can be found in the man pages for the module. For example, the UNIX module is covered in the `pam_unix(5)` man page.

The `use_first_pass` and `try_first_pass` options, which are supported by the `pam_unix` module, let users reuse the same password for authentication without retyping it.

If `login` specifies authentication through both `pam_local` and `pam_unix`, then the user is prompted to enter a password for each module. In situations where the passwords are the same, the `use_first_pass` module option prompts for only one password and uses that password to authenticate the user for both modules. If the passwords are different, the authentication fails. In general, this option should be used with an `optional` control flag, as shown below, to make sure that the user can still log in.

```
# Authentication management
#
login  auth  required  /usr/lib/security/$ISA/pam_unix.so.1
login  auth  optional  /usr/lib/security/$ISA/pam_local.so.1  use_first_pass
```

If the `try_first_pass` module option is used instead, the local module prompts for a second password if the passwords do not match or if an error is made. If both methods of authentication are necessary for a user to get access to all the needed tools, using this option could cause some confusion since the user could get access with only one type of authentication.

Using Secure Shell (Tasks)

This chapter covers the following topics:

- “Introduction to Secure Shell” on page 51
- “Using Secure Shell (Task Map)” on page 54
- “How to Create a Public/Private Key Pair” on page 55
- “How to Log Into Another Host Using Secure Shell” on page 56
- “How to Log in with No Password Using ssh-agent” on page 56
- “How to Set ssh-agent to Run Automatically” on page 58
- “How to Use Secure Shell Port Forwarding” on page 58
- “How to Copy Files with Secure Shell” on page 60
- “How to Transfer Files Remotely Using sftp” on page 60
- “How to Connect to Hosts Outside a Firewall from the Command Line” on page 62

Introduction to Secure Shell

Secure shell allows a user to securely access a remote host over an unsecured network. Authentication is provided by password and/or public keys. All network traffic is encrypted. Thus it prevents a would-be intruder from being able to read an intercepted communication or from spoofing the system.

Secure shell provides commands for remote login and file transfer. Secure shell can also be used as an on-demand virtual private network (VPN) to forward X Window system traffic or individual port numbers between the local and remote machines over the encrypted network link.

With secure shell, you can perform these actions:

- Log into another host securely over an unsecured network

- Copy files securely between the two hosts
- Run commands securely on the remote host

Solaris secure shell supports the two versions of the Secure Shell Protocol. Version 1 is the original version of the protocol. Version 2 is more secure and it amends some of the basic security design flaws of Version 1. As a result, Version 1 is deprecated and is provided only to assist users migrating to Version 2. Users are strongly discouraged from using Version 1.

Note – Hereafter in this text, v1 is used to represent Version 1 and v2 to represent Version 2.

The requirements for secure shell authentication are:

- **User authentication** – A user can be authenticated through:
 - **Passwords** – The user supplies the account password as in the login process.
 - **Public keys** – The user can create a public/private key pair that is stored on the local host and the remote hosts are provided with the public key, which is required to complete the authentication.

whereby the source host maintains the private key and target hosts are provided with the public key needed to complete authentication. This is a stronger authentication mechanism, because the private key never travels over the network. The public/private key pair is stored in the user's home directory under the `.ssh` subdirectory. The default names for the public and private keys are shown in the following table.

TABLE 4-1 Naming Conventions for Identity Files

Private Key, Public Key	Cipher and Protocol Version
<code>identity, identity.pub</code>	RSA v1
<code>id_rsa, id_rsa.pub</code>	RSA v2
<code>id_dsa, id_dsa.pub</code>	DSA v2

- **Host authentication** – Host authentication requires the remote host to have access to the local host's public key. A copy of the local host's public key is stored in `$HOME/.ssh/known_hosts` on the remote host.

The following table shows the authentication methods, protocol version compatibility, requirements, and relative security. Note that the default method is password-based authentication.

TABLE 4-2 Authentication Methods

Authentication Method (Protocol Version)	Local Requirements	Remote Requirements	Security Level
password-based (v1 or v2)	user account	user account	Medium
RSA/DSA public key (v2)	user account private key in <code>\$HOME/.ssh/id_rsa</code> or <code>\$HOME/.ssh/id_dsa</code> public key in <code>\$HOME/.ssh/id_rsa.pub</code> or <code>\$HOME/.ssh/id_dsa.pub</code>	user account user's public key (<code>id_rsa.pub</code> or <code>id_dsa.pub</code>) in <code>\$HOME/.ssh/authorized_keys</code>	Strong
RSA public key (v1)	user account private key in <code>\$HOME/.ssh/identity</code> public key in <code>\$HOME/.ssh/identity.pub</code>	user account user's public key (<code>identity.pub</code>) in <code>\$HOME/.ssh/authorized_keys</code>	Strong
.rhosts with RSA (v1)	user account	user account local host name in <code>/etc/hosts.equiv</code> , <code>/etc/shosts.equiv</code> , <code>\$HOME/.rhosts</code> , or <code>\$HOME/.shosts</code> local host public key in <code>\$HOME/.ssh/known_hosts</code> or <code>/etc/ssh/ssh_known_hosts</code>	Medium
.rhosts only (v1 or v2)	user account	user account local host name in <code>/etc/hosts.equiv</code> , <code>/etc/shosts.equiv</code> , <code>\$HOME/.rhosts</code> , or <code>\$HOME/.shosts</code>	Weak

Using Secure Shell (Task Map)

TABLE 4-3 Tasks for Using Secure Shell

Task	Description	For Instructions, Go To ...
Create a public/private key pair	Using public/private key pairs is the preferred method for authenticating yourself and encrypting your communications.	“How to Create a Public/Private Key Pair” on page 55
Logging in using secure shell	Encrypted secure shell communication is enabled by logging in remotely through a process similar to using <code>rsh</code> .	“How to Log Into Another Host Using Secure Shell” on page 56
Passwordless login with secure shell	You can log in using secure shell without having to provide a password, using the <code>ssh-agent</code> . The <code>ssh-agent</code> can be run manually or from a start-up script.	“How to Log in with No Password Using <code>ssh-agent</code> ” on page 56 “How to Set <code>ssh-agent</code> to Run Automatically” on page 58
Port forwarding in a secure shell	You can specify a local or remote port to be used in a secure shell connection.	“How to Use Secure Shell Port Forwarding” on page 58
Copying files using a secure shell	You can copy remote files securely.	“How to Copy Files with Secure Shell” on page 60
Transferring files using a secure shell	You can log into a remote host with secure shell using transfer commands similar to <code>ftp</code> .	“How to Transfer Files Remotely Using <code>sftp</code> ” on page 60
Connecting from a host inside a firewall to a host on the outside	Secure shell provides commands compatible with HTTP or SOCKS5 that can be specified in a configuration file or on the command line.	“How to Set Up Default Connections to Hosts Outside a Firewall” on page 61 “How to Connect to Hosts Outside a Firewall from the Command Line” on page 62

Using Secure Shell

▼ How to Create a Public/Private Key Pair

The standard procedure for creating a secure shell public/private key pair follows. For information on additional options, see `ssh-keygen(1)`.

1. Start the key generation program.

```
myLocalHost% ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key(/home/johndoe/.ssh/id_rsa):
```

2. Enter the path to the file that will hold the key.

By default, the file name `id_rsa`, which represents an RSA v2 key, appears in parentheses. This file can be selected by pressing Return, or an alternative filename can be entered.

```
Enter file in which to save the key(/home/johndoe/.ssh/id_rsa): <Return>
```

The public key name is created automatically by appending the string `.pub` to the private key name.

3. Enter a passphrase for using your key.

This passphrase is used for encrypting your private key. A good passphrase is 10–30 characters long, mixes alpha and numeric characters, and avoids simple English prose and names. A null entry means no passphrase is used, but this is *strongly discouraged* for user accounts. Note that the passphrase is not displayed when you type it in.

```
Enter passphrase(empty for no passphrase): <Type the passphrase>
```

4. Re-enter the passphrase to confirm it.

```
Enter same passphrase again: <Type the passphrase>
Your identification has been saved in /home/johndoe/.ssh/id_rsa.
Your public key has been saved in /home/johndoe/.ssh/id_rsa.pub.
The key fingerprint is:
0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 johndoe@myLocalHost
```

5. Check the results.

The key fingerprint (a colon-separated series of 2–digit hexadecimal values) is displayed. Check that the path to the key is correct. In the example, the path is `/home/johndoe/.ssh/id_rsa.pub`. At this point, you have created a public/private key pair.

6. Copy the public key to your home directory on the remote host and append it to `$HOME/.ssh/authorized_keys`.

▼ How to Log Into Another Host Using Secure Shell

1. Use `ssh`, specifying the name of the remote host.

```
myLocalHost% ssh myRemoteHost
```

The first time you run `ssh`, a prompt comes up questioning the authenticity of the remote host, as follows.

```
The authenticity of host 'myRemoteHost' can't be established.  
RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26  
Are you sure you want to continue connecting(yes/no)?
```

This is normal and you should type **yes** and continue. If you have used `ssh` in the past on this remote host and this warning message still appears, then there may be a breach in your security.

2. Enter the secure shell passphrase and the account password when prompted.

```
Enter passphrase for key '/home/johndoe/.ssh/id_rsa': <Return>  
johndoe@myRemoteHost's password: <Return>  
Last login: Fri Jul 20 14:24:10 2001 from myLocalHost  
myRemoteHost%
```

Conduct any transactions on the remote host. Any commands you send and any responses received will be encrypted.

Note – If you wish to subsequently change your pass-phrase, use `ssh-keygen` with the `-p` option.

3. When you are through with your remote session, type `exit` or use your usual method for exiting your shell.

```
myRemoteHost% exit  
myRemoteHost% logout  
Connection to myRemoteHost closed  
myLocalHost%
```

▼ How to Log in with No Password Using `ssh-agent`

If you want to omit passphrase and password entry when you are using secure shell, you can use the agent daemon. Use the `ssh-agent` command at the beginning of the session. Then store your private key(s) with the agent using `ssh-add`. If you have different accounts on different hosts, add those keys that you intend to use in the session. You can start the agent manually when needed as described below, or you can

set the agent to run automatically at the start of every session as described in “How to Set ssh-agent to Run Automatically” on page 58.

1. Start the agent daemon.

The `ssh-agent` starts the agent daemon and displays its process ID.

```
myLocalHost% eval `ssh-agent`  
Agent pid 9892  
myLocalHost%
```

2. Add your private key to the agent daemon.

The `ssh-add` command adds your private key to the agent daemon so that subsequent secure shell activity will not prompt for the passphrase.

```
myLocalHost% ssh-add  
Enter passphrase for /home/johndoe/.ssh/id_rsa:  
Identity added: /home/johndoe/.ssh/id_rsa (/home/johndoe/.ssh/id_rsa)  
myLocalHost%
```

3. Start a secure shell session.

```
myLocalHost% ssh myRemoteHost
```

Example—Using ssh-add Options

You can use `ssh-add` to add other keys to the daemon as well, for example you may concurrently have DSA v2, RSA v2, and RSA v1 keys. To list all keys stored in the daemon, use the `-l` option. To delete a single key from the daemon, use the `-d` option. To delete all keys, use the `-D` option.

```
myLocalHost% eval `ssh-agent`  
Agent pid 3347  
myLocalHost% ssh-add  
Enter passphrase for /home/johndoe/.ssh/id_rsa:  
Identity added: /home/johndoe/.ssh/id_rsa (/home/johndoe/.ssh/id_rsa)  
myLocalHost% ssh-add /home/johndoe/.ssh/id_dsa  
Enter passphrase for /home/johndoe/.ssh/id_dsa: <type passphrase>  
Identity added:  
/home/johndoe/.ssh/id_dsa (/home/johndoe/.ssh/id_dsa)  
myLocalHost% ssh-add -l  
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1  
/home/johndoe/.ssh/id_rsa (RSA)  
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53  
/home/johndoe/.ssh/id_dsa (DSA)  
myLocalHost% ssh-add -d  
Identity removed:  
/home/johndoe/.ssh/id_rsa (/home/johndoe/.ssh/id_rsa.pub)  
/home/johndoe/.ssh/id_dsa (DSA)
```

▼ How to Set ssh-agent to Run Automatically

If you want to avoid providing your passphrase and password whenever you use secure shell, you can start an agent daemon (`ssh-agent`) using the `.dtprofile` script.

1. To start the agent daemon automatically, add the following lines to the end of `$HOME/.dtprofile`:

```
if [ "$SSH_AUTH_SOCK" = "" -a -x /usr/bin/ssh-agent ]; then
    eval `/usr/bin/ssh-agent`
fi
```

2. To terminate the secure shell agent daemon on exiting the CDE session, add the following to `$HOME/.dt/sessions/sessionexit`:

```
if [ "$SSH_AGENT_PID" != "" -a -x /usr/bin/ssh-agent ]; then
    /usr/bin/ssh-agent -k
fi
```

This ensures that no one can use the secure shell agent after the CDE session is terminated.

3. Start a secure shell session.

```
myLocalHost% ssh myRemoteHost
```

There is no prompt for a passphrase.

▼ How to Use Secure Shell Port Forwarding

You can specify a local port to be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. The connection from this port is made over a secure channel to the remote host. For example, you might specify port 143 to obtain email remotely using IMAP4. Similarly, a port can be specified on the remote side.

1. To set a local port to be forwarded, specify the local port to listen to and the remote host and port to forward to.

```
myLocalHost% ssh -L localPort:remoteHost:remotePort
```

2. To set a remote port to receive a secure connection, specify the remote port to listen to and the local host and port to forward to.

```
myLocalHost% ssh -R remotePort:localHost:localPort
```

EXAMPLE 4-1 Using Local Port Forwarding to Receive Mail

The following example demonstrates how local port forwarding can be used to receive mail securely from a remote server.

EXAMPLE 4-1 Using Local Port Forwarding to Receive Mail (Continued)

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

This step has the effect of forwarding connections to port 9143 on `myLocalHost` to port 143 (the IMAP v2 server port) on `myRemoteHost`. When the user launches a mail application, he or she needs to specify the local port number. An example using `dtmail` is shown in Figure 4-1. (Note that the term *localhost* in this case and in Example 4-2 refers to the keyword designating user's local host. *localhost* should not be confused with *myLocalHost*, the hypothetical host name used to identify a local host in the examples in this chapter.)



FIGURE 4-1 Specifying Port Forwarding for Email

EXAMPLE 4-2 Using Remote Port Forward to Communicate Outside of a Firewall

This example demonstrates how a user in an enterprise environment can forward connections from a host on an external network to a host inside a corporate firewall.

```
myLocalHost% ssh -R 9022:myLocalHost:22 myOutsideHost
```

This step forwards connections to port 9022 on `myOutsideHost` to port 22 (the `sshd` server) on the local host.

```
myOutsideHost% ssh -p 9022 localhost
myLocalHost%
```

This step demonstrates how after the remote forwarding connection has been established, the user can use `ssh` to connect securely from the remote host.

▼ How to Copy Files with Secure Shell

Use the `scp` command to copy encrypted files between hosts, either a local and remote host or between two remote hosts. It operates similarly to `rcp` except that it prompts for passwords. See `scp(1)` for more information.

1. Type `scp`, and specify the source file, user name at remote destination, and directory.

```
myLocalHost% scp myfile.1 johndoe@myRemoteHost:~
```

2. Enter the secure shell passphrase when prompted.

```
Enter passphrase for key '/home/johndoe/.ssh/id_rsa': <Return>
myfile.1      25% |*****          |      640 KB  0:20 ETA
myfile.1
```

After you enter the pass phrase, the progress meter is displayed. See the second line above. The progress meter displays the file name, the percentage of the file that has been transferred at this point, a series of asterisks analogous to the percentage transferred, the quantity of data transferred, and the estimated time of arrival (ETA) of the complete file, that is the remaining amount of time.

▼ How to Transfer Files Remotely Using `sftp`

The `sftp` command works similarly `ftp` but uses a different set of commands. The following table lists some representative commands.

TABLE 4-4 Interactive `sftp` Commands

Category	Commands	Description
Navigation	<code>cd path,</code>	Change remote directory to <i>path</i> .
	<code>lcd path</code>	Change local directory to <i>path</i> .
Ownership	<code>chgrp group file</code>	Change the group for <i>file</i> to <i>group</i> , a numeric group GID.
	<code>chmod mode file</code>	Change the permissions of <i>file</i> .
File Copying	<code>get remote_file [local-path]</code>	Retrieve <i>remote_file</i> and store it on local host.
	<code>put local_file [remote_path]</code>	Store a local file on the remote host.
	<code>rename old_filename new_file</code>	Rename a local file.
Directory listing	<code>ls [path]</code>	List the contents of the remote directory.
Directory creation	<code>mkdir path</code>	Create a remote directory.

TABLE 4-4 Interactive sftp Commands (Continued)

Category	Commands	Description
Miscellaneous	exit, quit	Quit sftp.

▼ How to Set Up Default Connections to Hosts Outside a Firewall

You can use secure shell to make a connection from a host inside a firewall to a host on the other side of the firewall. This is done by specifying a proxy command for `ssh` either in a configuration file or as an option on the command line (see “How to Connect to Hosts Outside a Firewall from the Command Line” on page 62). In general, your `ssh` interactions can be customized through a configuration file, either your own personal file `$HOME/.ssh/config` or an administrative configuration file in `/etc/ssh/ssh_config`. See `ssh_config(4)`. There are two types of proxy commands: for HTTP or for SOCKS5 connections.

1. Specify proxy commands and hosts in a configuration file.

Use the following syntax to add as many lines as you need:

```
[Host outside_host]  
ProxyCommand proxy_command [-h proxy_server] \  
[-p proxy_port] outside_host | %h outside_port | %p
```

Use the `Host outside_host` option to limit this proxy command specification to instances when this host (or hosts if a wildcard is used) is specified on the command line.

The designation `proxy_command` can be replaced by either `/usr/lib/ssh/ssh-http-proxy-connect` for HTTP connections or `/usr/lib/ssh/ssh-socks5-proxy-connect` for SOCKS5 connections.

The `-h proxy_server` and `-p proxy_port` options specify a proxy server and port. If present, they override any environment variables specifying proxy servers and ports, such as `HTTPPROXY`, `HTTPPROXYPORT`, `http_proxy` (for specifying a URL), `SOCKS5_SERVER`, and `SOCKS5_PORT`. If the options are not used, then the relevant environment variables must be set. (See `ssh-socks5-proxy-connect(1)` and `ssh-http-proxy-connect(1)`).

Use `outside_host` to designate a specific host to connect to or use `%h` to specify the host on the command line. Use `outside_port` or `%p` to specify the port. Specifying `%h` and `%p` without using the `Host outside_host` option has the effect of applying the proxy command to the host argument whenever `ssh` is invoked.

2. Run secure shell, specifying the outside host.

For example, type:

```
myLocalHost% ssh myOutsideHost
```

This will look for a proxy command specification for `myOutsideHost` in your personal configuration file and if not found, in the system-wide configuration file, `ssh_config`. The proxy command will be substituted for `ssh`.

How to Connect to Hosts Outside a Firewall from the Command Line

The `-o` option for `ssh` lets you enter any line permitted in an `ssh` configuration file. In this case, the proxy command specification described above is used.

- **Run `ssh` and include a proxy command specification as a `-o` option.**

For example, type

```
ssh -o'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \  
-h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost
```

This will substitute the HTTP proxy command for `ssh`, use port 8080 and `myProxyServer` as the proxy server, and connect to port 22 on `myOutsideHost`.

Secure Shell Administration

This chapter describes how the secure shell works from the administrator's point of view and how it is configured.

- "A Typical Secure Shell Session" on page 63
- "Configuring the Secure Shell" on page 65
- "Maintaining Known Hosts on a Site-Wide Basis" on page 68
- "Secure Shell Files" on page 69

A Typical Secure Shell Session

The secure shell daemon (`sshd`) is normally started at boot from `/etc/init.d/sshd`. It listens for connections from clients. A secure shell session begins when the user runs `ssh`, `scp`, or `sftp`. A new `sshd` daemon is forked for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange with the client. These session characteristics are determined by client-side and server configuration files and potentially command-line parameters. The client and server must authenticate themselves to each other. After successful authentication, the user can execute commands remotely and move data between hosts.

Session Characteristics

The server-side behavior of `sshd` is controlled by keyword settings in the `/etc/ssh/sshd_config` file and potentially the command-line options when `sshd` is started. For example, `sshd_config` controls which types of authentication are permitted for accessing the server.

The behavior on the client side is controlled by the secure shell parameters in this order of precedence:

- command line options
- user's configuration file (`$HOME/.ssh/config`)
- system-wide configuration file (`/etc/ssh/ssh_config`)

For example, a user can override a system-wide configuration `Cipher` set to **blowfish** by specifying `-c 3des` on the command line.

Authentication

The steps in the authentication process are as follows:

1. The user runs `ssh`, `scp`, or `sftp`.
2. The client and server agree on a shared session key.

In v1, the remote host sends its host (RSA) key and a server (RSA) key to the client. (Note that the server key is typically generated every hour and stored in memory only.) The client checks that the remote host key is stored in the `$HOME/.ssh/known_hosts` file on the local host. The client then generates a 256-bit random number and encrypts it using the remote host's host and server keys. The encrypted random number is used as a session key to encrypt all further communications in the session.

In v2, the remote host uses DSA in its host key and does not generate a server key. Instead the shared session key is derived through a Diffie-Hellman agreement.
3. The local and remote hosts authenticate each other.

In v1, the client can use `.rhosts`, `.rhosts` with RSA, RSA challenge-response, or password-based authentication. In v2, only `.rhosts`, DSA and password-based authentication are permitted.

Command Execution and Data Forwarding

After authentication is complete, the user can use the secure shell, generally requesting a shell or executing a command. Through the `ssh` options, the user can make requests, such as allocating a pseudo-tty, forwarding X11 or TCP/IP connections, or enabling an `ssh-agent` over a secure connection. The basic components of a user session are as follows:

1. The user requests a shell or the execution of a command, which begins the session mode.

In this mode, data is sent or received through the terminal on the client side and the shell or command on the server side.

2. The user program terminates.
3. All X11 and TCP/IP forwarding is stopped. Any X11 and TCP/IP connections that already exist remain open.
4. The server sends the command `exit` to the client and both sides exit.

Configuring the Secure Shell

The characteristics of a secure shell session are controlled by configuration files, which can be overridden to a certain degree by parameters on the command line.

Secure Shell Client Configuration

In most cases, the client-side characteristics of a secure shell session are governed by the global user configuration file, `/etc/ssh/ssh_config`, which is set up by the administrator. The settings in the global user configuration file can be overridden by the user's configuration in `$HOME/.ssh/config`. In addition, the user can override both configuration files on the command line.

These parameters represent client requests and are permitted or denied on the server side by the `/etc/ssh/sshd_config` file (see `ssh_config(4)`). The configuration file keywords and command parameters are introduced below and are described in detail in the `ssh(1)`, `scp(1)`, `sftp(1)`, and `ssh_config(4)` man pages. Note that in the two user configuration files, the `Host` keyword indicates a host or wildcard expression to which all following keywords up to the next `Host` keyword apply.

Host-Specific Parameters

If it is useful to have different secure shell characteristics for different local hosts, the administrator can define separate sets of parameters in the `/etc/ssh/ssh_config` file to be applied according to host or regular expression. This is done by grouping entries in the file by `Host` specification. If the `Host` keyword is not used, the keywords in the client configuration file apply to whichever local host a user is working on.

Client-Side Authentication

The authentication method is determined by setting one of the following keywords to "yes": `DSAAuthentication`, `PasswordAuthentication`,

`RhostsAuthentication`, or `RhostsRSAAuthentication`. The keyword `UserRsh` specifies that `rlogin` and `rsh` be used, probably due to no secure shell support.

The `Protocol` keyword sets the secure shell protocol version to `v1` or `v2`. One can specify both versions separated by a comma, the idea being that the first is tried and upon failure the second is used.

`IdentityFile` specifies an alternate file name to hold the user's private key.

The keyword `Cipher` specifies the `v1` encryption algorithm, which may be `blowfish` or `3des`. The keyword `Ciphers` specifies an order of preference for the `v2` encryption algorithms: `3des-cbc`, `blowfish-cbc`, and `aes128-cbc`. The commands `ssh` and `scp` have a `-c` option for specifying the encryption algorithm on the command line.

Known Host Configuration

The known host files (`/etc/ssh/ssh_known_hosts` and `~/.ssh/known_hosts`) contain the public keys for all hosts with which the client can communicate using the secure shell. The `GlobalKnownHostsFile` keyword specifies an alternate file instead of `/etc/ssh/ssh_known_hosts`. `UserKnownHostsFile` specifies an alternate to `~/.ssh/known_hosts`.

`StrictHostKeyChecking` requires new hosts to be added manually to the known hosts file, and refuses any host whose public key has changed or whose public key is not in the known hosts file. The keyword `CheckHostIP` enables the IP address for hosts in the known host files to be checked, in case a key has been changed due to DNS spoofing.

Client-Side X11 and Port Forwarding

The `LocalForward` keyword specifies a local TCP/IP port to be forwarded over a secure channel to a specified port on a remote host. `GatewayPorts` enables remote hosts to connect to local forwarded ports.

The command `ssh` enables port forwarding through these options:

- `-L`, which specifies the local port to be forwarded to the specified port on the remote host
- `-R`, which specifies a remote port to be forwarded to the local host and specified port

`ForwardX11` redirects X11 connections to the remote host with the `DISPLAY` environment variable set. `XAuthLocation` specifies the location of the `xauth(1)` program.

Client-Side Connection and Other Parameters

The `NumberOfPasswordPrompts` keyword specifies how many times the user is prompted for a password before the secure shell quits. `ConnectionAttempts` specifies how many tries (at one per second) are made before the secure shell either quits or falls back to `rsh` if `FallBackToRsh` is set.

`Compression` enables compression of transmitted data. `CompressionLevel` sets a level of 1 to 9, trading off between speed and amount of compression.

`User` specifies an alternate user name. `Hostname` specifies an alternate name for a remote host. `ProxyCommand` specifies an alternate command name for starting the secure shell. Any command that can connect to your proxy server can be used. The command should read from its standard input and write to its standard output.

`Batchmode` disables password prompts, which is useful for scripts and other batch jobs.

`KeepAlive` enables messages to indicate network problems due to host crashes. `LogLevel` sets the verbosity level for `ssh` messages.

`EscapeChar` defines a single character used as a prefix for displaying special characters as plain text.

Secure Shell Server Configuration

The server-side characteristics of a secure shell session are governed by the `/etc/ssh/sshd_config` file, which is set up by the administrator.

Server-Side Authentication

Permitted authentication methods are indicated by these keywords: `DSAAuthentication`, `PasswordAuthentication`, `RhostsAuthentication`, `RhostsRSAAuthentication`, and `RSAAuthentication`.

`HostKey` and `HostDSAKey` identify files holding host public keys when the default file name is not used. `KeyRegenerationInterval` defines how often the server key is regenerated.

`Protocol` specifies version. `Ciphers` specifies encryption algorithms for v2. `ServerKeyBits` defines number of bits in the server's key.

Ports and Forwarding

`AllowTCPForwarding` specifies whether TCP forwarding is permitted.

`GatewayPorts` allows remote hosts to connect to ports forwarded for the client. `Port` specifies the port number that `sshd` listens on. `ListenAddress` designates a specific local address that `sshd` listens to. If there is no `ListenAddress` specification, `sshd` listens to all addresses by default.

`X11Forwarding` allows X11 forwarding. `X11DisplayOffset` specifies the first display number available for forwarding. This prevents `sshd` from interfering with real X11 servers. `XAuthLocation` specifies the location of the `xauth` program.

Session Controls

`KeepAlive` displays messages regarding broken connections and host crashes. `LogLevel` sets the verbosity level of messages from `sshd`. `SyslogFacility` provides a facility code for messages logged from `sshd`.

Server Connection and Other Parameters

`AllowGroups`, `AllowUsers`, `DenyGroups`, and `DenyUsers` control which users can or cannot use `ssh`.

`LoginGraceTime`, `MaxStartups`, `PermitRootLogin`, and `PermitEmptyPasswords` set controls on users logging in. `StrictModes` causes `sshd` to check file modes and ownership of the user's files and home directory before login. `UseLogin` specifies whether `login` is used for interactive login sessions. Turning this option on should not be necessary and is not recommended for the Solaris environment.

`Subsystem` configures a file transfer daemon for using `sftp`.

Maintaining Known Hosts on a Site-Wide Basis

Each host that needs to talk to another host securely must have its public key stored in the server host's `/etc/ssh/ssh_known_hosts` file. Although it is most convenient to update the `/etc/ssh/ssh_known_hosts` files by a script, this is heavily discouraged because it opens a major security vulnerability.

Secure Shell Files

The important files used in the secure shell are shown in the following table with the suggested UNIX permissions.

TABLE 5-1 Secure Shell Files

File Name	Description	Suggested Permissions and Owner
<code>/etc/ssh/sshd_config</code>	Contains configuration data for <code>sshd</code> , the secure shell daemon.	<code>-rw-r--r-- root</code>
<code>/etc/ssh/ssh_host_key</code>	Contains host private key.	<code>-rw----- root</code>
<code>/etc/ssh_host_key.pub</code>	Contains host public key. Used to copy host key to local <code>known_hosts</code> file.	<code>-rw-r--r-- root</code>
<code>/var/run/sshd.pid</code>	Contains the process ID of the secure shell daemon (<code>sshd</code>) listening for connections (if multiple daemons, contains the last one started).	<code>rw-r--r-- root</code>
<code>\$HOME/.ssh/authorized_keys</code>	Lists the RSA keys that can be used with v1 to log into the user's account or the DSA and RSA keys that can be used with v2.	<code>-rw-rw-r-- johndoe</code>
<code>/etc/ssh/ssh_known_hosts</code> , <code>\$HOME/.ssh/known_hosts</code>	Contains host public keys for all hosts with which the client may communicate securely. The global files should be prepared by the administrator (optional). The per-user file is maintained automatically. Whenever the user connects with an unknown host, the remote host key is added to the per-user file.	<code>/etc/ssh/ssh_known_hosts</code> <code>-rw-r--r-- root</code> <code>\$HOME/.ssh/known_hosts</code> <code>-rw-r--r-- johndoe</code>
<code>/etc/nologin</code>	If this file exists, <code>sshd</code> refuses to let anyone except root log in. The contents are displayed to users attempting to log in.	<code>-rw-r--r-- root</code>
<code>\$HOME/.rhosts</code>	Contains host-user name pairs specifying hosts to which the user can log in without a password. The file is also used by <code>rlogind</code> and <code>rshd</code> .	<code>-rw-r--r-- johndoe</code>
<code>\$HOME/.shosts</code>	Contains host-user name pairs specifying hosts to which the user can log in with no password using the secure shell.	<code>-rw-r--r-- johndoe</code>

TABLE 5-1 Secure Shell Files (Continued)

File Name	Description	Suggested Permissions and Owner
/etc/hosts.equiv	Contains hosts used in .rhosts authentication.	-rw-r--r-- root
/etc/ssh/shosts.equiv	Contains hosts used in .rhosts or secure shell authentication.	-rw-r--r-- root
\$HOME/.ssh/environment	Used for initialization to make assignments at login.	-rw---- johndoe
\$HOME/.ssh/rc	Runs initialization routines before user's home directory becomes available at login.	-rw---- johndoe
/etc/ssh/sshrc	Runs host-specific initialization routines specified by administrator for a user's home directory.	-rw-r--r-- root

TABLE 5-2 Secure Shell Commands

Command	Description
ssh(1)	A program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.
sshd(1m)	The daemon program for secure shell. It listens for connections from clients and provides secure encrypted communications between two untrusted hosts over an insecure network.
ssh-keygen(1)	Generates and manages authentication keys for ssh.
ssh-agent(1)	A program to hold private keys used for public key authentication. ssh-agent is started at the beginning of an X-session or a login session, and all other windows or programs are started as clients to the ssh-agent program. Through the use of environment variables, the agent can be located and automatically used for authentication when logging into other machines using ssh.
ssh-add(1)	Adds RSA or DSA identities (keys) to the authentication agent, ssh-agent.
scp(1)	Copies files between hosts on a network securely, using ssh for data transfer. Unlike rcp, scp asks for passwords or passphrases (if they are needed for authentication).
sftp(1)	An interactive file transfer program, similar to ftp, that performs all operations over an encrypted ssh transport. sftp connects and logs into the specified host name and then enters an interactive command mode.

Introduction to SEAM

This chapter provides an introduction to the SEAM product.

- “What Is SEAM?” on page 73
- “How SEAM Works” on page 74
- “Security Services” on page 80
- “SEAM Releases” on page 80

What Is SEAM?

SEAM (Sun Enterprise Authentication Mechanism) is a client/server architecture that offers strong user authentication, as well as data integrity and privacy, for providing secure transactions over networks. *Authentication* guarantees that the identities of both the sender and recipient of a network transaction are true; SEAM can also verify the validity of data being passed back and forth (*integrity*) and encrypt it during transmission (*privacy*). Using SEAM, you can log on to other machines, execute commands, exchange data, and transfer files securely. Additionally, SEAM provides *authorization* services, allowing administrators to restrict access to services and machines; moreover, as a SEAM user you can regulate other people’s access to your account.

SEAM is a *single-sign-on* system, meaning that you only need to authenticate yourself to SEAM once per session, and all subsequent transactions during the session are automatically secured. After SEAM has authenticated you, you do not need to authenticate yourself every time you use a SEAM-based command such as `kinit` or `klist`, or access data on an NFS file system. This means you do not have to send your password over the network, where it can be intercepted, each time you use these services.

SEAM is based on the Kerberos V5 network authentication protocol developed at the Massachusetts Institute of Technology (MIT). People who have used Kerberos V5 should therefore find SEAM very familiar. Since Kerberos V5 is a *de facto* industry standard for network security, SEAM promotes interoperability with other systems. In other words, because SEAM works with systems using Kerberos V5, it allows for secure transactions even over heterogeneous networks. Moreover, SEAM provides authentication and security both between domains and within a single domain.

Note – Because SEAM is based on, and designed to interoperate with, Kerberos V5, this manual often uses the terms “Kerberos” and “SEAM” more or less interchangeably — for example, “Kerberos realm” or “SEAM-based utility.” (Moreover, “Kerberos” and “Kerberos V5” are used interchangeably, as well.) The manual draws distinctions when necessary.

SEAM allows for flexibility in running Solaris applications. You can configure SEAM to allow both SEAM-based and non-SEAM-based requests for network services such as the NFS service. That means current Solaris applications still work even if they are running on systems on which SEAM is not installed. Of course, you can also configure SEAM to allow only SEAM-based network requests.

Additionally, applications do not have to remain committed to SEAM if other security mechanisms are developed. Because SEAM is designed to integrate modularly into the Generic Security Service API, applications that make use of the GSS-API can utilize whichever security mechanism best suits their needs.

How SEAM Works

The following is a generalized overview of the SEAM authentication system. For a more detailed description, see “How the Authentication System Works” on page 196.

From the user’s standpoint, SEAM is mostly invisible after the SEAM session has been started. Initializing a SEAM session is often no more than logging in and providing a Kerberos password.

The SEAM system revolves around the concept of a *ticket*. A ticket is a set of electronic information that serves as identification for a user or a service such as the NFS service. Just as your driver’s license identifies you and indicates what driving permissions you have, so a ticket identifies you and your network access privileges. When you perform a SEAM-based transaction — for example, if you `rlogin` in to another machine — you transparently send a request for a ticket to a *Key Distribution Center*, or KDC, which accesses a database to authenticate your identity. The KDC returns a ticket

granting you permission to access the other machine. “Transparently” means that you do not need to explicitly request a ticket.

Tickets have certain attributes associated with them. For example, a ticket can be *forwardable* (meaning that it can be used on another machine without a new authentication process), or *postdated* (not valid until a specified time). How tickets are used — for example, which users are allowed to obtain which types of ticket — is set by *policies* determined when SEAM is installed or administered.

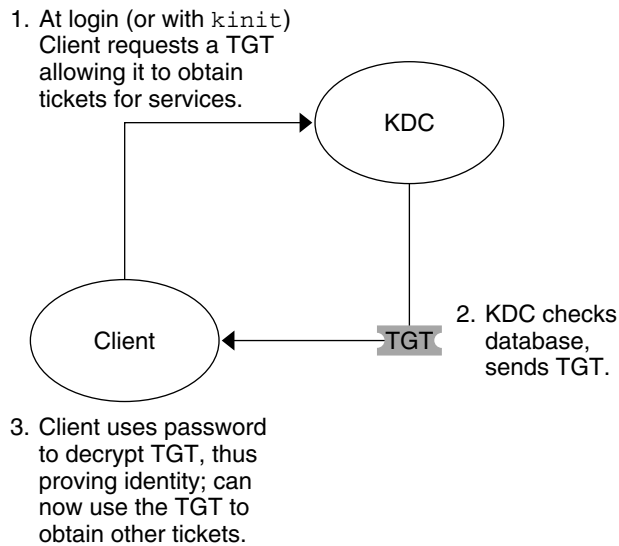
Note – You will frequently see the terms *credential* and *ticket*. In the greater Kerberos world, they are often used interchangeably. Technically, however, a credential is a ticket plus the *session key* for that session. This difference is explained in more detail in “Gaining Access to a Service Using SEAM” on page 196.

The following sections briefly explain the SEAM authentication process.

Initial Authentication: the Ticket-Granting Ticket

Kerberos authentication has two phases: an initial authentication that allows for all subsequent authentications, and the subsequent authentications themselves.

Figure 6–1 shows how the initial authentication takes place:



TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 6-1 Initial Authentication for SEAM Session

1. A client (a user, or a service such as NFS) begins a SEAM session by requesting a *ticket-granting ticket* (TGT) from the Key Distribution Center. This is often done automatically at login.

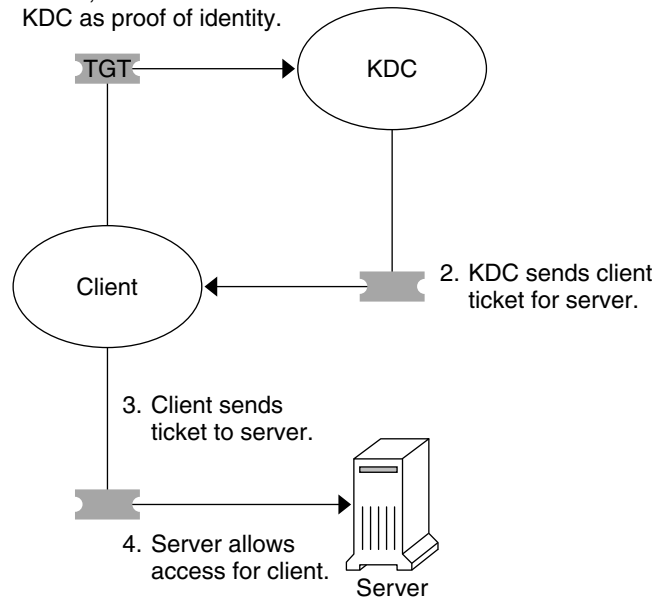
A ticket-granting ticket is needed to obtain other tickets for specific services. One analogy is to think of the ticket-granting ticket as similar to a passport. Like a passport, the ticket-granting ticket identifies you and allows you to obtain numerous “visas” — where the “visas” (tickets) are not for foreign countries but for remote machines or network services. Like passports and visas, the ticket-granting ticket and the other various tickets have limited lifetimes. The difference is that “Kerberized” commands notice that you have a passport and obtain the visas for you — you don’t have to perform the transactions yourself.

2. The KDC creates a ticket-granting ticket and sends it back, in encrypted form, to the client. The client decrypts the ticket-granting ticket using the client’s password.
3. Now in possession of a valid ticket-granting ticket, the client can request tickets for all sorts of network operations for as long as the ticket-granting ticket lasts. This is usually a few hours. Each time the client performs a unique network operation, it requests a ticket for that operation from the KDC.

Subsequent Authentications

After the client has received the initial authentication, each individual authentication follows the pattern shown in Figure 6-2:

1. Client requests ticket for server; sends TGT to KDC as proof of identity.



TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 6-2 Obtaining Access to a Service

1. The client requests a ticket for a particular service from the KDC, sending the KDC its ticket-granting ticket as proof of identity.
2. The KDC sends the ticket for the specific service to the client.

For example, suppose user `joe` wants to access a file system using NFS that has been shared with `krb5` authentication required. Since he is already authenticated (that is, he already has a ticket-granting ticket), as he attempts to access the files, the NFS client system will automatically and transparently obtain a ticket from the KDC for the NFS service.

3. The client sends the ticket to the server.

When using the NFS service, the NFS client will automatically and transparently send the ticket for the NFS service to the NFS server.

4. The server allows the client access.

Looking at these steps, you might have noticed that the server doesn't appear to ever communicate with the KDC. It does, though; it registers itself with the KDC, just as the first client does. For simplicity's sake we have left that part out.

Principals

A client in SEAM is identified by its *principal*. A principal is a unique identity to which the KDC can assign tickets. A principal can be a user, such as `joe`, or a service, such as `nfs`.

By convention, a principal name is divided into three parts: the *primary*, the *instance*, and the *realm*. A typical SEAM principal would be, for example, `joe/admin@ENG.EXAMPLE.COM`, where:

- `joe` is the primary. This can be a username, as shown here, or a service, such as `nfs`. It can also be the word `host`, signifying that this is a service principal set up to provide various network services.
- `admin` is the instance. An instance is optional in the case of user principals, but it is required for service principals. For example: if the user `joe` sometimes acts as a system administrator, he can use `joe/admin` to distinguish himself from his usual user identity. Likewise, if `joe` has accounts on two different hosts, he can use two principal names with different instances (for example, `joe/denver.example.com` and `joe/boston.example.com`). Notice that SEAM treats `joe` and `joe/admin` as two completely different principals.

In the case of a service principal, the instance is the fully qualified hostname. `bigmachine.eng.example.com` is an example of such an instance, so that the primary/instance might be, for example, `host/bigmachine.eng.example.com`.

- `ENG.EXAMPLE.COM` is the SEAM realm. Realms are discussed in "Realms" on page 78.

The following are all valid principal names:

- `joe`
- `joe/admin`
- `joe/admin@ENG.EXAMPLE.COM`
- `nfs/host.eng.example.com@ENG.EXAMPLE.COM`
- `host/eng.example.com@ENG.EXAMPLE.COM`

Realms

A realm is a logical network, like a domain, which defines a group of systems under the same *master KDC* (see below). Figure 6-3 shows how realms can relate to one another. Some realms are hierarchical (one being a superset of the other). Otherwise

the realms are non-hierarchical and the mapping between the two realms must be defined. A feature of SEAM is that it permits authentication across realms; each realm only needs to have a principal entry for the other realm in its KDC.

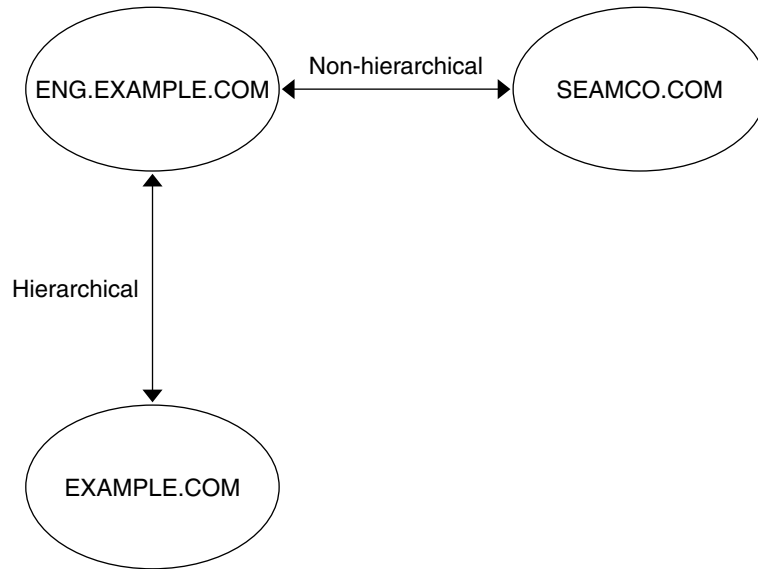


FIGURE 6-3 Realms

Realms and Servers

Each realm must include a server that maintains the master copy of the principal database. This is called the *master KDC server*. Additionally, each realm should contain at least one *slave KDC server*, which contains duplicate copies of the principal database. Both the master and the slave KDC servers create tickets used to establish authentication.

Realms can also include *NFS servers*, which provide NFS services, using Kerberos authentication. If you have installed SEAM 1.0 or 1.0.1, the realm may include a SEAM network *application server*, which provides access to Kerberized applications (such as `ftp`, `telnet`, and `rsh`).

Figure 6-4 shows what a hypothetical realm might contain.

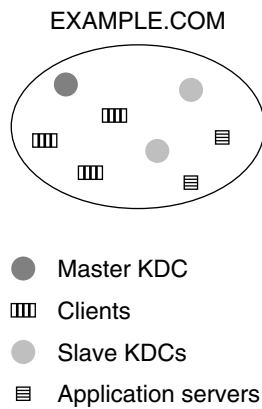


FIGURE 6-4 A Typical Realm

Security Services

In addition to providing secure authentication of users, SEAM provides two security services:

- *Integrity.* Just as authentication ensures that clients on a network are who they claim to be, integrity ensures that the data they send is valid and has not been tampered with during transit. This is done through cryptographic checksumming of the data. Integrity also includes user authentication.
- *Privacy.* Privacy takes security a step further. It not only includes verifying the integrity of transmitted data, but it encrypts the data before transmission, protecting it from eavesdroppers. It authenticates users, as well.

Note – Because of U.S. export restrictions, the privacy service might not be available to all SEAM users.

SEAM Releases

Parts of this product have been included in four releases. The table below describes which components are included in each release. The specific components of all of the releases are described in the following sections.

TABLE 6-1 SEAM Release Contents

Release Name	Contents
SEAM 1.0 in Solaris Easy Access Server (SEAS) 3.0	Full release of SEAM for the Solaris 2.6 and 7 releases
SEAM in Solaris 8 release	SEAM client software only
SEAM 1.0.1 in the Solaris 8 Admin Pack	Full release of SEAM for the Solaris 8 release
SEAM in Solaris 9 release	Full release of SEAM without the remote applications

SEAM 1.0 Components

Like the MIT distribution of Kerberos V5, SEAM includes:

- Key Distribution Center (KDC) (master):
 - Kerberos database administration daemon — `kadmin`
 - Kerberos ticket processing daemon — `krb5kdc`
- Slave KDCs
- Database administration programs — `kadmin` and `kadmin.local`
- Database propagation software — `kprop`
- User programs for obtaining, viewing, and destroying tickets — `kinit`, `klist`, `kdestroy` — and for changing your SEAM password — `kpasswd`
- Applications — `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet` — and daemons for these applications — `ftpd`, `rlogind`, `rshd` and `telnetd`
- Administration utilities — `ktutil`, `kdb5_util`
- Several libraries

In addition, the SEAM product includes the following:

- SEAM Administration Tool (`gkadmin`) — Allows you to administer the KDC. This Java based GUI allows an administrator to perform the tasks usually performed through the `kadmin` command.
- The Pluggable Authentication Module (PAM) — Allows applications to use various authentication mechanisms; PAM can be used to make login and logouts transparent to the user.
- A utility (`gsscred`) and a daemon (`gssd`) — These programs help map UNIX UIDs to principal names; needed because SEAM NFS servers use UNIX IDs to identify users and not principal names, which are stored in a different format altogether.

- GSS_API framework — The Generic Security Service Application Programming Interface (GSS-API) allows applications to use multiple security mechanisms without having to recompile the application every time a new mechanism is added. Because GSS-API is machine-independent, it is appropriate for applications on the Internet. GSS-API provides applications with the ability to include the integrity and privacy security services, as well as authentication.
- The RPCSEC_GSS Application Programming Interface (API) — Allows NFS services to use Kerberos authentication. RPCSEC_GSS is a new security flavor that provides security services that are independent of the mechanisms being used; RPCSEC_GSS sits “on top” of the GSS-API layer. Any pluggable GSS_API-based security mechanism can be used by applications using RPCSEC_GSS.
- A preconfiguration procedure — Allows you to set the parameters for installing and configuring SEAM, making SEAM installation automatic; especially useful for multiple installations.
- Kernel modifications — Allows for faster performance.

SEAM Components in the Solaris 9 Release

The Solaris 9 release includes all of the parts of the SEAM 1.0 release except for the applications and the preconfiguration procedure.

SEAM Components in the Solaris 8 Release

The Solaris 8 release included only the client-side portions of SEAM, so many of these components are not included. This enables systems running the Solaris 8 release to become SEAM clients without having to install SEAM separately. To use this functionality you must install a KDC using either SEAS 3.0 or Solaris 8 Admin Pack, the MIT distribution, or Windows2000. The client-side components are not useful without a configured KDC to distribute tickets. The following components were included in this release:

- User programs for obtaining, viewing, and destroying tickets — `kinit`, `klist`, `kdestroy` — and for changing your SEAM password — `kpasswd`
- Key table administration utility — `ktutil`
- Additions to the Pluggable Authentication Module (PAM) — Allows applications to use various authentication mechanisms; PAM can be used to make login and logouts transparent to the user.
- GSS_API plug-ins — Provides Kerberos protocol and cryptographic support
- NFS client and server support

SEAM 1.0.1 Components

The SEAM 1.0.1 release includes all of the portions of the SEAM 1.0 release that are not already included in the Solaris 8 release. This includes:

- Key Distribution Center (KDC) (master):
 - Kerberos database administration daemon — `kadmin`
 - Kerberos ticket processing daemon — `krb5kdc`
- Slave KDCs
- Database administration programs — `kadmin` and `kadmin.local`
- Database propagation software — `kprop`
- Applications — `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet` — and daemons for these applications — `ftpd`, `rlogind`, `rshd` and `telnetd`
- Administration utility — `kdb5_util`
- SEAM Administration Tool (`gkadmin`) — Allows you to administer the KDC. This Java-based GUI allows an administrator to perform the tasks usually performed through the `kadmin` command.
- A preconfiguration procedure — Allows you to set the parameters for installing and configuring SEAM, making SEAM installation automatic; especially useful for multiple installations.
- Several libraries

Planning for SEAM

This chapter should be studied by individuals involved in the installation and maintenance of SEAM. The chapter includes a discussion of several installation and configuration considerations that you must resolve before installing or configuring SEAM.

This is a list of the issues that should be resolved by a System Administrator or other knowledgeable support staff:

- “Realms” on page 85
- “Mapping Hostnames Onto Realms” on page 87
- “Client and Service Principal Names” on page 87
- “Slave KDCs” on page 88
- “Database Propagation” on page 88
- “Clock Synchronization” on page 88

SEAM Configuration Decisions

Before installing SEAM, you must resolve several configuration issues. Although changing the configuration after the initial install is not impossible, it becomes more difficult with each new client added to the system. In addition, some changes require a full re-installation, so it is better to consider long-term goals when planning.

Realms

A realm is logical network, like a domain, which defines a group of systems under the same master KDC. As with establishing a DNS domain name, issues such as the realm

name, the number and size of each realm, and the relationship of a realm to other realms should be resolved before configuring SEAM.

Realm Names

Realm names can be any ASCII string. Usually it is the same as your DNS domain name, in uppercase. This helps differentiate problems with SEAM from problems with the DNS namespace, while using a name that is familiar. If you do not use DNS or choose to use a different string, then you can use any string, although using realm names that follow the standard internet naming structure is wise.

Number of Realms

The number of realms that your installation requires depends on several factors:

- The number of clients to be supported. Too many clients in one realm makes administration more difficult and eventually requires splitting the realm. The primary factors that determine the number of clients that can be supported are: the amount of SEAM traffic that each client generates, the bandwidth of the physical network and the speed of the hosts. Since each installation will have different limitations, there is no rule for determining the maximum number of clients.
- How far apart the clients are. It might make sense to set up several small realms if the clients are in a different geographic region.
- The number of hosts that are available to be installed as KDCs. Each realm should have at least two KDC servers (master and slave).

Realm Hierarchy

When configuring multiple realms, you need to decide how to tie the realms together. You can establish a hierarchical relation between the realms that provides automatic paths to the related domains, but requires that all realms in the hierarchical chain are configured properly. The automatic paths can ease the administration burden; however, if there are many levels of domains, you might not want to use the default path because it requires too many transactions.

You can also choose to establish the connection directly. A direct connection is most useful when too many levels exist between two hierarchical domains or when there is no hierarchical relationship. The connection must be defined in `/etc/krb5/krb5.conf` on all hosts using the connection, so some additional work required. See “Realms” on page 78 for an introduction and “Configuring Cross-Realm Authentication” on page 100 for the configuration procedures for multiple realms.

Mapping Hostnames Onto Realms

Mapping hostnames onto realm names is defined in the `domain_realm` section of the `krb5.conf` file. These mappings can be defined for a whole domain and for individual hosts, depending on the requirements. See the `krb5.conf(4)` man page for more information.

Client and Service Principal Names

When using SEAM, it is strongly recommended that DNS services are already configured and running on all hosts. If DNS is used, it must be enabled on all systems or on none of them. If DNS is available, then the principal should contain the Fully Qualified Domain Name (FQDN) of each host. For example, if the host name is `boston`, the DNS domain name is `example.com`, and the realm name is `EXAMPLE.COM`, then the principal name for the host should be `host/boston.example.com@EXAMPLE.COM`. The examples in this book use the FQDN for each host.

For the principal names which include the FQDN of an host, it is important to match the string describing the DNS domain name in `/etc/resolv.conf`. SEAM requires that the DNS domain name be in lower case letters when entering the FQDN for a principal. The DNS domain name may include upper and lower case letters, but only use lower case letters when creating a host principal. In the example above, it doesn't matter if the DNS domain name is `example.com` or `Example.COM` or any other variations. The principal name for the host would still be `host/boston.example.com@EXAMPLE.COM`.

SEAM can run without DNS services, but some key functionality, like the ability to communicate to other realms, will not work. If DNS is not configured, then a simple host name can be used as the instance name. In this case the principal would be `host/boston@EXAMPLE.COM`. If DNS is enabled later, all host principals must be deleted and replaced in the KDC database.

Ports for the KDC and Admin Services

By default, port 88 and port 750 are used for the KDC and port 749 is used for the KDC administration daemon. Different port numbers can be used, but changing them requires that the `/etc/services` and `/etc/krb5/krb5.conf` files be changed on every client. In addition the `/etc/krb5/kdc.conf` file on each KDC must be updated.

Slave KDCs

Slave KDCs generate credentials for clients just like the master KDC. The slave KDCs provide backup in case the master is unavailable. Each realm should have at least one slave KDC. Additional slave KDCs might be required, depending on these factors:

- The number of physical segments in the realm. Normally, the network should be set up so that each segment can function, at least minimally, without the rest of the realm. To do this requires a KDC to be accessible from each segment. The KDC in this instance could be either a master or a slave.
- The number of clients in the realm. Adding more slave KDC servers can reduce the load on the current servers.

It is possible to add too many slave KDCs. Remember that the KDC database must be propagated to each server, so the more KDC servers installed, the longer it can take to get the data updated throughout the realm. Also, since each slave retains a copy of the KDC database, more slaves increase the risk of a security compromise.

In addition, one or more of the slave KDCs can be configured to be swapped easily with the master KDC. The advantage to following this procedure on at least one of the slave KDCs is that if the master KDC fails for any reason, you will have a system preconfigured that will be easy to swap as the master. See “Swapping Master and Slave KDCs” on page 112 for instructions on how to configure a swappable slave KDC.

Database Propagation

The database stored on the master KDC must be regularly propagated to the slave KDCs. One of the first issues to be resolved is how often to update the slave KDCs. The desire to have up-to-date information available to all of the clients needs to be weighed against the amount of time it takes to complete the update. See “Administering the Kerberos Database” on page 116 for more information about database propagation.

In large installations, with many KDCs in one realm, it is possible for one or more of the slaves to propagate the data so that the process is done in parallel. This reduces the amount of time that the update takes, but it also increases the level of complexity in administering the realm.

Clock Synchronization

All hosts participating in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as *clock*

skew), which provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, requests are rejected.

One way to synchronize all of the clocks is to use the Network Time Protocol (NTP) software (see “Synchronizing Clocks between KDCs and SEAM Clients” on page 110 for more information). Other ways of synchronizing the clocks are available, so using NTP is not required. Some form of synchronization should be used to prevent access failures due to clock skew.

Configuring SEAM

This chapter provides configuration procedures for KDC servers, network application servers, NFS servers, and SEAM clients. Many of these procedures require root access, so they should be used by System Administrators or advanced users. Cross-realm configuration procedures and other topics related to the KDC servers are also covered.

- “Configuring KDC Servers” on page 92
- “Configuring Cross-Realm Authentication” on page 100
- “Configuring SEAM NFS Servers” on page 102
- “Configuring SEAM Clients” on page 107
- “Synchronizing Clocks between KDCs and SEAM Clients” on page 110
- “Swapping Master and Slave KDCs” on page 112
- “Administering the Kerberos Database” on page 116
- “Increasing Security” on page 123

SEAM Configuration Task Map

Parts of the configuration process depend on other parts and must be done in a specific order. These procedures often establish services that are required to use SEAM. Other procedures are not dependent, and can be done when appropriate. The table below shows a suggested order for a SEAM installation.

TABLE 8-1 First Steps: SEAM Configuration Order

Task	Description	For Instructions, Go To ...
1. Plan for your SEAM Installation	Consider configuration issues and make decisions about them before starting the software configuration process.	“SEAM Configuration Decisions” on page 85

TABLE 8-1 First Steps: SEAM Configuration Order (Continued)

Task	Description	For Instructions, Go To ...
2. (Optional) Install NTP	In order for SEAM to work properly, the clocks on all systems in the realm must be kept in sync.	"Synchronizing Clocks between KDCs and SEAM Clients" on page 110
3. Configure the master KDC server	Steps to configure and build the master KDC server and database for a realm.	"How to Configure a Master KDC" on page 93
4. (Optional) Configure a slave KDC server	Steps to configure and build a slave KDC server for a realm.	"How to Configure a Slave KDC" on page 97
5. (Optional) Increase security on the KDC servers	Steps to prevent security breaches on the KDC servers.	"How to Restrict Access for KDC servers" on page 123
6. (Optional) Configure swappable KDC servers	Follow the steps in this procedure to make the task of swapping the master and a slave KDC easier.	"How to Configure a Swappable Slave KDC" on page 112

Once the required steps have been completed, the following procedures may be used when required.

TABLE 8-2 Next Steps: Additional SEAM Tasks

Task	Description	For Instructions, Go To ...
Configure cross-realm authentication	Steps to enable communications from one realm to another.	"Configuring Cross-Realm Authentication" on page 100
Configure SEAM clients	Steps to enable a client to use SEAM services.	"Configuring SEAM Clients" on page 107
Configure SEAM NFS server	Steps to enable a server to share a file system requiring Kerberos authentication.	"Configuring SEAM NFS Servers" on page 102

Configuring KDC Servers

After installing the SEAM software, you must configure the KDC servers. Configuring a master KDC and at least one slave KDC provides the service that issues credentials. These credentials are the basis for SEAM, so the KDCs must be installed before attempting other tasks.

The most significant difference between a master and a slave KDC is that only the master can handle database administration requests. For instance, changing a password or adding a new principal must be done on the master KDC. These changes

can then be propagated to the slave KDCs. Both the slave and master KDCs generate credentials; this provides redundancy in case the master KDC is not able to respond.

▼ How to Configure a Master KDC

In this procedure the following configuration parameters are used:

- realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- master KDC = `kdc1.example.com`
- slave KDC = `kdc2.example.com`
- admin principle = `kws/admin`
- online help URL =
`http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in the *SEAM Installation and Release Notes*.

1. Prerequisites for configuring a master KDC.

This procedure requires that the master KDC software is installed. In addition, DNS must be running. See “Swapping Master and Slave KDCs” on page 112 for specific naming instructions if this master is to be swappable.

2. Become superuser on the master KDC.

3. Edit the Kerberos configuration file (`krb5.conf`).

You need to change the realm names and the names of the servers. See the `krb5.conf(4)` man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM

[realms]
    EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc1.example.com
    }

[domain_realm]
    .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
```

```
#
[logging]
    default = FILE:/var/krb5/kdc.log
    kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
    }
}
```

In this example, the lines for `domain_realm`, `kdc`, `admin_server`, and all `domain_realm` entries were changed. In addition, the line defining the `help_url` was edited.

4. Edit the KDC configuration file (`kdc.conf`).

You need to change the realm name. See the `kdc.conf(4)` man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
    kdc_ports = 88,750

[realms]
    EXAMPLE.COM= {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        admin_keytab = /etc/krb5/kadm5.keytab
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
    }
}
```

In this example, the realm name definition in the `realms` section was changed.

5. Create the KDC database using `kdb5_util`.

The `kdb5_util` command creates the KDC database and also, when used with the `-s` option, creates a stash file that is used to authenticate the KDC to itself before the `kadmind` and `krb5kdc` daemons are started.

```
kdc1 # /usr/sbin/kdb5_util create -r EXAMPLE.COM -s
Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM'
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: <type the key>
Re-enter KDC database master key to verify: <type it again>
```

The `-r` option followed by the realm name is not required if the realm name is equivalent to the servers name space domain name.

6. Edit the Kerberos access control list file (`kadm5.ac1`).

Once populated, `/etc/krb5/kadm5.ac1` should contain all of the principal names that are allowed to administer the KDC. The first entry added might look like the following:

```
kws/admin@EXAMPLE.COM *
```

This entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an `"*"` to match all admin principals. This could be a security risk, so it is more secure to include a list of all of the admin principals.

7. Start `kadmin.local`.

The next sub-steps create principals used by SEAM.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

a. Add administration principals to the database using `kadmin.local`.

You can add as many admin principals as you need. You must add at least one admin principal to complete the KDC configuration process. For this example, a `kws/admin` principal is added. You can substitute an appropriate principal name instead of `"kws."`

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM: <type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM: <type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

b. Create a keytab file for `kadmin` using `kadmin.local`.

This command sequence creates a special keytab file with principal entries for `kadmin` and `changepw`. These principals are needed for the `kadmin` service.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES-CBC-CRC
added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES-CBC-CRC
added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

c. Quit `kadmin.local`

You have added all of the required principals for the next steps.

```
kadmin.local: quit
```

8. Start the Kerberos daemons.

```
kdc1 # /etc/init.d/kdc start
kdc1 # /etc/init.d/kdc.master start
```

9. Start `kadmin`.

At this point, you can add principals using the SEAM Administration Tool. The command line example is shown for simplicity. You must log on with one of the admin principal names that you created earlier in this procedure.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Enter kws/admin password>
kadmin:
```

a. Create the master KDC host principal using `kadmin`.

The host principal is used by Kerberized applications (such as `klist` and `kprop`).

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

b. Optional: Create the master KDC root principal using `kadmin`.

This principal is used for authenticated NFS-mounting, and so might not be necessary on a master KDC.

```
kadmin: addprinc root/kdc1.example.com
Enter password for principal root/kdc1.example.com@EXAMPLE.COM:      <type the password>
Re-enter password for principal root/kdc1.example.com@EXAMPLE.COM:    <type it again>
Principal "root/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

c. Add the master KDCs host principal to the master KDCs keytab file.

Adding the host principal to the keytab file allows for this principal to be used automatically.

```
kadmin: ktadd host/kdc1.example.com
kadmin: Entry for principal host/kdc1.example.com with
kvno 3, encryption type DES-CBC-CRC added to keytab
WRFILE:/etc/krb5/krb5.keytab
kadmin: quit
```

d. Quit `kadmin`

```
kadmin: quit
```

10. Add an entry for each KDC into the propagation configuration file (`kpropd.acl`).

See the `kprop` (1M) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
```

11. Optional: Synchronize the master KDCs clock using NTP or another clock synchronization mechanism.

It is not necessary to install and use NTP, but every clock must be within default time defined in the `libdefaults` section of the `krb5.conf` file in order for authentication

to succeed. See “Synchronizing Clocks between KDCs and SEAM Clients” on page 110 for information about NTP.

▼ How to Configure a Slave KDC

In this procedure, a new slave KDC named `kdc3` is configured. This procedure uses the following configuration parameters:

- realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- master kdc = `kdc1.example.com`
- slave kdc = `kdc2.example.com` and `kdc3.example.com`
- admin principle = `kws/admin`
- online help URL =
`http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in the *SEAM Installation and Release Notes*.

1. Prerequisites for configuring a slave KDC.

This procedure requires that the master KDC has been configured and that the SEAM slave KDC software has been installed on `kdc3`. See “Swapping Master and Slave KDCs” on page 112 for specific instructions if this slave is to be swappable.

2. On the master KDC: Become superuser.

3. On the master KDC: Start `kadmin`.

You must log on with one of the admin principal names that you created when configuring the master KDC.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Enter kws/admin password>
kadmin:
```

a. On the master KDC: Add slave host principals to the database, if not already done, using `kadmin`.

In order for the slave to function, it must have a host principal.

```
kadmin: addprinc -randkey host/kdc3.example.com
Principal "host/kdc3@EXAMPLE.COM" created.
kadmin:
```

- b. Optional: On the master KDC, create the slave KDC `root` principal using `kadmin`.**

This principal is only needed if the slave will be NFS-mounting an authenticated file system.

```
kadmin: addprinc root/kdc3.example.com
Enter password for principal root/kdc3.example.com@EXAMPLE.COM: <type the password>
Re-enter password for principal root/kdc3.example.com@EXAMPLE.COM: <type it again>
Principal "root/kdc3.example.com@EXAMPLE.COM" created.
kadmin:
```

- c. Quit `kadmin`**

```
kadmin: quit
```

- 4. On the master KDC: Edit the Kerberos configuration file (`krb5.conf`).**

You need to add an entry for each slave. See the `krb5.conf(4)` man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM

[realms]
    EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        kdc = kdc3.example.com
        admin_server = kdc1.example.com
    }

[domain_realm]
    .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
    default = FILE:/var/krb5/kdc.log
    kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

- 5. On the master KDC: Add an entry for each slave KDC into the database propagation configuration file (`kpropd.acl`).**

See the `kprop(1M)` man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
```

6. On all Slaves: Copy the KDC administration files from the master KDC server.

This step needs to be followed on all slave KDCs, since the master KDC server has updated information that each KDC server needs. You can use `ftp` or a similar transfer mechanism to grab copies of the following files from the master:

- `/etc/krb5/krb5.conf`
- `/etc/krb5/kdc.conf`
- `/etc/krb5/kpropd.acl`

7. On the new slave: Add the slave's host principal to the slave's keytab file using `kadmin`.

You must log on with one of the admin principal names that you created when configuring the master KDC. This entry will allow `kprop` and other Kerberized applications to function.

```
kdc3 # /usr/sbin/kadmin -p kws/admin
Enter password: <Enter kws/admin password>
kadmin: ktadd host/kdc3.example.com
kadmin: Entry for principal host/kdc3.example.com with
kvno 3, encryption type DES-CBC-CRC added to keytab
WRFILE:/etc/krb5/krb5.keytab
kadmin: quit
```

8. On the master KDC: Add slave KDC names to the `cron` job, which automatically runs the backups, by running `crontab -e`.

Add the name of each slave KDC server at the end of the `kprop_script` line.

```
10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com kdc3.example.com
```

You might also want to change the time of the backups. This configuration starts the backup process every day at 3:10 AM.

9. On the master KDC: Back up and propagate the database using `kprop_script`.

If a backup copy of the database is already available, it is not necessary to complete another backup. See "How to Manually Propagate the Kerberos Database to the Slave KDCs" on page 120 for further instructions.

```
kdc1 # /usr/lib/krb5/kprop_script kdc3.example.com
Database propagation to kdc3.example.com: SUCCEEDED
```

10. On the new slave: Create a stash file using `kdb5_util`.

```
kdc3 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key
```

```
Enter KDC database master key: <type the key>
```

11. On the new slave: Start the KDC daemon (`krb5kdc`).

```
kdc3 # /etc/init.d/kdc start
```

12. Optional: On the new slave, synchronize the master KDCs clock using NTP or another clock synchronization mechanism.

It is not necessary to install and use NTP, but every clock must be within the default time defined in the `libdefaults` section of the `krb5.conf` file in order for authentication to succeed. See “Synchronizing Clocks between KDCs and SEAM Clients” on page 110 for information about NTP.

Configuring Cross-Realm Authentication

You have several ways of linking realms together so that users in one realm can be authenticated in another. Normally this is accomplished by establishing a secret key to be shared between the two realms. The relationship of the realms can be either hierarchal or directional (see “Realm Hierarchy” on page 86).

▼ How to Establish Hierarchical Cross-Realm Authentication

For this example, we will use two realms, `ENG.EAST.EXAMPLE.COM` and `EAST.EXAMPLE.COM`. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

1. Prerequisites for establishing hierarchical cross-realm authentication.

This procedure requires that the master KDC for each realm has been configured. To fully test the process, several clients or slave KDCs must be installed.

2. Become `root` on the first master KDC.

3. Create ticket-granting ticket service principals for the two realms using `kadmin`.

You must log on with one of the `admin` principal names that was created when configuring the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password: <Enter kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
Enter password for principal krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM: <type the password>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM: <type the password>
kadmin: quit
```

Note – The password entered for each service principal must be identical in both KDCs; which means that the password for `krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM` must be the same in both realms.

4. Add entries to the Kerberos configuration file to define domain names for every realm (`krb5.conf`).

```
# cat /etc/krb5/krb5.conf
[libdefaults]
.
.
[domain_realm]
    .eng.east.example.com = ENG.EAST.EXAMPLE.COM
    .east.example.com = EAST.EXAMPLE.COM
```

In this example, domain names for the `ENG.EAST.EXAMPLE.COM` and `EAST.EXAMPLE.COM` realms are defined. It is important to include the subdomain first, since the file is searched top down.

5. Copy the Kerberos configuration file to all clients in this realm.

In order for the cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (`/etc/krb5/krb5.conf`) installed.

6. Repeat these steps in the second realm.

▼ How to Establish Direct Cross-Realm Authentication

This example uses two realms: `ENG.EAST.EXAMPLE.COM` and `SALES.WEST.EXAMPLE.COM`. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

1. Prerequisites for establishing direct cross-realm authentication.

This procedure requires that the master KDC for each realm has been configured. To fully test the process, several clients or slave KDCs must be installed.

2. Become superuser on one of the master KDC servers.

3. Create ticket-granting ticket service principals for the two realms using `kadmin`.

You must log on with one of the admin principal names that was created when configuring the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password: <Enter kws/admin password>
```

```
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM: <type the password>
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal
krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM: <type the password>
kadmin: quit
```

Note – The password entered for each service principal must be identical in both KDCs; which means that the password for `krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM` must be the same in both realms.

4. Add entries in the Kerberos configuration file to define the direct path to the remote realm (`kdc.conf`).

This example is for the clients in the `ENG.EAST.EXAMPLE.COM` realm. You would swap the realm names to get the appropriate definitions in the `SALES.WEST.EXAMPLE.COM` realm.

```
# cat /etc/krb5/krb5.conf
[libdefaults]
.
.
.
[capaths]
    ENG.EAST.EXAMPLE.COM = {
        SALES.WEST.EXAMPLE.COM = .
    }

    SALES.WEST.EXAMPLE.COM = {
        ENG.EAST.EXAMPLE.COM = .
    }
```

5. Copy the Kerberos configuration file to all clients in the current realm.

In order for the cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (`krb5.conf`) installed.

6. Repeat these steps for the second realm.

Configuring SEAM NFS Servers

NFS services use UNIX UIDs to identify a user and cannot directly use principals. To translate the principal to a UID, a credential table that maps user principals to UNIX UIDs must be created. The procedures below focus on the tasks necessary to configure

a SEAM NFS server, to administer the credential table, and to initiate Kerberos security modes for NFS-mounted file systems. The following table describes the tasks covered in this section.

TABLE 8-3 Configuring SEAM NFS Server Task Map

Task	Description	For Instructions, Go To ...
Configure a SEAM NFS server	Steps to enable a server to share a file system requiring Kerberos authentication.	"How to Configure SEAM NFS Servers" on page 103
Create a credential table	Steps to generate a credential table.	"How to Create a Credential Table" on page 104
How to change the credential table that maps user principles to UNIX UIDs.	Steps to update information in the credential table.	"How to Add a Single Entry to the Credential Table" on page 105
Share a file system with Kerberos authentication	Steps to share a file system with security modes so that Kerberos authentication is required.	"How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 105

▼ How to Configure SEAM NFS Servers

This procedure requires that the master KDC has been configured. To fully test the process you need several clients. The following configuration parameters are used:

```
realm name = EXAMPLE.COM
DNS domain name = example.com
NFS server = denver.example.com
admin principle = kws/admin
```

1. Prerequisites for configuring a SEAM NFS server.

The SEAM client software must be installed.

2. Optional: Install NTP client or other clock synchronization mechanism.

See "Synchronizing Clocks between KDCs and SEAM Clients" on page 110 for information about NTP.

3. Start kadmin.

Using the SEAM Administration Tool to add a principal is explained in "How to Create a New Principal" on page 151. The example below shows how to add the required principals using the command line. You must log on with one of the admin principal names that you created when configuring the master KDC.

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password: <Enter kws/admin password>
kadmin:
```

a. Create the server's NFS service principal.

```
kadmin: addprinc -randkey nfs/denver.example.com  
Principal "nfs/denver.example.com" created.  
kadmin:
```

b. Optional: Create a root principal for the NFS server.

```
kadmin: addprinc root/denver.example.com  
Enter password for principal root/denver.example.com@EXAMPLE.COM: <type the password>  
Re-enter password for principal root/denver.example.com@EXAMPLE.COM: <type it again>  
Principal "root/denver.example.com@EXAMPLE.COM" created.  
kadmin:
```

c. Add the server's NFS service principal to the server's keytab.

```
kadmin: ktadd nfs/denver.example.com  
kadmin: Entry for principal nfs/denver.example.com with  
kvno 3, encryption type DES-CBC-CRC added to keytab  
WRFILE:/etc/krb5/krb5.keytab  
kadmin: quit
```

d. Quit kadmin

```
kadmin: quit
```

4. Create the gsscred table.

See "How to Create a Credential Table" on page 104 for more information.

5. Share the NFS file system using Kerberos security modes.

See "How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 105 for more information.

6. On each client: authenticate both the user and root principals.

See "Setting Up Root Authentication to Mount NFS File Systems" on page 110 for more information.

▼ How to Create a Credential Table

The gsscred credential table is used by an NFS server to map SEAM principals to a UID. In order for NFS clients to be able to mount file systems from an NFS server using Kerberos authentication, this table must be created or made available.

1. Edit /etc/gss/gsscred.conf and change the mechanism.

Change the mechanism to files.

2. Create the credential table using gsscred.

The command gathers information from all of the sources listed with the passwd entry in /etc/nsswitch.conf. You might need to temporarily remove the files

entry, if you do not want the local password entries included in the credential table. See the `gsscred(1M)` man page for more information.

```
# gsscred -m kerberos_v5 -a
```

▼ How to Add a Single Entry to the Credential Table

This procedure requires that the `gsscred` table has already been installed on the NFS server.

1. **Become superuser on a NFS server.**
2. **Add an entry to the table using `gsscred`.**

```
# gsscred -m mech [ -n name [ -u uid ] ] -a
```

<i>mech</i>	The security mechanism to be used.
<i>name</i>	The principal name for the user, as defined in the KDC.
<i>uid</i>	The UID for the user, as defined in the password database.
<i>-a</i>	Adds the UID to principal name mapping.

Example—Changing a Single Entry to the Credential Table

The following example adds an entry for the user named `sandy`, which is mapped to UID 3736. The UID is pulled from the password file if it is not included on the command line.

```
# gsscred -m kerberos_v5 -n sandy -u 3736 -a
```

▼ How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes

1. **Become superuser on the NFS server.**

Example—Sharing a File System With Multiple Kerberos Security Modes

In this example, all three Kerberos security modes have been selected. If no security mode is specified when a mount request is made, the first mode listed is used on all NFS V3 clients (in this case, `krb5`). Additional information can be found in “SEAM Commands” on page 189.

```
# share -F nfs -o sec=krb5:krb5i:krb5p /export/home
```

Configuring SEAM Clients

SEAM clients include any host, not a KDC server, on the network that needs to use SEAM services. This section provides a procedure for installing a SEAM client, as well as specific information about using root authentication to mount NFS file systems.

▼ How to Configure a SEAM Client

The following configuration parameters are used:

```
realm name = EXAMPLE.COM
DNS domain name = example.com
master KDC = kdc1.example.com
slave KDC = kdc2.example.com
client = client.example.com
admin principal = kws/admin
user principal = mre
online help URL =
http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in the *SEAM Installation and Release Notes*.

1. Prerequisites for configuring a SEAM client.

The SEAM client software must be installed.

2. Edit the Kerberos configuration file (krb5.conf).

To change the file from the SEAM default version, you need to change the realm names and the names of the servers, as well as identifying the path to the help files for gkadmin.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM

[realms]
    EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc1.example.com
    }

[domain_realm]
    .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
    default = FILE:/var/krb5/kdc.log
    kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

3. Optional: Synchronize with the master KDC's clock using NTP or another clock synchronization mechanism.

See "Synchronizing Clocks between KDCs and SEAM Clients" on page 110 for information about NTP.

4. Optional: Create a user principal if one does not already exist.

You only need to create a user principal, if the user associated with this host does not have a principal assigned already. See "How to Create a New Principal" on page 151 for instructions using the SEAM Administration Tool. A command line example is shown below.

```
client1 # /usr/sbin/kadmin -p kws/admin
Enter password: <Enter kws/admin password>
kadmin: addprinc mre
Enter password for principal mre@EXAMPLE.COM: <type the password>
Re-enter password for principal mre@EXAMPLE.COM: <type it again>
kadmin:
```

5. Create a root principal.

```
kadmin: addprinc root/client1.example.com
Enter password for principal root/client1.example.com@EXAMPLE.COM: <type the password>
```

```
Re-enter password for principal root/client1.example.com@EXAMPLE.COM: <type it again>
kadmin: quit
```

6. (Optional) If you want a user on the SEAM client to automatically mount Kerberized NFS file systems using Kerberos authentication, you must authenticate the root user.

This process is done most securely by using the `kinit` command; however, users will need to use `kinit` as `root` every time they need to mount a file system secured by Kerberos. You can choose to use a keytab file instead. See “Setting Up Root Authentication to Mount NFS File Systems” on page 110 for detailed information about the keytab requirement.

```
client1 # /usr/bin/kinit root/client1.example.com
Password for root/client1.example.com@EXAMPLE.COM: <Enter password>
```

To use the keytab file option, add the `root` principal to the client’s keytab using `kadmin`:

```
client1 # /usr/sbin/kadmin -p kws/admin
Enter password: <Enter kws/admin password>
kadmin: ktadd root/client1.example.com
kadmin: Entry for principal root/client.example.com with
kvno 3, encryption type DES-CBC-CRC added to keytab
WRFILE:/etc/krb5/krb5.keytab
kadmin: quit
```

7. If you want the client to warn users about Kerberos ticket expiration, create an entry in the `/etc/krb5/warn.conf` file.

See `warn.conf` (4) for more information.

Example-Setting Up a SEAM Client Using a Non-SEAM KDC

It is possible to set up a SEAM Client to work with a non-SEAM KDC. In this case, a line must be included in `/etc/krb5/krb5.conf` in the `realms` section to change the protocol used when communicating with the Kerberos password-changing server. The format of this line is shown below.

```
[realms]
    EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc1.example.com
        kpasswd_protocol = SET_CHANGE
    }
```

Setting Up Root Authentication to Mount NFS File Systems

If users want to access a non-Kerberized NFS file system, either the NFS file system can be mounted as `root`, or the file system can be accessed automatically through the automounter whenever they access it (without requiring `root` permissions).

Mounting a Kerberized NFS file system is very much the same, but it does incur an additional obstacle. To mount a Kerberized NFS file system, users must use the `kinit` command as `root` to obtain credentials for the client's `root` principal, because a client's `root` principal is typically not in the client's keytab. This is true even when the automounter is set up. Not only is this an extra step, but it forces all users to know their system's `root` password and the `root` principal's password.

To bypass this, you can add a client's `root` principal to the client's keytab, which will automatically provide credentials for `root`. Although this enables users to mount NFS file systems without running the `kinit` command and enhances ease-of-use, it is a security risk. For example, if someone gains access to a system with the `root` principal in its keytab, the person has the capability of obtaining credentials for `root`. So make sure you take the appropriate security precautions. See "Administering Keytabs" on page 172 for more information.

Synchronizing Clocks between KDCs and SEAM Clients

All hosts participating in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as *clock skew*), which provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, client requests will be rejected.

The clock skew also determines how long application servers must keep track of all Kerberos protocol messages, in order to recognize and reject replayed requests. So, the longer the clock skew value, the more information that application servers have to collect.

The default value for the maximum clock skew is 300 seconds (five minutes), which you can change in the `libdefaults` section of the `krb5.conf` file.

Note – For security reasons, do not increase the clock skew beyond 300 seconds.

Since it is important to maintain synchronized clocks between the KDCs and SEAM clients, it is recommended that you use the Network Time Protocol (NTP) software to do this. The Network Time Protocol (NTP) public domain software from the University of Delaware is included in the Solaris software starting with the Solaris 2.6 release.

Note – Another way to synchronize clocks is to use the `rdate` command and cron jobs, which can be a less involved process than using NTP. However, this section will continue to focus on using NTP. And, if you use the network to synchronize the clocks, the clock synchronization protocol must itself be secure.

NTP enables you to manage precise time and/or network clock synchronization in a network environment. NTP is basically a server/client implementation. You pick one system to be the master clock (NTP server), and then you set up all your other systems to synchronize their clocks with the master clock (NTP clients). This is all done through the `xntpd` daemon, which sets and maintains a UNIX system time-of-day in agreement with Internet standard time servers. Figure 8–1 shows an example of the using the server/client NTP implementation.

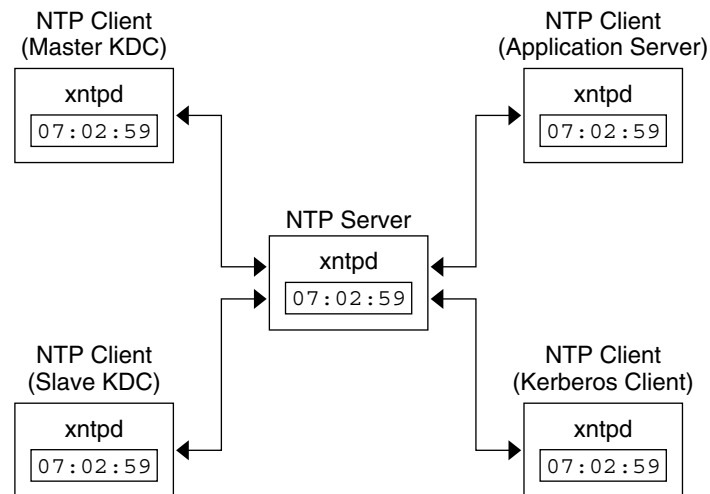


FIGURE 8-1 Synchronizing Clocks Using NTP

To ensure that the KDCs and SEAM clients maintain synchronized clocks, implement the following steps:

1. Set up an NTP server on your network (this can be any system except the master KDC). See “How to Set Up an NTP Server” in *System Administration Guide: Resource Management and Network Services*XRE.
2. As you configure the KDCs and SEAM clients on the network, set them up to be NTP clients of the NTP server. See “How to Set Up an NTP Client” in *System Administration Guide: Resource Management and Network Services*.

Swapping Master and Slave KDCs

These procedures should be used to make the swapping of a master with a slave KDC easier. This should only be done if the master KDC server fails for some reason or if the master needs to be re-installed (new hardware for example).

▼ How to Configure a Swappable Slave KDC

This procedure should be done on the slave KDC server that you want to have available to become the master.

1. Use alias names for master and swappable slave KDC servers during the installation.

When defining the hostnames for the KDCs, make sure that each system has an alias included in DNS and use the alias names when defining the hosts in `/etc/krb5/krb5.conf`.

2. Disable hostname checking in the KDC startup script on both master and swappable slave.

Using alias names for the master and swappable slave KDC servers means that the check which verifies that the current nodename is in `/etc/krb5/krb5.conf` before starting the KDC server fails. To use the alias names so that the swapping is easy to do, you need to comment out two lines in `/etc/init.d/kdc` as shown below:

```
if [ -f $KDC_CONF_DIR/kdc.conf ]
then
#       node=`uname -n`
#       if grep -i "kdc.*=.*$node" /etc/krb5/krb5.conf > /dev/null 2>&1 ;
#           then
#               $BINDIR/krb5kdc
#           fi
fi
;;
```


3. Install master KDC software.

Installing the master KDC software provides the binaries and other files that will be needed during a swap, which includes all of the files that a slave KDC server requires. Do not reboot the system when the installation is complete.

4. Follow steps to install a slave KDC.

Prior to any swapping, this server should function just like any other slave KDC in the realm. See "How to Configure a Slave KDC" on page 97 for instructions. Do not install the slave software. All of the files that are required are installed when the master software is installed.

5. Move master KDC commands.

To prevent the master KDC commands from being run from this slave, move `kprop`, `kadmind` and `kadmin.local` to a reserved place.

```
kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
```

6. Disable kadmind startup in /etc/init.d/kdc.master.

To prevent the slave from handling requests to change the KDC database, comment out the line that starts `kadmind` in the script:

```
kdc4 # cat /etc/init.d/kdc.master

.
.

case "$1" in
'start')

        if [ -f $KDC_CONF_DIR/kdc.conf ]
        then
#                $BINDIR/kadmind
        fi
        ;;
```

7. Comment out kprop line in the root crontab file.

This step prevents the slave from propagating its copy of the KDC database.

```
kdc4 # crontab -e
#ident    "@(#)root      1.19    98/07/06 SMI"    /* SVr4.0 1.1.3.1    */
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * 0,4 /etc/cron.d/logchecker
10 3 * * 0 /usr/lib/newsyslog
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
```

```

1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5kprop_script kdc1.example.sun.com #SUNWkr5ma

```

▼ How to Swap a Master and Slave KDC

This procedure requires that the slave KDC server has been set up as a swappable slave (see “How to Configure a Swappable Slave KDC” on page 112). In this procedure the master server that is being swapped out is named `kdc1` and the slave that will become the new master is named `kdc4`.

1. On the old master: Kill the `kadmind` process.

Killing the `kadmind` process prevents any changes from being made to the KDC database.

```
kdc1 # /etc/init.d/kdc.master stop
```

2. On the old master: Comment out `kprop` line in the root crontab file.

This step prevents the old master from propagating its copy of the KDC database.

```

kdc1 # crontab -e
#ident "@(#)root 1.19 98/07/06 SMI" /* SVr4.0 1.1.3.1 */
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * 0,4 /etc/cron.d/logchecker
10 3 * * 0 /usr/lib/newsyslog
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.sun.com #SUNWkr5ma

```

3. On the old master: Disable `kadmind` startup in `/etc/init.d/kdc.master`.

To prevent the master from restarting `kadmind` if the server is rebooted, comment out the line that starts `kadmind` in the script:

```

kdc1 # cat /etc/init.d/kdc.master

.
.

case "$1" in
'start')

        if [ -f $KDC_CONF_DIR/kdc.conf ]
        then

#                $BINDIR/kadmind

```

```
fi
;;
```

4. On the old master: Run kprop_script to back up and propagate the database.

```
kdc1 # /usr/lib/krb5/kprop_script kdc4.example.com
Database propagation to kdc4.example.com: SUCCEEDED
```

5. On the old master: Move master KDC commands.

To prevent the master KDC commands from being run, move kprop, kadmind and kadmin.local to a reserved place.

```
kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
```

6. On the DNS server: Change alias names for the master.

To change the servers, edit the example.com zone file and change the entry for masterkdc.

```
masterkdc IN CNAME kdc4
```

7. On the DNS server: Restart internet domain name server.

Run the following command on both servers to get the new alias information:

```
# pkill -1 in.named
```

8. On the new master: Move master KDC commands.

```
kdc4 # mv /usr/lib/krb5/kprop.save /usr/lib/krb5/kprop
kdc4 # mv /usr/lib/krb5/kadmind.save /usr/lib/krb5/kadmind
kdc4 # mv /usr/sbin/kadmin.local.save /usr/sbin/kadmin.local
```

9. On the new master: Create a keytab file for kadmin using kadmin.local.

This command sequence creates a special keytab file with principal entries for admin and changepw. These principals are needed for the kadmind service.

```
kdc4 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc4.example.com
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type DES-CBC-CRC
added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc4.example.com
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type DES-CBC-CRC
added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

10. On the new master: Enable kadmind startup in /etc/init.d/kdc.master.

```
kdc4 # cat /etc/init.d/kdc.master
```

```
.
.
```

```

case "$1" in
'start')

    if [ -f $KDC_CONF_DIR/kdc.conf ]
    then
        $BINDIR/kadmind
    fi
    ;;

```

11. On the new master: Start kadmind.

```
kdc4 # /etc/init.d/kdc.master start
```

12. Enable the kprop line in the root crontab file.

```

kdc4 # crontab -e
#ident    "@(#)root        1.19    98/07/06 SMI"    /* SVr4.0 1.1.3.1    */
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * 0,4 /etc/cron.d/logchecker
10 3 * * 0 /usr/lib/newsyslog
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
10 3 * * * /usr/lib/krb5/kprop_script kdc1.example.sun.com #SUNWkr5ma

```

Administering the Kerberos Database

The Kerberos database is the backbone of Kerberos and must be maintained properly. This section provides some of the procedures on how to administer the Kerberos database, such as backing up and restoring the database, setting up parallel propagation, and administering the stash file. The steps to set up the database initially can be found in “How to Configure a Master KDC” on page 93.

Backing Up and Propagating the Kerberos Database

Propagating the Kerberos database from the master KDC to the slave KDCs is one of the most important configuration tasks. If propagation doesn’t happen often enough, the master KDC and slave KDCs will become out-of-sync, so if the master KDC goes

down, the slave KDCs will not have the most recent database information. Also, if a slave KDC has been configured as a master for purposes of load balancing, the clients using that slave as a master KDC will not have the latest information. Therefore, it is important to make sure the propagation occurs often enough, based on how often you change the Kerberos database.

When you configure the master KDC, you set up the `kprop_script` in a cron job to automatically back up the Kerberos database to the `/var/krb5/slave_datatrans` dump file and propagate it to the slave KDCs. But, as with any file, the Kerberos database can become corrupted. If this happens on one of the slave KDCs, you might never notice, since the next automatic propagation of the database installs a fresh copy. However, if it happens to the master KDC, the corrupted database is propagated to all of the slaves during the next propagation. And, the corrupted backup overwrites the previous uncorrupted backup file on the master KDC.

Because there is no “safe” backup copy in this scenario, you should also set up a cron job to periodically copy the `slave_datatrans` dump file to another location or to create another separate backup copy by using the `dump` command of `kdb5_util`. Then, if your database becomes corrupted, you can restore the most recent backup on the master KDC by using the `load` command of `kdb5_util`.

Another important note is that, because the database dump file contains principal keys, you need to protect the file from being accessed by unauthorized users (by default, the database dump file has read/write permissions only as `root`). This includes using only the `kprop` command to propagate the database dump file, which encrypts the data being transferred. Also, `kprop` propagates the data only to the slave KDCs, which minimizes the chance of accidentally sending the database dump to unauthorized hosts.



Caution – If the Kerberos database is updated after it has been propagated and if the database subsequently is corrupted before the next propagation, the slaves will not contain the updates: the updates will be lost. Because of this scenario, if you add significant updates to the database before a regularly scheduled propagation, you should manually propagate the database to avoid data loss.

`kpropd.ac1` File

The `kpropd.ac1` file on a KDC provides a list of host principal names, one per line, that specifies the systems from which the KDC can receive an updated database through the propagation mechanism. If the master KDC is used to propagate all the slave KDCs, the `kpropd.ac1` file on each slave needs to contain only the host principal name of the master.

However, the SEAM installation and subsequent configuration steps in this guide instruct you to add the same `kpropd.ac1` file to the master and slave KDCs. The file contains all the KDC host principal names. This configuration allows you to propagate

from any KDC, in case the propagating KDCs become temporarily unavailable. And, keeping an identical copy on all KDCs makes it easy to maintain.

kprop_script Command

The `kprop_script` command uses the `kprop` command to propagate the Kerberos database to other KDCs. (If the `kprop_script` is run on a slave KDC, it propagates the slave's copy of the Kerberos database to other KDCs.) The `kprop_script` accepts a list of host names for arguments, separated by spaces, which denote the KDCs to propagate.

When the `kprop_script` is run, it creates a backup of the Kerberos database to the `/var/krb5/slave_data` file and copies the file to the specified KDCs. The Kerberos database is locked until the propagation is finished.

▼ How to Back Up the Kerberos Database

1. Become superuser on the master KDC.
2. Back up the Kerberos database by using the `dump` command of `kdb5_util`.

```
# /usr/sbin/kdb5_util dump [-verbose] [-d dbname] [filename [principals...]]
```

<code>-verbose</code>	Prints the name of each principal and policy that is being backed up.
<code>dbname</code>	The name of the database to back up. Note that ".db" is appended to whatever database name is specified, and an absolute path for the file can be specified. If the <code>-d</code> option is not specified, the default database name is <code>/var/krb5/principal</code> , which actually becomes <code>/var/krb5/principal.db</code> .
<code>filename</code>	The file to back up the database. An absolute path for the file can be specified. If you don't specify a file, the database is dumped to standard output.
<code>principal</code>	A list of one or more principals (separated by a space) to back up. You must use fully-qualified principal names. If you don't specify principals, the entire database is backed up.

Example—Backing Up the Kerberos Database

The following example backs up the Kerberos database to a file called `dumpfile`. Because the `-verbose` option is specified, each principal is printed as it is backed up.

```
# kdb5_util dump -verbose dumpfile
kadmin/kdc1.eng.example.com@ENG.EXAMPLE.COM
krbtgt/eng.example.com@ENG.EXAMPLE.COM
kadmin/history@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
changepw/kdc1.eng.example.com@ENG.EXAMPLE.COM
#
```

The following example backs up the `pak` and `pak/admin` principals from the Kerberos database.

```
# kdb5_util dump -verbose dumpfile pak/admin@ENG.EXAMPLE.COM pak@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
#
```

▼ How to Restore the Kerberos Database

1. Become superuser on the master KDC.
2. Restore the Kerberos database by using the `load` command of `kdb_util`.

```
# /usr/sbin/kdb5_util load [-verbose] [-d dbname] [-update] [filename]
```

`-verbose`

Prints the name of each principal and policy that is being restored.

dbname

The name of the database to restore. Note that ".db" is appended to whatever database name is specified, and an absolute path for the file can be specified. If the `-d` option is not specified, the default database name is `/var/krb5/principal`, which actually becomes `/var/krb5/principal.db`.

`-update`

Updates the existing database; otherwise a new database is created or the existing database is overwritten.

filename

The file from which to restore the database. An absolute path for the file can be specified.

Example—Restoring the Kerberos Database

The following example restores the database called `database1.db` into the current directory from the `dumpfile` file. Since the `-update` option isn't specified, a new database is created by the restore.

```
# kdb5_util load -d database1 dumpfile
```

▼ How to Manually Propagate the Kerberos Database to the Slave KDCs

This procedure shows you how to propagate the Kerberos database using the `kprop` command. You can use this if you need to sync a slave KDC with the master KDC outside the periodic cron job. And, unlike the `kprop_script`, you can use `kprop` to propagate just the current database backup without first making a new backup of the database.

1. **Become superuser on the master KDC.**
2. **(Optional) Back up the database by using the `kdb5_util` command.**
3. **Propagate the database to a slave KDC by using the `kprop` command.**

```
# /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
```

```
# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave_KDC
```

If you want to back up the database and propagate it to a slave KDC outside the periodic cron job, you can also use the `kprop_script` command as follows:

```
# /usr/lib/krb5/kprop_script slave_KDC
```

Setting Up Parallel Propagation

In most cases, the master KDC is used exclusively to propagate its database to the slave KDCs. However, if your site has a lot of slave KDCs, you might want to consider load-sharing the propagation process, known as *parallel propagation*.

Parallel propagation allows specific slave KDCs to share the propagation duties with the master KDC. This enables the propagation to be done faster and to lighten the work for the master KDC.

For example, say your site has one master and six slaves (shown in Figure 8–2), where `slave-1` through `slave-3` consist of one logical grouping and `slave-4` through `slave-6` consist of the other. To set up parallel propagation, you could have the

master KDC propagate the database to `slave-1` and `slave-4`, and those slaves could in turn propagate the database to the slaves in their group.

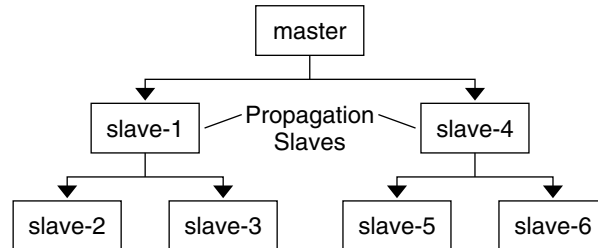


FIGURE 8-2 Example Parallel Propagation Configuration

▼ How to Set Up Parallel Propagation

This is not a detailed step-by-step procedure, but a high-level list of configuration steps to enable parallel propagation.

1. On the master KDC, change the `kprop_script` entry in its cron job to include arguments for only the slaves that will perform the succeeding propagation (*propagation slaves*).
2. On each propagation slave, add a `kprop_script` entry to its cron job, which must include arguments for the slaves to propagate. To successfully propagate in parallel, the cron job should be set up to run after the propagation slave is itself propagated with the new database.

Note – Determining how long it will take for a propagation slave to be propagated depends on factors such as network bandwidth and the size of the database.

3. On each slave KDC, set up the appropriate permissions to be propagated. This is done by adding the host principal name of its propagating KDC to its `kpropd.acl` file.

Example — Setting Up Parallel Propagation

Using the example in Figure 8-2, the master KDC's `kprop_script` entry would look something like this:

```
0 3 * * * /usr/lib/krb5/kprop_script slave-1.example.com slave-4.example.com
```

`slave-1`'s `kprop_script` entry would look something like this (note that the propagation on the slave starts an hour after it is propagated by the master):

```
0 4 * * * /usr/lib/krb5/kprop_script slave-2.example.com slave-3.example.com
```

The `kpropd.acl` file on the propagation slaves would have to contain the following entry:

```
host/master.example.com@EXAMPLE.COM
```

The `kpropd.acl` file on the slaves being propagated by `slave-1` would have to contain the following entry:

```
host/slave-1.example.com@EXAMPLE.COM
```

Administering the Stash File

The stash file contains the master key for the Kerberos database, which is automatically created when you create a Kerberos database. If the stash file gets corrupted, you can use the `stash` command of `kdb5_util` (1M) to replace the corrupted file. The only time you should need to remove a stash file is after removing the Kerberos database with the `destroy` command of `kdb5_util`. Because the stash file isn't automatically removed with the database, you have to remove it to finish the cleanup.

▼ How to Remove a Stash File

1. **Become superuser on the KDC that contains the stash file.**
2. **Remove the stash file.**

```
# rm stash_file
```

stash_file

The path to the stash file. By default, the stash file is located at `/var/krb5/.k5.realm`.

If you need to recreate the stash file, you can use the `-f` option of the `kdb5_util` command.

Increasing Security

These procedures list steps that you can use to increase security on SEAM application servers and on KDC servers.

▼ How to Restrict Access for KDC servers

Both master and slave KDC servers have copies of the KDC database stored locally. Restricting access to these servers so that the databases are secure is important to the overall security of the SEAM installation.

1. Disable remote services in `/etc/inetd.conf`.

To provide a secure KDC server, all non-essential network services should be disabled by commenting out the entry that starts the service in `/etc/inetd.conf`. In most circumstances the only services that would need to run would be `time` and `krdb5_kprop`. In addition, any services that use loopback `tli` (`ticlts`, `ticotsord`, and `ticots`) can be left enabled. After editing, the file should look something like (to shorten the example many comments have been removed):

```
kdc1 # cat /etc/inetd.conf
#
#ident    "@(#)inetd.conf 1.33    98/06/02 SMI"    /* SVr4.0 1.5 */
.
.
#name     dgram    udp     wait    root    /usr/sbin/in.tnamed    in.tnamed
#
#shell    stream   tcp     nowait  root    /usr/sbin/in.rshd     in.rshd
#login    stream   tcp     nowait  root    /usr/sbin/in.rlogind  in.rlogind
#exec     stream   tcp     nowait  root    /usr/sbin/in.rexecd   in.rexecd
#comsat   dgram    udp     wait    root    /usr/sbin/in.comsat   in.comsat
#talk     dgram    udp     wait    root    /usr/sbin/in.talkd    in.talkd
#
#uucp     stream   tcp     nowait  root    /usr/sbin/in.uucpd    in.uucpd
#
#finger   stream   tcp     nowait  nobody  /usr/sbin/in.fingerd  in.fingerd
#
# Time service is used for clock synchronization.
#
time      stream   tcp     nowait  root    internal
time      dgram    udp     wait    root    internal
#
.
.
#
100234/1  tli rpc/ticotsord wait    root    /usr/lib/gss/gssd     gssd
#dtspc    stream   tcp     nowait  root    /usr/dt/bin/dtspcd    /usr/dt/bin/dtspcd
```

```
#100068/2-5 dgram rpc/udp wait root /usr/dt/bin/rpc.cmsd rpc.cmsd
100134/1 tli rpc/ticotsord wait root /usr/lib/ktkt_warnd kwarnd
krb5_prop stream tcp nowait root /usr/lib/krb5/kpropd kpropd
```

Reboot the server after the changes are made.

2. Restrict access to the hardware supporting the KDC.

In order to restrict physical access, make sure that the server and its monitor are located in a secure facility. Normal users should not be able to access this server in any way.

3. Store KDC database backups on local disks or on the slaves.

Making tape backups of your KDC should only be done if the tapes are stored securely. This is also true for copies of keytab files. It would be best to store these files on a local file system that is not shared to other systems. The storage file system can be on either the master KDC server or any of the slaves.

SEAM Error Messages and Troubleshooting

This chapter provides resolutions for error messages that you might receive, as well as some troubleshooting tips for various problems when using SEAM. This is a list of the error message and troubleshooting information in this chapter.

- “SEAM Administration Tool Error Messages” on page 125
- “Common SEAM Error Messages (A-M)” on page 126
- “Common SEAM Error Messages (N-Z)” on page 132
- “Problems With the Format of the `krb5.conf` File” on page 135
- “Problems Propagating the Kerberos Database” on page 135
- “Problems Mounting a Kerberized NFS File System” on page 136
- “Problems Authenticating as Root” on page 137

SEAM Error Messages

This section provides information about SEAM error messages, including why each error occurred and a way to fix it.

SEAM Administration Tool Error Messages

Unable to view the list of principals or policies; use the Name field.

Cause: The admin principal that you logged on with does not have the list privilege (1) in the Kerberos ACL file (`kadm5.ac1`), so you cannot view the principal or policy lists.

Solution: The admin principal that you logged on with does not have the list privilege (1) in the Kerberos ACL file (`kadm5.ac1`), so you cannot view the principal or policy lists.

```
JNI: Java array creation failed
JNI: Java class lookup failed
JNI: Java field lookup failed
JNI: Java method lookup failed
JNI: Java object lookup failed
JNI: Java object field lookup failed
JNI: Java string access failed
JNI: Java string creation failed
```

Cause: There is a serious problem with the Java Native Interface used by the SEAM Administration Tool (`gkadmin`).

Solution: Exit `gkadmin` and restart it; if the problem persists, please report a bug.

Common SEAM Error Messages (A-M)

This section provides an alphabetical list (A-M) of the more common error messages for the SEAM commands, SEAM daemons, PAM framework, GSS interface, and the Kerberos library.

```
major_error minor_error gssapi error importing name
```

Cause: An error occurred while importing a service name.

Solution: Make sure the service principal is in the host's keytab file.

```
Bad krb5 admin server hostname while initializing kadmin
interface
```

Cause: An invalid host name is configured for the admin server (master KDC) in the `krb5.conf` file.

Solution: Make sure the correct host name is specified in the `krb5.conf` file for the admin server (master KDC).

```
Cannot contact any KDC for requested realm
```

Cause: No KDC responded in the requested realm.

Solution: Make sure at least one KDC (either the master or slave) is reachable or that the `krb5kdc` daemon is running on the KDCs. Look in `/etc/krb5/krb5.conf` for the list of configured KDCs (`kdc = kdc_name`).

Cannot determine realm for host

Cause: Kerberos cannot determine the realm name for the host.

Solution: Make sure there is a default realm name or the domain name mappings are set up in the Kerberos configuration file (`krb5.conf`).

Cannot find KDC for requested realm

Cause: No KDC was found in the requested realm.

Solution: Make sure the Kerberos configuration file (`krb5.conf`) specifies a KDC in the realm section.

cannot initialize realm *realm_name*

Cause: The KDC may not have a stash file.

Solution: Make sure the KDC has a stash file. If not, create one using the `kdb5_util(1M)` command and try running `krb5kdc` again (`/etc/init.d/kdc`).

Cannot resolve KDC for requested realm

Cause: Kerberos cannot determine any KDC for the realm.

Solution: Make sure the Kerberos configuration file (`krb5.conf`) specifies a KDC in the realm section.

Cannot reuse password

Cause: The password you entered has been used before by this principal.

Solution: Choose a password that has not been chosen before, at least not within the number of passwords kept in the KDC database for each principal (this is enforced by the principal's policy).

Can't get forwarded credentials

Cause: Credential forwarding could not be established.

Solution: Make sure the principal has forwardable credentials.

Can't open/find Kerberos configuration file

Cause: The Kerberos configuration file (`krb5.conf`) was not available.

Solution: Make sure the `krb5.conf` file is available in the correct location and has the correct permissions (it should be writable by root and readable by everyone else).

Client/server realm mismatch in initial ticket request

Cause: A realm mismatch between the client and server occurred in the initial ticket request.

Solution: Make sure the server you are communicating with is in the same realm as the client or that the realm configurations are correct.

Client or server has a null key

Cause: The principal has a null key.

Solution: Modify the principal to have a non-null key by using the `cpw` command of `kadmin(1M)`.

Communication failure with server while initializing `kadmin` interface

Cause: The host entered for the admin server (master KDC) did not have `kadmind` running.

Solution: Make sure you specified the correct host name for the master KDC. If you specified the correct host name, make sure that `kadmind` is running on the master KDC you specified.

Credentials cache file permissions incorrect

Cause: You do not have the appropriate read or write permissions on the credentials cache (`/tmp/krb5cc_uid`).

Solution: Make sure you have read and write permissions on the credentials cache.

Credentials cache I/O operation failed XXX

Cause: Kerberos had a problem writing to the system's credentials cache (`/tmp/krb5cc_uid`).

Solution: Make sure the credentials cache has not been removed and there is space left on the device by using the `df` command.

Decrypt integrity check failed

Cause: You might have an invalid ticket.

Solution:

1. Make sure your credentials are valid. Destroy your tickets with `kdestroy` and create new tickets with `kinit`.
2. Make sure the target host has a keytab with the correct version of the service key. Use `kadmin(1M)` to view the key version number of the service principal (for example, `host/FQDN_hostname`) in the Kerberos database and use `klist -k` on the target host to make sure it has the same key version number.

`df: cannot statvfs filesystem: Invalid argument`

Cause: The `df` command cannot access the Kerberized NFS file system, which is currently mounted, to generate its report, because you no longer have the appropriate root credentials. Destroying your credentials for a mounted Kerberized file system does not automatically unmount the file system.

Solution: You must create new root credentials to access the Kerberized file system. If you no longer require access to the Kerberized file system, unmount the file system.

failed to obtain credentials cache

Cause: During `kadmin` initialization, a failure occurred when `kadmin` tried to obtain credentials for the `admin` principal.

Solution: Make sure you used the correct principal and/or password when executing `kadmin`.

Field is too long for this implementation

Cause: The message size being sent by a Kerberized application was too long. The maximum message size that can be handled by Kerberos is 65535 bytes. In addition, there are limits on individual fields within a protocol message sent by Kerberos.

Solution: Make sure that your Kerberized applications are sending valid message sizes.

GSS-API (or Kerberos) error

Cause: This is a generic GSS-API or Kerberos error message and can be caused by several different problems.

Solution: Look at the `/etc/krb5/kdc.log` file to find the more specific GSS-API error message that was logged when this error occurred.

Hostname cannot be canonicalized

Cause: Kerberos cannot make the host name fully qualified.

Solution: Make sure the host name is in DNS and the host-name-to-address and address-to-host-name mappings are consistent.

Illegal cross-realm ticket

Cause: The ticket sent did not have the correct cross-realms. The realms may not have the correct trust relationships set up.

Solution: Make sure the realms you are using have the correct trust relationships.

Improper format of Kerberos configuration file

Cause: The Kerberos configuration file (`krb5.conf`) has invalid entries.

Solution: Make sure all the relations in the `krb5.conf` file are followed by the "=" sign and a value, and verify that the brackets are present in pairs for each subsection.

Inappropriate type of checksum in message

Cause: The message contained an invalid checksum type.

Solution: Check which valid checksum types are specified in the `krb5.conf` and `kdc.conf` files.

Incorrect net address

Cause: There was a mismatch in the network address. The network address in the ticket being forwarded was different from the network address where the ticket was processed. This may occur when forwarding tickets.

Solution: Make sure the network addresses are correct; destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

Invalid flag for file lock mode

Cause: An internal Kerberos error occurred.

Solution: Please report a bug.

Invalid message type specified for encoding

Cause: Kerberos could not recognize the message type sent by the Kerberized application.

Solution: If you are using a Kerberized application developed by your site or a vendor, make sure it is using Kerberos correctly.

Invalid number of character classes

Cause: The password you entered for the principal does not contain enough password classes, as enforced by the principal's policy.

Solution: Make sure you enter a password with the minimum number of password classes that the policy requires.

KADM err: Memory allocation failure

Cause: There is not enough memory to run `kadmin`.

Solution: Free up memory and try running `kadmin` again.

KDC can't fulfill requested option

Cause: The KDC did not allow the requested option. A possible problem may be that postdating or forwardable options were being requested and the KDC did not allow it. Another problem may be that you requested the renewal of a TGT but you didn't have a renewable TGT.

Solution: Determine if you are requesting an option that either the KDC does not allow or if you are requesting something you don't have.

KDC policy rejects request

Cause: The KDC policy did not allow the request. For example, the request to the KDC did not have an IP address in its request, or forwarding was requested but the KDC did not allow it.

Solution: Make sure you are using `kinit` with the correct options. If necessary, modify the policy associated with the principal or change the principal's attributes to allow the request. You can modify the policy or principal by using `kadmin(1M)`.

KDC reply did not match expectations

Cause: The KDC reply did not contain the expected principal name, or other values in the response were incorrect.

Solution: Make sure the KDC you are communicating with complies with RFC1510, the request you are sending is a Kerberos V5 request, or that the KDC is available.

Key table entry not found

Cause: There is no entry for the service principal in the network application server's keytab.

Solution: Add the appropriate service principal to the server's keytab so it can provide the Kerberized service.

Key version number for principal in key table is incorrect

Cause: A principal's key version is different in the keytab and in the Kerberos database. Either a service's key has been changed or you may be using an old service ticket.

Solution: If a service's key has been changed (for example, by using `kadmin`), you need to extract the new key and store it in the host's keytab where the service is running.

Alternately, you may be using an old service ticket that has an older key. You may want to do a `kdestroy` and then a `kinit` again.

login: load_modules: can not open module
/usr/lib/security/pam_krb5.so.1

Cause: Either the Kerberos PAM module is missing or it is not a valid executable binary.

Solution: Make sure the Kerberos PAM module is in `/usr/lib/security` and it is a valid executable binary. Also, make sure `/etc/pam.conf` contains the correct path to `pam_krb5.so.1`.

Looping detected inside `krb5_get_in_tkt`

Cause: Kerberos made several attempts to get the initial tickets but failed.

Solution: Make sure at least one KDC is responding to authentication requests.

Master key does not match database

Cause: The loaded database dump was not created from a database containing the master key, which is located in `/var/krb5/.k5.REALM`.

Solution: Make sure the master key in the loaded database dump matches the master key located in `/var/krb5/.k5.REALM`.

Matching credential not found

Cause: The matching credential for request was not found. Your request requires credentials that are not available in the credentials cache.

Solution: Destroy your tickets with `kdestroy` and create new tickets with `kinit`.

Message out of order

Cause: Messages sent using sequential-order privacy arrived out of order. Some messages may have been lost in transit.

Solution: You should re-initialize the Kerberos session.

Message stream modified

Cause: There was a mismatch between the computed checksum and message checksum. The message may have been modified while in transit, which may indicate a security leak.

Solution: Make sure that the messages are being sent across the network correctly. Since this message may also indicate possible tampering of messages while they are being sent, destroy your tickets using `kdestroy` and reinitialize the Kerberos services you are using.

Common SEAM Error Messages (N-Z)

This section provides an alphabetical list (N-Z) of the more common error messages for the SEAM commands, SEAM daemons, PAM framework, and the Kerberos library.

No credentials cache file found

Cause: Kerberos could not find the credentials cache (`/tmp/krb5cc_uid`).

Solution: Make sure the credential file exists and is readable. If it isn't, try performing a `kinit` again.

Operation requires "*privilege*" privilege

Cause: The admin principal being used does not have the appropriate privilege configured in the `kadm5.ac1` file.

Solution: Use a principal that has the appropriate privileges or configure the principal being used to have the appropriate privileges by modifying the `kadm5.ac1` file. Usually, a principal with `/admin` as part of its name has the appropriate privileges.

PAM-KRB5: Kerberos V5 authentication failed: password incorrect

Cause: Your UNIX password and Kerberos passwords are different. Most non-Kerberized commands, such as `login`, are set up through PAM to automatically authenticate with Kerberos by using the same password that you specified for your UNIX password. If your passwords are different, the Kerberos authentication fails.

Solution: You must enter your Kerberos password when prompted.

Password is in the password dictionary

Cause: The password that you entered is in a password dictionary that is being used. It is not a good choice for a password.

Solution: Choose a password that has a mix of password classes.

Permission denied in replay cache code

Cause: The system's replay cache could not be opened. The server may have been first run under a user ID different than your current user ID.

Solution: Make sure the replay cache has the appropriate permissions. The replay cache is stored on the host where the Kerberized server application is running (`/usr/tmp/rc_service_name`). Instead of changing the permissions on the current replay cache, you can also remove the replay cache before running the Kerberized server under a different user ID.

Protocol version mismatch

Cause: Most likely a Kerberos V4 request was sent to the KDC. SEAM supports only the Kerberos V5 protocol.

Solution: Make sure your applications are using the Kerberos V5 protocol.

Request is a replay

Cause: The request has already been sent to this server and processed. The tickets may have been stolen and someone else is trying to reuse the tickets.

Solution: Wait for a few minutes and re-issue the request.

Requested principal and ticket don't match

Cause: The service principal you are connecting to and the service ticket you have do not match.

Solution: Make sure DNS is functioning properly. If you are using another vendor's software, make sure it is using principal names correctly.

Requested protocol version not supported

Cause: Most likely a Kerberos V4 request was sent to the KDC. SEAM supports only the Kerberos V5 protocol.

Solution: Make sure your applications are using the Kerberos V5 protocol.

Required parameters in `krb5.conf` missing while initializing `kadmin` interface

Cause: There is a missing parameter (such as the `admin_server` parameter) in the `krb5.conf` file.

Solution: Determine what the missing parameter is and add it to `krb5.conf`.

Server rejected authentication (during `sendauth` exchange)

Cause: The server you are trying to communicate with rejected the authentication. Most often this error occurs when doing Kerberos database propagation. Some common causes may be problems with the `kpropd.acl` file, DNS, or keytabs.

Solution: If you get this error when running applications other than `kprop`, investigate whether the server's keytab is correct.

The ticket isn't for us

Ticket/authenticator don't match

Cause: There was a mismatch between the ticket and authenticator. The principal name in the request may not have matched the service principal's name, because the ticket was being sent with an FQDN name of the principal while the service expected non-FQDN or vice versa.

Solution: If you get this error when running applications other than `kprop`, investigate whether the server's keytab is correct.

Ticket expired

Cause: Your ticket times have expired.

Solution: Destroy your tickets with `kdestroy` and create new tickets with `kinit`.

Ticket is ineligible for postdating

Cause: The principal does not allow its tickets to be postdated.

Solution: Modify the principal with `kadmin(1M)` to allow postdating.

Ticket not yet valid

Cause: The postdated ticket is not valid yet.

Solution: Create new tickets with the correct date or wait until the current tickets are valid.

Truncated input file detected

Cause: The database dump file being used in the operation is not a complete dump file.

Solution: Create the dump file again or use a different database dump file.

Wrong principal in request

Cause: There was an invalid principal name in the ticket. It may be a DNS or FQDN problem.

Solution: Make sure the principal of the service matches the principal in the ticket.

SEAM Troubleshooting

This section provides troubleshooting information for the SEAM software.

Problems With the Format of the `krb5.conf` File

If the `krb5.conf` file is not formatted properly, the `telnet` command will fail. However, the `dtlogin` and `login` commands will still succeed, even if the `krb5.conf` file is specified as required for the commands. If this occurs, the following error message is displayed:

```
Error initializing krb5: Improper format of Kerberos configuration
```

If there is a problem with the format of the `krb5.conf` file, you are vulnerable to security breaches. You should fix the problem before allowing SEAM features to be used.

Problems Propagating the Kerberos Database

If propagating the Kerberos database fails, try `/usr/krb5/bin/rlogin -x` between the slave KDC and master KDC and vice versa.

Note – If the KDCs have been set up to restrict access, `rlogin` is disabled and cannot be used to troubleshoot this problem. To enable `rlogin` on a KDC, you must uncomment the `eklogin` entry in the `/etc/inetd.conf` file and restart `inetd`, as follows:

```
# ps -eaf | grep inetd           displays the process ID of inetd
# kill -1 pid_of_inetd
```

After you finish troubleshooting the problem, you need to change the `inetd.conf` file back to its original state and restart `inetd` again.

If `rlogin` does not work, problems are likely to be the keytabs on the KDCs. If `rlogin` does work, the problem is not in the keytab or the name service, since `rlogin` and the propagation software use the same `host/host_name` principal. In this case, make sure the `kpropd.ac1` file is correct.

Problems Mounting a Kerberized NFS File System

- If mounting a Kerberized NFS file system fails, make sure the `/var/tmp/rc_nfs` file exists on the NFS server. If it is not owned by root, remove it and try the mount again.
- If you have a problem accessing a Kerberized NFS file system, make sure there is an entry for `gssd` in the `inetd.conf` file on your system and the NFS server.
- If you see either the `invalid argument` or `bad directory` error message when trying to access a Kerberized NFS file system, the problem may be that you are not using a fully-qualified DNS name when trying to mount the NFS file system. The host being mounted is not the same as the host name part of the service principal in the server's keytab.

This may also occur if your server has multiple ethernet interfaces and you have set up DNS to use a "name per interface" scheme instead of a "multiple address records per host" scheme. For SEAM, you should set up multiple address records per host as follows¹ :

```
my.host.name.      A          1.2.3.4
                  A          1.2.4.4
                  A          1.2.5.4

my-en0.host.name.  A          1.2.3.4
my-en1.host.name.  A          1.2.4.4
my-en2.host.name.  A          1.2.5.4

4.3.2.1           PTR        my.host.name.
```

¹ Ken Hornstein, "Kerberos FAQ," [<http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html>], accessed 11 December 1998.


```
4.4.2.1 PTR my.host.name.  
4.5.2.1 PTR my.host.name.
```

In this example, the setup allows one reference to the different interfaces and allows a single service principal instead of three service principals in the server's keytab.

Problems Authenticating as Root

If the authentication fails when you try to become superuser on your system and you have already added the root principal to your host's keytab, there are two potential problems to check. First, make sure the root principal in the keytab has a fully-qualified name as its instance. If it does, check the `/etc/resolv.conf` file to make sure the system is correctly set up as a DNS client.

Administering Principals and Policies

This chapter provides procedures for managing principals and the policies associated with them. It also shows how to manage a host's keytab.

This chapter should be used by anyone who needs to administer principals and policies. Before using this chapter, you should be familiar with principals and policies, including any planning considerations. Refer to Chapter 6 and Chapter 7 respectively.

This is a list of step-by-step instructions in this chapter.

- "How to View the List of Principals" on page 147
- "How to View a Principal's Attributes" on page 149
- "How to Create a New Principal" on page 151
- "How to Duplicate a Principal" on page 153
- "How to Modify a Principal" on page 153
- "How to Delete a Principal" on page 155
- "How to Set Up Defaults for Creating New Principals" on page 155
- "How to Modify the Kerberos Administration Privileges" on page 156
- "How to View the List of Policies" on page 159
- "How to View a Policy's Attributes" on page 161
- "How to Create a New Policy" on page 163
- "How to Duplicate a Policy" on page 165
- "How to Modify a Policy" on page 165
- "How to Delete a Policy" on page 166
- "How to Add a Service Principal to a Keytab" on page 173
- "How to Remove a Service Principal From a Keytab" on page 174
- "How to Display the Keylist (Principals) in a Keytab" on page 175
- "How to Temporarily Disable Authentication for a Service on a Host" on page 176

Ways to Administer Principals and Policies

The Kerberos database on the master KDC contains all of your realm's Kerberos principals, their passwords, policies, and other administrative information. To create and delete principals, and modify their attributes, you can use the `kadmin(1M)` or `gkadmin(1M)` commands.

The `kadmin` command provides an interactive command-line interface that enables you to maintain Kerberos principals, policies, and keytabs. There are two versions of the `kadmin` command: `kadmin`, which uses Kerberos authentication to operate securely from anywhere on the network, and `kadmin.local`, which must be run directly on the master KDC. Other than `kadmin` using Kerberos to authenticate the user, the functionality of the two versions is identical. The local version is necessary to enable you to set up enough of the database to be able to use the remote version.

Also, provided with the SEAM product is the SEAM Administration Tool, `gkadmin`, which is an interactive graphical user interface (GUI) that essentially provides the same functionality as the `kadmin` command. See "SEAM Administration Tool" on page 140 for more information.

SEAM Administration Tool

The SEAM Administration Tool is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. It provides much the same functionality as the `kadmin` command; however, it does not support the management of keytabs. You must use the `kadmin` command to administer keytabs, which is described in "Administering Keytabs" on page 172.

Like the `kadmin` command, the SEAM Tool uses Kerberos authentication and encrypted RPC to operate securely from anywhere on the network. The SEAM Tool enables you to:

- Create new principals based on default values or existing principals
- Create new policies based on existing policies
- Add comments for principals
- Set up default values for creating new principals
- Log in as another principal without exiting the tool
- Print or save principal and policy lists
- View and search principal and policy lists

The SEAM Tool also provides context-sensitive and general online help.

The following task maps provide pointers to the various tasks you can do with the SEAM Tool:

- “Administering Principals Task Map” on page 146
- “Administering Policies Task Map” on page 158

Also, go to “SEAM Tool Panel Descriptions” on page 167 for descriptions of all the principal and policy attributes you can either specify or view in the SEAM Tool.

Command-Line Equivalents of the SEAM Tool

This section lists the `kadmin` commands that provide the same functionality as the SEAM Tool and can be used without running an X Window system. Even though most of the procedures in this chapter use the SEAM Tool, many of the procedures also provide corresponding examples using the command-line equivalents.

TABLE 10-1 Command-Line Equivalents of the SEAM Tool

Procedure	<code>kadmin</code> Command
Viewing the list of principals	<code>list_principals</code> or <code>get_principals</code>
Viewing a principal’s attributes	<code>get_principal</code>
Creating a new principal	<code>add_principal</code>
Duplicating a principal	No command-line equivalent
Modifying a principal	<code>modify_principal</code> and <code>change_password</code>
Deleting a principal	<code>delete_principal</code>
Setting up defaults for creating new principals	No command-line equivalent
Viewing the list of policies	<code>list_policies</code> or <code>get_policies</code>
Viewing a policy’s attributes	<code>get_policy</code>
Creating a new policy	<code>add_policy</code>
Modifying a policy	<code>modify_policy</code>
Duplicating a policy	No command-line equivalent
Deleting a policy	<code>delete_policy</code>

Files Modified by the SEAM Tool

The only file that the SEAM Tool modifies is the `$HOME/.gkadmin` file. It contains the default values for creating new principals and can be updated by choosing Properties from the Edit menu.

Print and Online Help Features of the SEAM Tool

The SEAM Tool provides both print and online help features. From the Print menu, you can send the following to a printer or a file:

- List of available principals on the specified master KDC
- List of available policies on the specified master KDC
- The currently selected or loaded principal
- The currently selected or loaded policy

From the Help menu, you can obtain context-sensitive help and general help. When you choose Context-Sensitive Help from the Help menu, the Context-Sensitive Help window is displayed and the tool is switched to help mode. In help mode, when you click on any of the fields, labels, or buttons on the window, help on that item is displayed in the Help window. To switch back to the tool's normal mode, click Dismiss in the Help window.

You can also choose Help Contents, which opens an HTML browser that provides pointers to the general overview and task information that is provided in this chapter.

Working With Large Lists in the SEAM Tool

As your site starts accumulating a large number of principals and policies, the time it takes the SEAM Tool to load and display the principal and policy lists will become increasingly longer and will slow down your productivity with the tool. There are several ways to work around this.

First, you can completely eliminate the time to load the lists by not having the SEAM Tool load the lists. You can set this option by choosing Properties from the Edit menu and unchecking the Show Lists field. Of course, when the tool doesn't load the lists, it can't display the lists and you can no longer use the list panels to select principals or policies. Instead, you must enter a principal or policy name in the new Name field that is provided, then select the operation you want to perform on it. Basically, entering a name becomes equivalent to selecting an item from the list.

Another way to work with large lists is to cache them. In fact, caching the lists for a limited time is set as the default behavior for the SEAM Tool. The SEAM Tool must still initially load the lists into the cache, but after that, the tool can use the cache

rather than retrieving the lists again. This eliminates the need to keep loading the lists from the server, which is what takes so long.

You can set list caching by also choosing Properties from the Edit menu. There are two cache settings. You can choose to cache the list forever, or you can specify a time limit when the tool must reload the lists from the server into the cache.

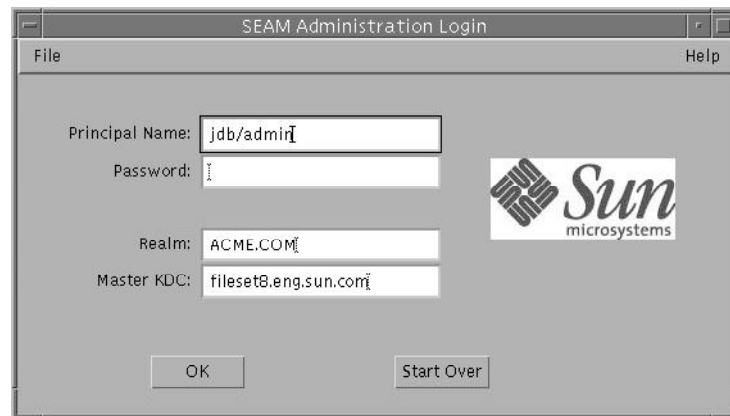
Caching the lists still enables you to use the list panels to select principals and policies, so it doesn't affect how you use the SEAM Tool like the first option does. Also, even though caching doesn't enable you to see the changes of others, you are still able to see the latest list information based on your changes, since your changes update the lists both on the server and in the cache. And, if you want to update the cache to see the changes of others and get the latest copy of the lists, you can use the Refresh menu whenever you want to refresh the cache from the server.

▼ How to Start the SEAM Tool

1. Start the SEAM Tool by using the `gkadmin` command.

```
$ /usr/sbin/gkadmin
```

The Login window is displayed.



2. If you don't want to use the default values, specify new ones.

The Login window automatically fills in with default values. The default principal name is determined by taking your current identity from the `USER` environment variable and appending `/admin` to it (`username/admin`). The default Realm and Master KDC fields are selected from the `/etc/krb5/krb5.conf` file. If you ever want to go back to the default values, click Start Over.

Note – The administration operations that the principal name can perform are dictated by the Kerberos ACL file, `/etc/krb5/kadm5.ac1`. See “Using the SEAM Tool With Limited Kerberos Administration Privileges” on page 170 for information about limited privileges.

3. Enter a password for the specified principal name.

4. Click OK.

The following window is displayed.



Administering Principals

This section provides the step-by-step instructions to administer principals using the SEAM Tool. It also provides command-line equivalent examples, when available,

using the `kadmin` command after each procedure.

Administering Principals Task Map

TABLE 10-2 Administering Principals Task Map

Task	Description	For Instructions, Go To ...
View the List of Principals	View the list of principals by clicking the Principals tab.	"How to View the List of Principals" on page 147
View a Principal's Attributes	View a principal's attributes by selecting the Principal in the Principal List and clicking the Modify button.	"How to View a Principal's Attributes" on page 149
Create a New Principal	Create a new principal by clicking the Create New button in the Principal List panel.	"How to Create a New Principal" on page 151
Duplicate a Principal	Duplicate a principal by selecting the principal to duplicate in the Principal List and clicking the Duplicate button.	"How to Duplicate a Principal" on page 153
Modify a Principal	<p>Modify a principal by selecting the principal to modify in the Principal List and clicking the Modify button.</p> <p>Note that you cannot modify a principal's name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.</p>	"How to Modify a Principal" on page 153
Delete a Principal	Delete a principal by selecting the principal to delete in the Principal List and clicking the Delete button.	"How to Delete a Principal" on page 155
Set Up Defaults for Creating New Principals	Set up defaults for creating new principals by choosing Properties from the Edit menu.	"How to Set Up Defaults for Creating New Principals" on page 155
Modify the Kerberos Administration Privileges (kadm5 . ac1 File)	<p><i>Command line only.</i> The Kerberos administration privileges determine what operations a principal can perform on the Kerberos database, such as add and modify. You need to edit the <code>/etc/krb5/kadm5 . ac1</code> file to modify the Kerberos administration privileges for each principal.</p>	"How to Modify the Kerberos Administration Privileges" on page 156

Automating the Creation of New Principals

Even though the SEAM Tool provides ease-of-use, it doesn't provide a way to automate the creation of new principals. Automation is especially useful if you need to add ten or even 100 new principals in a short amount of time. However, by using the `kadmin.local` command in a Bourne shell script, you can do just that.

The following shell script line is an example of how to do this:

```
sed -e 's/^(.*)$/ank +needchange -pw \1 \1/' < princnames |  
time /usr/sbin/kadmin.local> /dev/null
```

This example has been split over two lines to make it more readable. The script reads in a file called `princnames` that contains principal names and their passwords and adds them to the Kerberos database. You would have to create the `princnames` file to contain a principal name and its password on each line, separated by one or more spaces. The `+needchange` option configures the principal so the user is prompted for a new password when logging in with the principal for the first time, which helps ensure that the passwords in the `princnames` file are not a security risk.

This is just one example. You can build more elaborate scripts, such as using the information in the name service to obtain the list of user names for the principal names. What you do and how you do it is up to your site needs and your scripting expertise.

▼ How to View the List of Principals

An example of the corresponding command-line equivalent follows this procedure.

- 1. If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 143 for details.

- 2. Click the Principals tab.**

The list of principals is displayed.



3. To display a specific principal or sublist of principals, enter a filter string in the Filter Pattern field and press return. If the filter succeeds, the list of principals matching the filter is displayed.

The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you enter the filter string `ge`, the filter mechanism will display only the principals with the `ge` string in them (for example, `george` or `edge`).

If you want to display the entire list of principals, click Clear Filter.

Example—Viewing the List of Principals (Command Line)

The following example uses the `list_principals` command of `kadmin` to list all the principals that match `test*`. Wildcards can be used with the `list_principals` command.

```
kadmin: list_principals test*
test1@EXAMPLE.COM
test2@EXAMPLE.COM
kadmin: quit
```

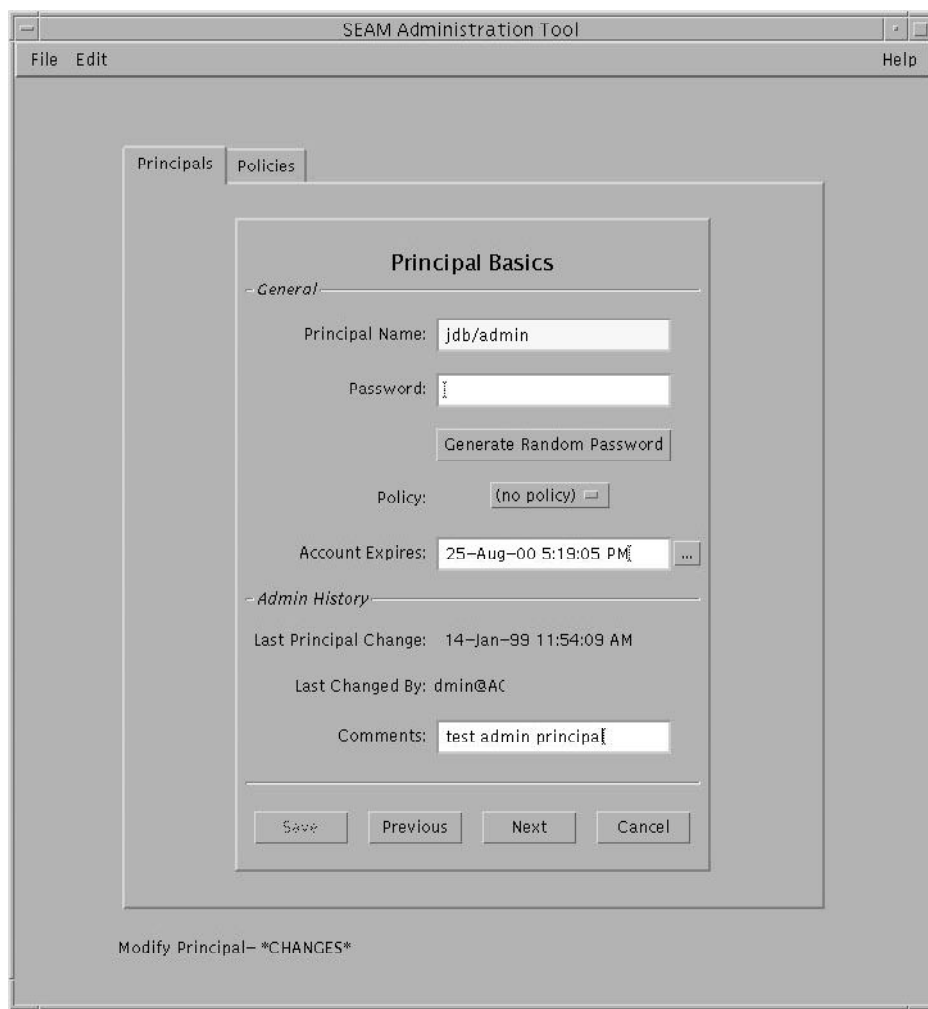
▼ How to View a Principal's Attributes

An example of the corresponding command-line equivalent follows this procedure.

- 1. If necessary, start the SEAM Tool.**
See “How to Start the SEAM Tool” on page 143 for details.
- 2. Click the Principals tab.**
- 3. Select the principal in the list that you want to view and click Modify.**
The Principal Basics panel containing some of the principal's attributes is displayed.
- 4. Continue to click Next to look at all the principal's attributes.**
Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to “SEAM Tool Panel Descriptions” on page 167 for all the principal attribute descriptions.
- 5. When you are finished viewing, click Cancel.**

Example—Viewing a Principal's Attributes

The following example shows the first window when viewing the `jdbc/admin` principal.



Example—Viewing a Principal’s Attributes (Command Line)

The following example uses the `get_principal` command of `kadmin` to view the attributes of the `jdb/admin` principal.

```
kadmin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM
Expiration date: Fri Aug 25 17:19:05 PDT 2000
Last password change: [never]
Password expiration date: Wed Apr 14 11:53:10 PDT 1999
```

```
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
Last modified: Thu Jan 14 11:54:09 PST 1999 (admin/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

▼ How to Create a New Principal

An example of the corresponding command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.

See “How to Start the SEAM Tool” on page 143 for details.

Note – If you are creating a new principal that may need a new policy, you should create the new policy before creating the new principal. Go to “How to Create a New Policy” on page 163.

2. Click the Principals tab.

3. Click New.

The Principal Basics panel containing some of the attributes for a principal is displayed.

4. Specify a principal name and password.

Both the principal name and password are mandatory.

5. Specify values for the principal’s attributes and continue to click Next to specify more attributes.

Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to “SEAM Tool Panel Descriptions” on page 167 for all the principal attribute descriptions.

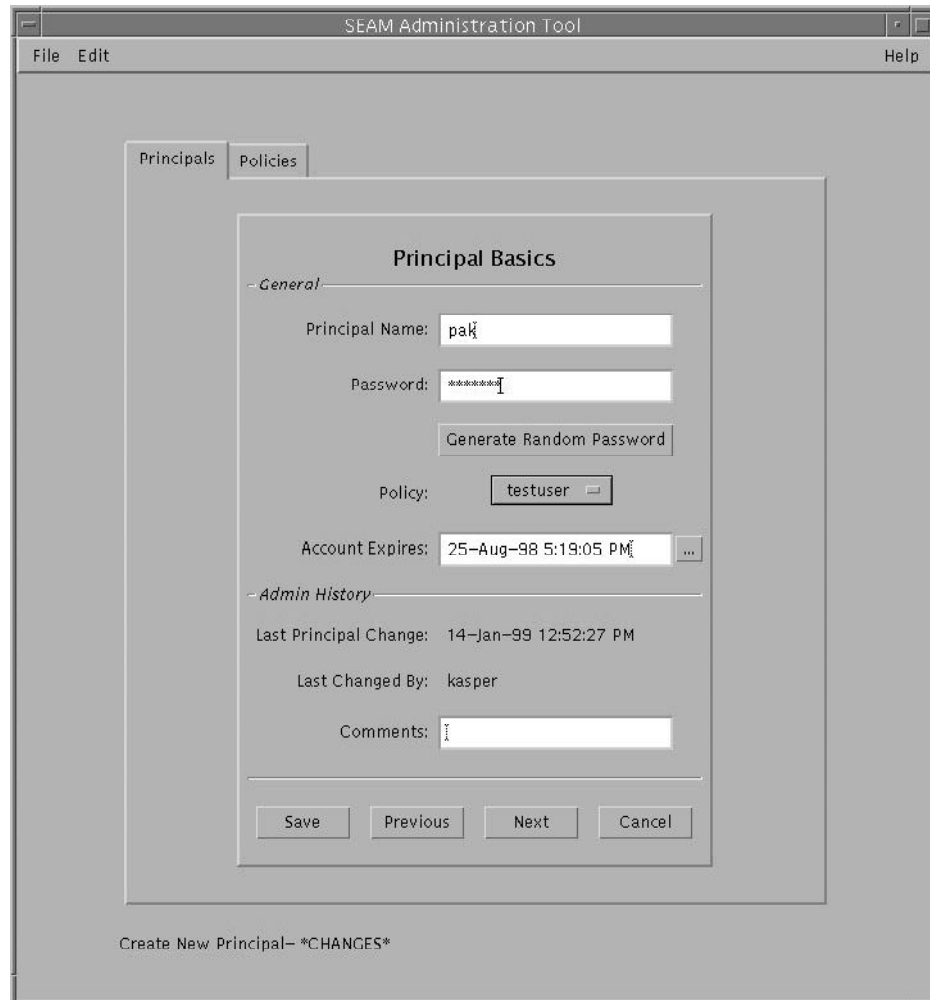
6. Click Save to save the principal, or click Done on the last panel.

7. If needed, set up Kerberos administration privileges for the new principal in the /etc/krb5/kadm5.acl file.

See “How to Modify the Kerberos Administration Privileges” on page 156 for more details.

Example—Creating a New Principal

The following example shows the Principal Basics panel when creating a new principal called pak. So far, the policy has been set to testuser.



The screenshot shows the SEAM Administration Tool window. The title bar reads "SEAM Administration Tool". The menu bar includes "File", "Edit", and "Help". The main window has two tabs: "Principals" and "Policies". The "Principals" tab is active, and the "Principal Basics" panel is displayed. The panel is divided into two sections: "General" and "Admin History".

General

- Principal Name: pak
- Password: *****
- Generate Random Password: [button]
- Policy: testuser
- Account Expires: 25-Aug-98 5:19:05 PM

Admin History

- Last Principal Change: 14-Jan-99 12:52:27 PM
- Last Changed By: kasper
- Comments: [text area]

At the bottom of the panel are four buttons: Save, Previous, Next, and Cancel. Below the panel, the text "Create New Principal- *CHANGES*" is visible.

Example—Creating a New Principal (Command Line)

The following example uses the `add_principal` command of `kadmin` to create a new principal called pak. The principal's policy is set to testuser.


```
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": <type the password>
Re-enter password for principal "pak@EXAMPLE.COM": <type the password again>
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

▼ How to Duplicate a Principal

This procedure explains how to use all or some of the attributes of an existing principal to create a new principal. There is no command-line equivalent for this procedure.

- 1. If necessary, start the SEAM Tool.**

See “How to Start the SEAM Tool” on page 143 for details.

- 2. Click the Principals tab.**

- 3. Select the principal in the list that you want to duplicate and click Duplicate.**

The Principal Basics panel is displayed. All the attributes of the selected principal are duplicated except for the Principal Name and Password fields, which are empty.

- 4. Specify a principal name and password.**

Both the principal name and password are mandatory. If you want to make an exact duplicate of the principal you selected, click Save and skip to the last step.

- 5. Specify different values for the principal’s attributes and continue to click Next to specify more attributes.**

Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to “SEAM Tool Panel Descriptions” on page 167 for all the principal attribute descriptions.

- 6. Click Save to save the principal, or click Done on the last panel.**

- 7. If needed, set up Kerberos administration privileges for the principal in `/etc/krb5/kadm5.acl` file.**

See “How to Modify the Kerberos Administration Privileges” on page 156 for more details.

▼ How to Modify a Principal

An example of the corresponding command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.

See “How to Start the SEAM Tool” on page 143 for details.

2. Click the Principals tab.

3. Select the principal in the list that you want to modify and click Modify.

The Principal Basics panel containing some of the attributes for the principal is displayed.

4. Modify the principal’s attributes and continue to click Next to modify more attributes.

Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to “SEAM Tool Panel Descriptions” on page 167 for all the principal attribute descriptions.

Note – You cannot modify a principal’s name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.

5. Click Save to save the principal, or click Done on the last panel.

6. Modify the Kerberos administration privileges for the principal in the `/etc/krb5/kadm5.acl` file.

See “How to Modify the Kerberos Administration Privileges” on page 156 for more details.

Example—Modifying a Principal’s Password (Command Line)

The following example uses the `change_password` command of `kadmin` to modify the password for the `jdb` principal. `change_password` does not let you change the password to one that is in the principal’s password history.

```
kadmin: change_password jdb
Enter password for principal "jdb": <type the new password>
Re-enter password for principal "jdb": <type the password again>
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

To modify other attributes for a principal, you must use the `modify_principal` command of `kadmin`.

▼ How to Delete a Principal

An example of the corresponding command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
See “How to Start the SEAM Tool” on page 143 for details.
2. **Click the Principals tab.**
3. **Specify the principal in the list that you want to delete and click Delete.**
After you confirm the deletion, the principal is deleted.
4. **Remove the principal from the Kerberos ACLs file, `/etc/krb5/kadm5.acl`.**
See “How to Modify the Kerberos Administration Privileges” on page 156 for more details.

Example—Deleting a Principal (Command Line)

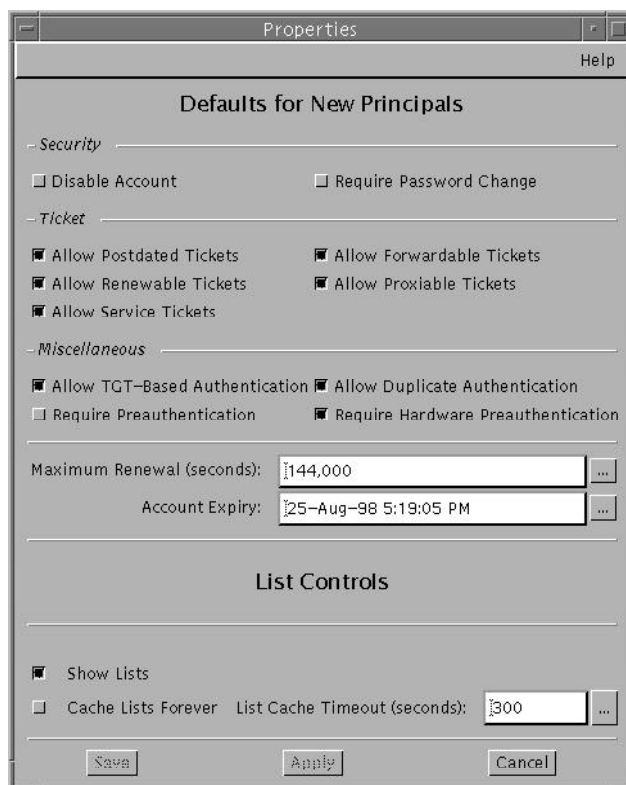
The following example uses the `delete_principal` command of `kadmin` to delete the `jdb` principal.

```
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
kadmin: quit
```

▼ How to Set Up Defaults for Creating New Principals

There is no command-line equivalent for this procedure.

1. **If necessary, start the SEAM Tool.**
See “How to Start the SEAM Tool” on page 143 for details.
2. **Choose Properties from the Edit Menu.**
The Properties window is displayed.



3. Select the defaults you want when you create new principals.

Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window.

4. Click Save.

▼ How to Modify the Kerberos Administration Privileges

Even though your site probably has a lot of user principals, you usually want only a few users to be able to administer the Kerberos database. Privileges to administer the Kerberos database are determined by the Kerberos Access Control List (ACL) file, `kadm5.ac1(4)`. The `kadm5.ac1` file enables you to allow or disallow privileges for individual principals, or you can use the `'*` wildcard in the principal name to specify privileges for groups of principals.

1. Become superuser on the master KDC.

2. Edit the `/etc/krb5/kadm5.acl` file.

An entry in the `kadm5.acl` file must have the following format:

```
principal privileges [principal_target]
```

principal

The principal to which the privileges are granted. Any part of the principal name can include the `*` wildcard, which is useful for providing the same privileges for a group of principals. For example, if you wanted to specify all principals with the `admin` instance, you would use `*/admin@realm`. Note that a common use of an `admin` instance is to grant separate privileges (such as administration access to the Kerberos database) to a separate Kerberos principal. For example, the user `jdb` might have a principal for his administrative use, called `jdb/admin`. This way, `jdb` obtains `jdb/admin` tickets only when he actually needs to use those privileges.

privileges

Specifies what operations can or cannot be performed by the principal. This is a string of one or more of the following list of characters or their uppercase counterparts. If the character is uppercase (or not specified), then the operation is disallowed. If the character is lowercase, then the operation is permitted.

- | | |
|--------|--|
| a | [Dis]allows the addition of principals or policies. |
| d | [Dis]allows the deletion of principals or policies. |
| m | [Dis]allows the modification of principals or policies. |
| c | [Dis]allows the changing of passwords for principals. |
| i | [Dis]allows inquiries to the database. |
| l | [Dis]allows the listing of principals or policies in the database. |
| x or * | Allows all privileges (<code>admcil</code>). |

principal_target

When a principal is specified in this field, the *privileges* apply to *principal* only when it operates on the *principal_target*. Any part of the principal name can include the `*` wildcard, which is useful to group principals.

Example—Modifying the Kerberos Administration Privileges

The following entry in the `kadm5.ac1` file gives any principal in the `EXAMPLE.COM` realm with the `admin` instance all the privileges on the database.

```
*/admin@EXAMPLE.COM *
```

The following entry in the `kadm5.ac1` file gives the `jdb@EXAMPLE.COM` principal the privilege to add, list, and inquire about any principal that has the `root` instance.

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```

Administering Policies

This section provides step-by-step instructions to administer policies using the SEAM Tool. It also provides command-line equivalent examples, when available, using the `kadmin` command after each procedure.

Administering Policies Task Map

TABLE 10-3 Administering Policies Task Map

Task	Description	For Instructions, Go To ...
View the List of Policies	View the list of policies by clicking the Policies tab.	"How to View the List of Policies" on page 159
View a Policy's Attributes	View a policy's attributes by selecting the Policy in the Policy List and clicking the Modify button.	"How to View a Policy's Attributes" on page 161
Create a New Policy	Create a new policy by clicking the Create New button in the Policy List panel.	"How to Create a New Policy" on page 163
Duplicate a Policy	Duplicate a policy by selecting the policy to duplicate in the Policy List and clicking the Duplicate button.	"How to Duplicate a Policy" on page 165

TABLE 10-3 Administering Policies Task Map (Continued)

Task	Description	For Instructions, Go To ...
Modify a Policy	Modify a policy by selecting the policy to modify in the Policy List and clicking the Modify button. Note that you cannot modify a policy's name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.	"How to Modify a Policy" on page 165
Delete a Policy	Delete a policy by selecting the policy to delete in the Policy List and clicking the Delete button.	"How to Delete a Policy" on page 166

▼ How to View the List of Policies

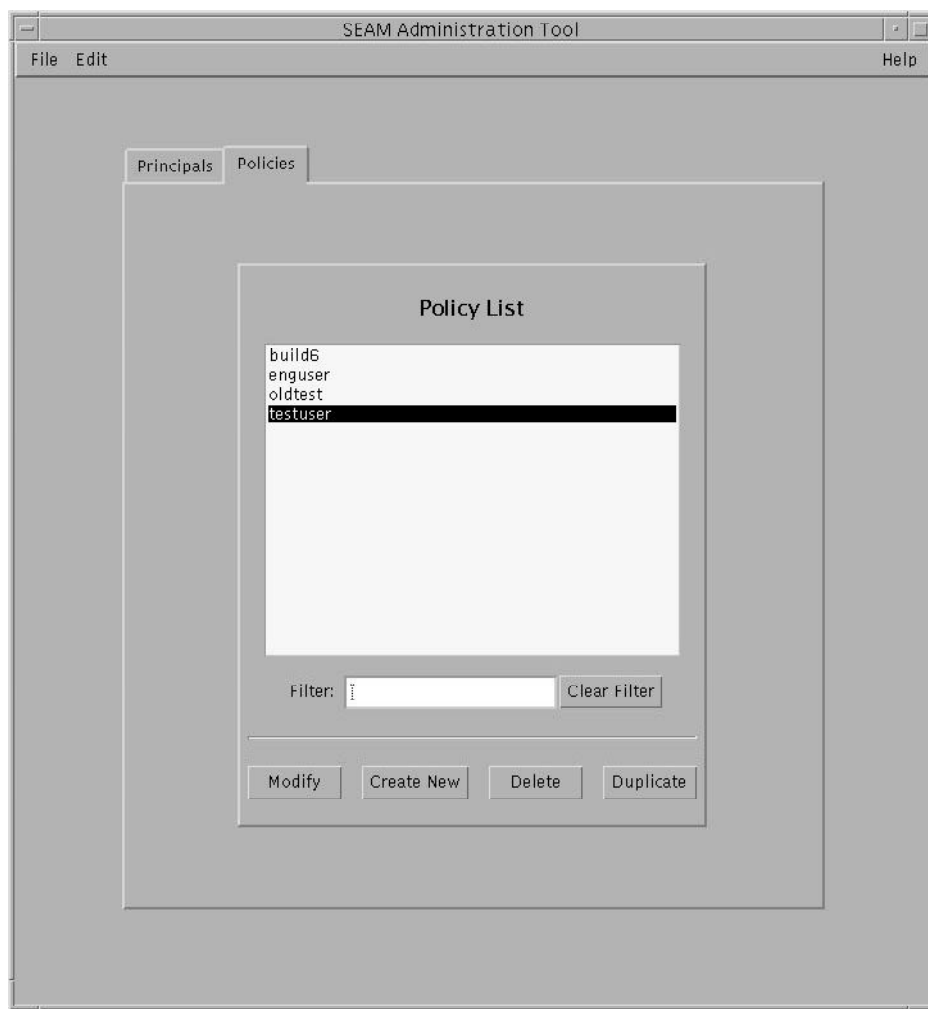
An example of the corresponding command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.

See "How to Start the SEAM Tool" on page 143 for details.

2. Click the Policies tab.

The list of policies is displayed.



- To display a specific principal or sublist of policies, enter a filter string in the Filter Pattern field and press return. If the filter succeeds, the list of policies matching the filter is displayed.**

The filter string must consist of one or more characters. And, because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you enter the filter string *ge*, the filter mechanism will display only the policies with the *ge* string in them (for example, *george* or *edge*).

If you want to display the entire list of policies, click Clear Filter.

Example—Viewing the List of Policies (Command Line)

The following example uses the `list_policies` command of `kadmin` to list all the policies that match `*user*`. Wildcards can be used with the `list_policies` command.

```
kadmin: list_policies *user*
testuser
enguser
kadmin: quit
```

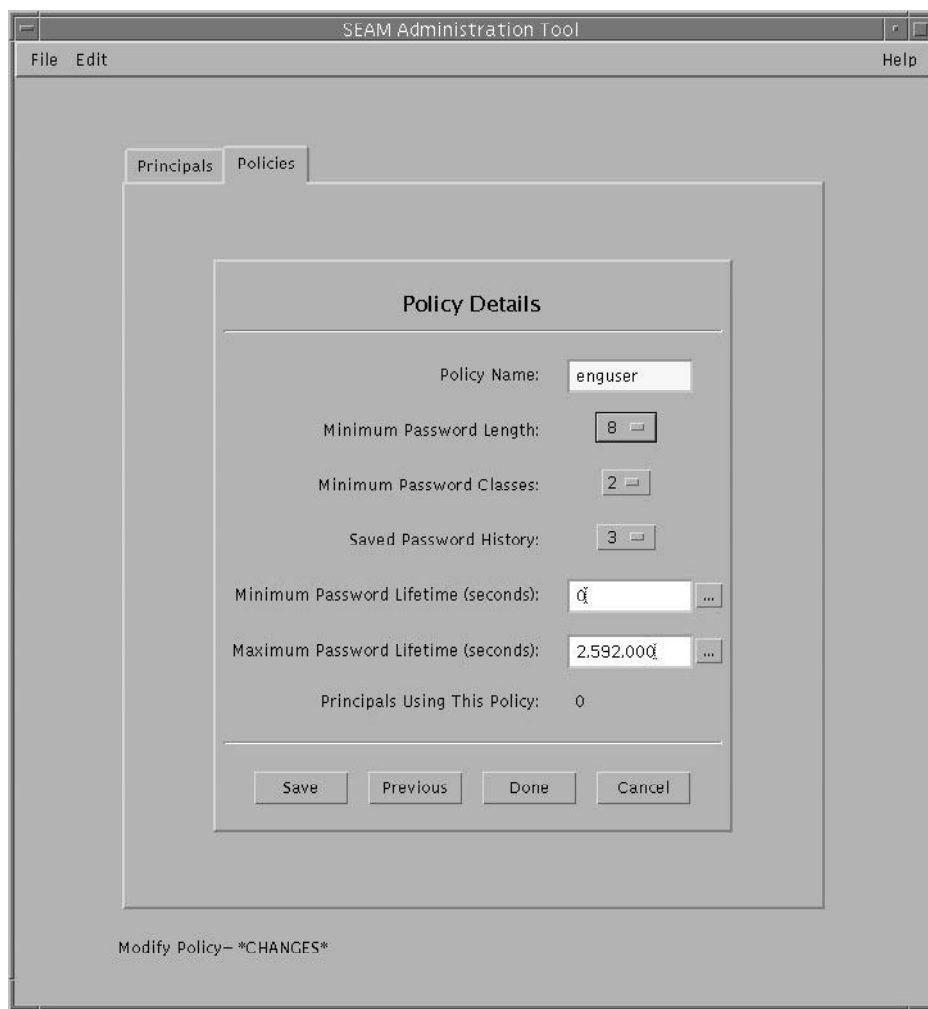
▼ How to View a Policy's Attributes

An example of the corresponding command-line equivalent follows this procedure.

- 1. If necessary, start the SEAM Tool.**
See "How to Start the SEAM Tool" on page 143 for details.
- 2. Click the Policies tab.**
- 3. Select the policy in the list that you want to view and click Modify.**
The Policy Details panel is displayed.
- 4. When you are finished viewing, click Cancel.**

Example—Viewing a Policy's Attributes

The following example shows the Policy Details panel when viewing the `test` policy.



Example—Viewing a Policy’s Attributes (Command Line)

The following example uses the `get_policy` command of `kadmin` to view the attributes of the `enguser` policy.

```
kadmin: get_policy enguser
Policy: enguser
Maximum password life: 2592000
Minimum password life: 0
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
```

```
Reference count: 0  
kadmin: quit
```

The reference count is the number of principals using that policy.

▼ How to Create a New Policy

An example of the corresponding command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.

See “How to Start the SEAM Tool” on page 143 for details.

2. Click the Policies tab.

3. Click New.

The Policy Details panel is displayed.

4. Specify a name for the policy in the Policy Name field.

The policy name is mandatory.

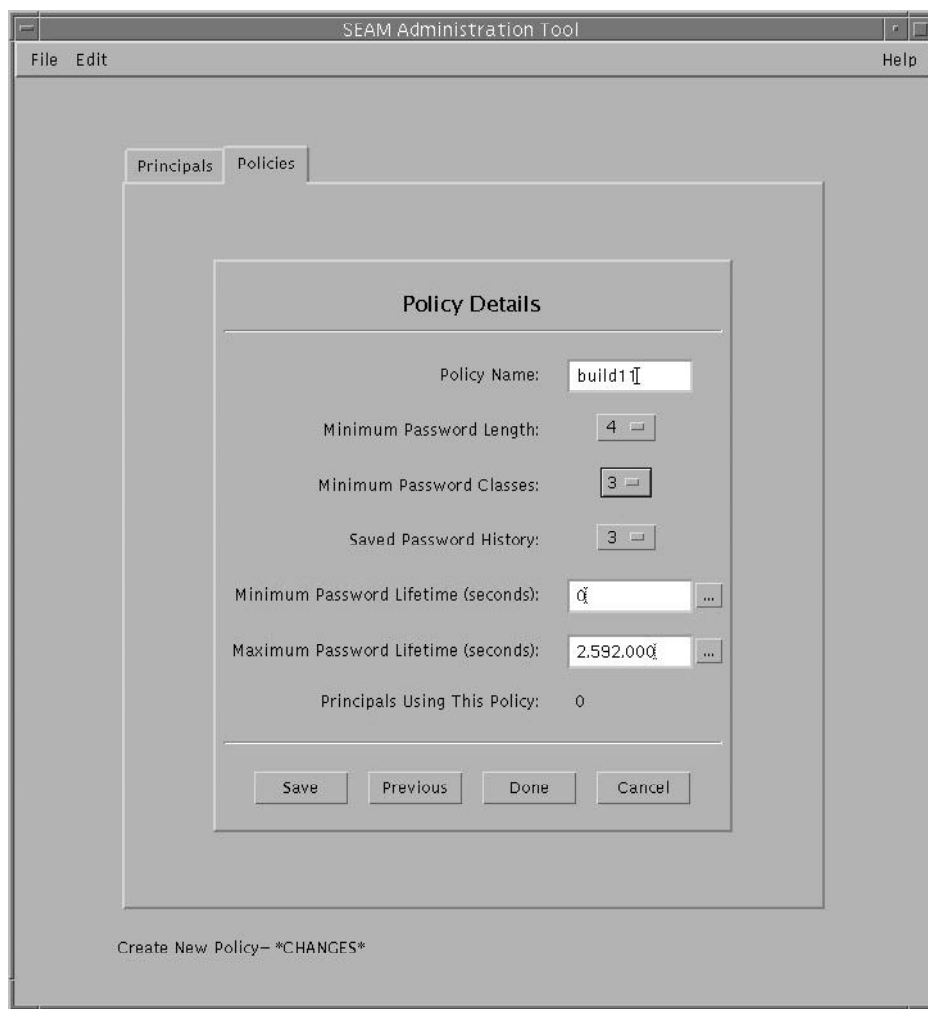
5. Specify values for the policy’s attributes.

Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to Table 10–7 for all the policy attribute descriptions.

6. Click Save to save the policy, or click Done.

Example—Creating a New Policy

The following example shows creating a new policy called `build11`. So far, the Minimum Password Classes has been changed to 3.



Example—Creating a New Policy (Command Line)

The following example uses the `add_policy` command of `kadmin` to create the `build11` policy that requires at least 3 character classes in a password.

```
$ kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```

▼ How to Duplicate a Policy

This procedure explains how to use all or some of the attributes of an existing policy to create a new policy. There is no command-line equivalent for this procedure.

- 1. If necessary, start the SEAM Tool.**

See “How to Start the SEAM Tool” on page 143 for details.

- 2. Click the Policies tab.**

- 3. Select the policy in the list that you want to duplicate and click Duplicate.**

The Policy Details panel is displayed. All the attributes of the selected policy are duplicated except for the Policy Name field, which is empty.

- 4. Specify a name for the duplicated policy in the Policy Name field.**

The policy name is mandatory. If you want to make an exact duplicate of the policy you selected, click Save and skip to the last step.

- 5. Specify different values for the policy’s attributes.**

Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to Table 10–7 for all the policy attribute descriptions.

- 6. Click Save to save the policy, or click Done.**

▼ How to Modify a Policy

An example of the corresponding command-line equivalent follows this procedure.

- 1. If necessary, start the SEAM Tool.**

See “How to Start the SEAM Tool” on page 143 for details.

- 2. Click the Policies tab.**

- 3. Select the policy in the list that you want to modify and click Modify.**

The Policy Details panel is displayed.

- 4. Modify the policy’s attributes.**

Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, go to Table 10–7 for all the policy attribute descriptions.

Note – You cannot modify a policy’s name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.

5. Click **Save** to save the policy, or click **Done**.

Example—Modifying a Policy (Command Line)

The following example uses the `modify_policy` command of `kadmin` to modify the minimum length of a password to five characters for the `build11` policy.

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ How to Delete a Policy

An example of the corresponding command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
See “How to Start the SEAM Tool” on page 143 for details.
2. **Click the Policies tab.**

Note – Before deleting a policy, you must cancel the policy from all principals currently using it (you need to modify the principals’ Policy attribute). The policy cannot be deleted if it is in use by any principal.

3. **Specify the policy in the list that you want to delete and click Delete.**
After you confirm the deletion, the policy is deleted.

Example—Deleting a Policy (Command Line)

The following example uses the `delete_policy` command of `kadmin` command to delete the `build11` policy.

```
kadmin: delete_policy build11
Are you sure you want to delete the policy "build11"? (yes/no): yes
kadmin: quit
```

Before deleting a policy, you must cancel the policy from all principals currently using it (you need to use the `modify_principal -policy` command of `kadmin` on the principals). The `delete_policy` command will fail if it is in use by a principal.

SEAM Tool Reference

This section provides reference information for the SEAM Tool.

SEAM Tool Panel Descriptions

This section provides descriptions for each of the principal and policy attributes that you can either specify or view in the SEAM Tool. The attributes are organized by the panel in which they are displayed.

TABLE 10-4 Principal Basic Panel Attributes

Attribute	Description
Principal Name	The name of the principal (the <i>primary/instance</i> part of a fully-qualified principal name). A principal is a unique identity to which the KDC can assign tickets. If you are modifying a principal, you cannot edit a principal's name.
Password	The password for the principal. You can use the Generate Random Password button to create a random password for the principal.
Policy	A menu of available policies for the principal.
Account Expires	The date and time on which the principal's account expires. When the account expires, the principal can no longer get a ticket-granting ticket (TGT) and may not be able to log in.
Last Principal Change	The date on which information for the principal was last modified. (Read-only)
Last Changed By	The name of the principal that last modified the account for this principal. (Read-only)
Comments	Comments related to the principal (for example, 'Temporary Account')

TABLE 10-5 Principal Details Panel Attributes

Attribute	Description
Last Success	The date and time when the principal last logged in successfully. (Read-only)

TABLE 10-5 Principal Details Panel Attributes (Continued)

Attribute	Description
Last Failure	The date and time when the last login failure for the principal occurred. (Read-only)
Failure Count	The number of times that there has been a login failure for the principal. (Read-only)
Last Password Change	The date and time when the principal's password was last changed. (Read-only)
Password Expires	The date and time when the principal's current password will expire.
Key Version	The key version number for the principal; this is normally changed only when a password has been compromised.
Maximum Lifetime (seconds)	The maximum length of time for which a ticket can be granted for the principal (without renewal).
Maximum Renewal (seconds)	The maximum length of time for which an existing ticket can be renewed for the principal.

TABLE 10-6 Principal Flags Panel Attributes

Attribute (Radio Buttons)	Description
Disable Account	When checked, prevents the principal from logging in. This is an easy way to freeze a principal account temporarily for any reason.
Require Password Change	When checked, expires the principal's current password, forcing the user to use the <code>kpasswd</code> command to create a new password. This is useful if there is a security breach and you need to make sure that old passwords are replaced.
Allow Postdated Tickets	When checked, allows the principal to obtain postdated tickets. For example, you may need to use postdated tickets for cron jobs that must run after hours and can't obtain tickets in advance because of short ticket lifetimes.
Allow Forwardable Tickets	When checked, allows the principal to obtain forwardable tickets. Forwardable tickets are tickets that are forwarded to the remote host to provide a single-sign-on session. For example, if you are using forwardable tickets and you authenticate yourself through <code>ftp</code> or <code>rsh</code> , other services, such as NFS services, are available without your being prompted for another password.
Allow Renewable Tickets	When checked, allows the principal to obtain renewable tickets. A principal can automatically extend the expiration date or time of a ticket that is renewable (rather than having to get a new ticket after the first one expires). Currently, the NFS service is the only service that can renew tickets.

TABLE 10-6 Principal Flags Panel Attributes (Continued)

Attribute (Radio Buttons)	Description
Allow Proxiable Tickets	<p>When checked, allows the principal to obtain proxiable tickets.</p> <p>A proxiable ticket is a ticket that can be used by a service on behalf of a client to perform an operation for the client. With a proxiable ticket, a service can take on the identity of a client and obtain a ticket for another service, but it cannot obtain a ticket-granting ticket.</p>
Allow Service Tickets	<p>When checked, allows service tickets to be issued for the principal.</p> <p>You should not allow service tickets to be issued for the <code>kadmind/hostname</code> and <code>changepw/hostname</code> principals. This ensures that these principals can only update the KDC database.</p>
Allow TGT-Based Authentication	<p>When checked, allows the service principal to provide services to another principal. More specifically, it allows the KDC to issue a service ticket for the service principal.</p> <p>This attribute is valid only for service principals. When not checked, service tickets cannot be issued for the service principal.</p>
Allow Duplicate Authentication	<p>When checked, allows the user principal to obtain service tickets for other user principals.</p> <p>This attribute is valid only for user principals. When not checked, the user principal can still obtain service tickets for service principals, but not for other user principals.</p>
Required Preauthentication	<p>When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until it can authenticate (through software) that it is really the principal requesting the TGT. This preauthentication is usually done through an extra password, for example, from a DES card.</p> <p>When not checked, the KDC does not need to preauthenticate the principal before it sends a requested TGT to it.</p>
Required Hardware Authentication	<p>When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until it can authenticate (through hardware) that it is really the principal requesting the TGT. Hardware preauthentication can be something like a Java ring reader.</p> <p>When not checked, the KDC does not need to preauthenticate the principal before it sends a requested TGT to it.</p>

TABLE 10-7 Policy Basics Panel Attributes

Attribute	Description
Policy Name	<p>The name of the policy. A policy is a set of rules governing a principal's password and tickets.</p> <p>If you are modifying a policy, you cannot edit a policy's name.</p>

TABLE 10-7 Policy Basics Panel Attributes (Continued)

Attribute	Description
Minimum Password Length	The minimum length for the principal's password.
Minimum Password Classes	The minimum number of different character types required in the principal's password. For example, a minimum classes value of 2 means that the password must have at least two different character types, such as letters and numbers (hi2mom). A value of 3 means that the password must have at least three different character types, such as letters, numbers, and punctuation (hi2mom!). And so on. A value of 1 basically sets no restriction on the number of password character types.
Saved Password History	The number of previous passwords that have been used by the principal and cannot be reused.
Minimum Password Lifetime (seconds)	The minimum time that the password must be used before it can be changed.
Maximum Password Lifetime (seconds)	The maximum time that the password can be used before it must be changed.
Principals Using This Policy	The number of principals to which this policy currently applies. (Read-only)

Using the SEAM Tool With Limited Kerberos Administration Privileges

All the features of the SEAM Administration Tool are available if your `admin` principal has all the privileges to administer the Kerberos database. But it is possible to have limited privileges, such as being allowed to view only the list of principals or to change a principal's password. With limited Kerberos administration privileges, you can still use the SEAM Administration Tool; however, various parts of the SEAM Tool will change based on what Kerberos administration privileges you do not have. Table 10-8 shows how the SEAM Tool changes based on your Kerberos administration privileges.

The most visual change to the SEAM Tool is when you don't have the list privilege. Without the list privilege, the List panels do not display the list of principals and policies for you to manipulate. Instead, you must use the Name field in the List panels to specify a principal or policy you want to work on.

If you log on to the SEAM Tool and you don't have sufficient privileges to perform useful tasks with it, the following message will display and you will be sent back to the Login window:

```
Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.
```

To change the privileges for a principal to administer the Kerberos database, go to “How to Modify the Kerberos Administration Privileges” on page 156.

TABLE 10–8 Using SEAM Tool With Limited Kerberos Administration Privileges

If You Don't Have This Privilege ...	Then the SEAM Tool Changes as Follows ...
a (add)	The Create New and Duplicate buttons are not available in the Principal and Policy List panels. Without the add privilege, you can't create new or duplicate principal or policies.
d (delete)	The Delete button is not available in the Principal and Policy List panels. Without the delete privilege, you can't delete principal or policies.
m (modify)	<p>The Modify button is not available in the Principal and Policy List panels. Without the modify privilege, you can't modify principal or policies.</p> <p>Also, with the Modify button unavailable, you can't modify a principal's password, even if you have the change password privilege.</p>
c (change password)	<p>The Password field in the Principal Basics panel is read-only and cannot be changed. Without the change password privilege, you can't modify a principal's password.</p> <p>Note that even if you have the change password privilege, you must also have the modify privilege to change a principal's password.</p>
i (inquiry to database)	<p>The Modify and Duplicate buttons are not available in the Principal and Policy List panels. Without the inquiry privilege, you can't modify or duplicate a principal or policy.</p> <p>Also, with the Modify button unavailable, you can't modify a principal's password, even if you have the change password privilege.</p>
l (list)	The list of principals and policies in the List panels are unavailable. Without the list privilege, you must use the Name field in the List panels to specify the principal or policy you want to work on.

Administering Keytabs

Every host providing a service must have a local file, called a *keytab* (short for key table), containing the principal for the appropriate service, called a *service key*. A service key is used by a service to authenticate itself to the KDC and is known only by Kerberos and the service itself. For example, if you have a Kerberized NFS server, that server must have a keytab that contains its `nfs` service principal.

To add a service key to a keytab, you add the appropriate service principal to a host's keytab by using the `ktadd` command of `kadmin`. And, because you are adding a service principal to a keytab, the principal must already exist in the Kerberos database so `kadmin` can verify its existence. On the master KDC, the keytab file is located at `/etc/krb5/kadm5.keytab`, by default. On application servers providing Kerberized services, the keytab file is located at `/etc/krb5/krb5.keytab`, by default.

A keytab is analogous to a user's password. Just as it is important for users to protect their passwords, it is equally important for application servers to protect their keytabs. You should always store keytabs on a local disk, and make them readable only by root, and you should never send a keytab over an unsecured network.

There is also a special instance to add a `root` principal to a host's keytab. Basically, if you want a user on the SEAM client to mount Kerberized NFS file systems using Kerberos authentication automatically, you must add the client's `root` principal to the client's keytab. Otherwise, users must use the `kinit` command as root to obtain credentials for the client's `root` principal whenever they want to mount a Kerberized NFS file system, even when using the automounter. See "Setting Up Root Authentication to Mount NFS File Systems" on page 110 for detailed information.

Note – When setting up a master KDC, you need to add the `kadmin` and `changepw` principals to the `kadm5.keytab` file, so the KDC can decrypt administrators' Kerberos tickets to determine whether or not it should give them access to the database.

Another command that you can use to administer keytabs with is the `ktutil` command. `ktutil` is an interactive command-line interface utility that enables you to manage a local host's keytab without having Kerberos administration privileges, because `ktutil` doesn't interact with the Kerberos database like `kadmin` does. So, after a principal is added to a keytab, you can use `ktutil` to view the keylist in a keytab or to temporarily disable authentication for a service.

Administering Keytabs Task Map

TABLE 10-9 Administering Keytabs Task Map

Task	Description	For Instructions, Go To ...
Add a Service Principal to a Keytab	Use the <code>ktadd</code> command of <code>kadmin</code> to add a service principal to a keytab.	“How to Add a Service Principal to a Keytab” on page 173
Remove a Service Principal from a Keytab	Use the <code>ktremove</code> command of <code>kadmin</code> to remove a service from a keytab.	“How to Remove a Service Principal From a Keytab” on page 174
Display the Keylist (Principals) in a Keytab	Use the <code>ktutil</code> command to display the keylist in a keytab.	“How to Display the Keylist (Principals) in a Keytab” on page 175
Temporarily Disable Authentication for a Service on a Host	This procedure is a quick way to temporarily disable authentication for a service on a host without having to have <code>kadmin</code> privileges. Before using <code>ktutil</code> to delete the service principal from the server’s keytab, copy the original keytab to a temporary location. When you want to enable the service again, copy the original keytab back.	“How to Temporarily Disable Authentication for a Service on a Host” on page 176

▼ How to Add a Service Principal to a Keytab

- 1. Make sure the principal already exists in the Kerberos database.**
See “How to View the List of Principals” on page 147 for more information.
- 2. Become superuser on the host that needs a principal added to its keytab.**
- 3. Start the `kadmin` command.**

```
# /usr/sbin/kadmin
```

- 4. Add a principal to a keytab by using the `ktadd` command.**

```
kadmin: ktadd [-k keytab] [-q] [principal | -glob principal_exp]
```

<code>-k <i>keytab</i></code>	Specifies the keytab file. By default, <code>/etc/krb5/krb5.keytab</code> is used.
<code>-q</code>	Displays less verbose information.
<code><i>principal</i></code>	Principal to be added to the keytab. You can add the following service principals: <code>host</code> , <code>root</code> , <code>nfs</code> , and <code>ftp</code> .

`-glob principal_exp`

All principals matching the principal expression are added to the keytab. The rules for principal expression are the same as for the `list_principals` command of `kadmin`.

5. Quit the `kadmin` command.

```
kadmin: quit
```

Example—Adding a Service Principal to a Keytab

The following example adds the `kadmin/admin` and `kadmin/changepw` principals to a master KDC's keytab. For this example, the keytab file must be the one specified in the `kdc.conf` file.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/admin kadmin/changepw
Entry for principal kadmin/admin@EXAMPLE.COM with kvno 3, encryption type DES-CBC-CRC
  added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@EXAMPLE.COM with kvno 3, encryption type DES-CBC-CRC
  added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

The following example adds `denver`'s host principal to `denver`'s keytab file, so `denver`'s network services can be authenticated by the KDC.

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver@example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver@example.com@EXAMPLE.COM with kvno 2,
  encryption type DES-CBC-CRC added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ How to Remove a Service Principal From a Keytab

1. Become superuser on the host with a service principal that must be removed from its keytab.
2. Start the `kadmin` command.

```
# /usr/sbin/kadmin
```

3. Optional. To display the current list of principals (keys) in the keytab, use the `ktutil` command.

See “How to Display the Keylist (Principals) in a Keytab” on page 175 for detailed instructions.

4. Remove a principal from a keytab by using the `ktremove` command.

```
kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
```

<code>-k <i>keytab</i></code>	Specifies the keytab file. By default, <code>/etc/krb5/krb5.keytab</code> is used.
<code>-q</code>	Displays less verbose information.
<code><i>principal</i></code>	Principal to be removed from the keytab.
<code><i>kvno</i></code>	Removes all entries for the specified principal whose <code>kvno</code> (key version number) matches <code><i>kvno</i></code> .
<code>all</code>	Removes all entries for the specified principal.
<code>old</code>	Removes all entries for the specified principal except those with the highest <code>kvno</code> .

5. Quit the `kadmin` command.

```
kadmin: quit
```

Example—Removing a Service Principal From a Keytab

The following example removes denver's host principal from denver's keytab file.

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
       removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ How to Display the Keylist (Principals) in a Keytab

1. Become superuser on the host with the keytab.

Note – Although you can create keytabs owned by other users, the default location for the keytab requires root ownership.

2. Start the `ktutil` command.

```
# /usr/bin/ktutil
```

3. Read the keytab into the keylist buffer by using the `read_kt` command.

```
ktutil: read_kt keytab
```

4. **Display the keylist buffer by using the `list` command.**

```
ktutil: list
```

The current keylist buffer is displayed.

5. **Quit the `ktutil` command.**

```
ktutil: quit
```

Example—Displaying the Keylist (Principals) in a Keytab

The following example displays the keylist in the `/etc/krb5/krb5.keytab` file on the `denver` host.

```
denver # /usr/bin/ktutil
ktutil: read_kt /etc/krb5/krb5.keytab
ktutil: list
slot KVNO Principal
-----
1      5 host/denver@EXAMPLE.COM
ktutil: quit
```

▼ How to Temporarily Disable Authentication for a Service on a Host

You may find instances when you need to temporarily disable the authentication mechanism for a service, such as `rlogin` or `ftp`, on a network application server. For example, you may want to stop users from logging into a system while performing maintenance procedures. The `ktutil` command enables you to do this by removing the service principal from the server's keytab, without requiring `kadmin` privileges. To enable authentication again, all you need to do is copy the original keytab that you saved back to its original location.

Note – Most services are set up by default to require authentication to work. If this is not the case, the service will still work even if you disable authentication for the service.

1. **Become superuser on the host with the keytab.**

Note – Although you can create keytabs owned by other users, the default location for the keytab requires root ownership.

2. **Save the current keytab to a temporary file.**

3. Start the `ktutil` command.

```
# /usr/bin/ktutil
```

4. Read the keytab into the keylist buffer by using the `read_kt` command.

```
ktutil: read_kt keytab
```

5. Display the keylist buffer by using the `list` command.

```
ktutil: list
```

The current keylist buffer is displayed. Note the slot number for the service you want to disable.

6. To temporarily disable a host's service, remove the specific service principal from the keylist buffer by using the `delete_entry` command.

```
ktutil: delete_entry slot_number
```

slot_number

The slot number of the service principal to be deleted, which is displayed by the `list` command.

7. Write the keylist buffer to the keytab by using the `write_kt` command.

```
ktutil: write_kt keytab
```

8. Quit the `ktutil` command.

```
ktutil: quit
```

9. When you want to enable the service again, copy the temporary (original) keytab back to its original location.

Example—Temporarily Disabling a Service on a Host

The following example temporarily disables the host service on the `denver` host. To enable the host service back on `denver`, you would copy the `krb5.keytab.temp` file to the `/etc/krb5/krb5.keytab` file.

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.temp
denver # /usr/bin/ktutil
      ktutil:read_kt /etc/krb5/krb5.keytab
      ktutil:list
slot KVNO Principal
-----
1      8 root/denver@EXAMPLE.COM
2      5 host/denver@EXAMPLE.COM
      ktutil:delete_entry 2
      ktutil:list
```

slot KVNO Principal

```
-----  
1      8 root/denver@EXAMPLE.COM  
ktutil:write_kt /etc/krb5/krb5.keytab  
ktutil: quit
```

Using SEAM

This chapter is intended for anyone on a system that already has SEAM installed on it. It explains how to use the “Kerberized” commands provided by SEAM: `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`. You should already be familiar with these commands (in their non-Kerberized versions) before reading about them here. You’ll find that the Kerberized and non-Kerberized versions are substantially the same. In many cases, in fact, you can use these commands without ever knowing or caring that they are Kerberized. The differences lie in using features that take advantage of Kerberos (for example, forwarding a ticket when you use `rlogin`).

Because this chapter is intended for the general reader, it includes information on tickets: getting, viewing, and destroying them. It also includes information on choosing or changing a Kerberos password.

For an overview of SEAM, see Chapter 6.

This is a list of topics covered in this chapter:

- “Do You Need to Worry About Tickets?” on page 180
- “How to Create a Ticket” on page 180
- “How to View Tickets” on page 181
- “How to Destroy Tickets” on page 182
- “Changing Your Password” on page 184
- “Advice on Choosing a Password” on page 183

Ticket Management

This section explains how to obtain, view, and destroy tickets. For an introduction to tickets, see “How SEAM Works” on page 74.

Do You Need to Worry About Tickets?

With SEAM installed, Kerberos is built into the `login` command, and you will get tickets automatically when you log in.

Most of the Kerberized commands also automatically destroy your tickets when they exit. However, you might want to explicitly destroy your Kerberos tickets with `kdestroy` when you are through with them, just to be sure. See “How to Destroy Tickets” on page 182 for more information on `kdestroy`.

For information on ticket lifetimes, see “Ticket Lifetimes” on page 193.

▼ How to Create a Ticket

Normally a ticket is created automatically when you log in and you need not do anything special to obtain one. However, you might need to create a ticket if your ticket expires.

To create a ticket, use the `kinit` command.

```
% /usr/bin/kinit
```

`kinit` prompts you for your password. For the full syntax of the `kinit` command, see the `kinit(1)` man page.

Example — Creating a Ticket

This example shows a user, `jennifer`, creating a ticket on her own system:

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM:      <enter password>
```

Here the user `david` creates a ticket good for three hours with the `-l` option:

```
% kinit -l 3h david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:           <enter password>
```

This example shows `david` creating a forwardable ticket (with `-f`) for himself. With this forwardable ticket, he can (for example) log in to a second system, and then `telnet` to a third system.

```
% kinit -f david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:           <enter password>
```

For more on how forwarding tickets works, see “Types of Tickets” on page 192.

▼ How to View Tickets

Not all tickets are alike. One ticket might be, for example, *forwardable*; another might be *postdated*; while a third might be both. You can see which tickets you have, and what their attributes are, by using the `klist` command with the `-f` option:

```
% /usr/bin/klist -f
```

The following symbols indicate the attributes associated with each ticket, as displayed by `klist`:

F	Forwardable
f	Forwarded
P	Proxiabile
p	Proxy
D	Postdateable
d	Postdated
R	Renewable
I	Initial
i	Invalid

“Types of Tickets” on page 192 describes the various attributes a ticket can have.

Example — Viewing Tickets

This example shows that the user `jennifer` has an *initial* ticket, which is *forwardable* (F) and *postdated* (d), but not yet validated (i):

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: jennifer@ENG.EXAMPLE.COM

Valid starting          Expires                Service principal
09 Mar 99 15:09:51     09 Mar 99 21:09:51   nfs/EXAMPLE.SUN.COM@EXAMPLE.SUN.COM
    renew until 10 Mar 99 15:12:51, Flags: Fdi
```

The example below shows that the user `david` has two tickets that were *forwarded* (f) to his host from another host. The tickets are also (re)*forwardable* (F):

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting          Expires                Service principal
07 Mar 99 06:09:51    09 Mar 99 23:33:51    host/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 99 17:09:51, Flags: fF

Valid starting          Expires                Service principal
08 Mar 99 08:09:51    09 Mar 99 12:54:51    nfs/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 99 15:22:51, Flags: fF
```

▼ How to Destroy Tickets

Tickets are generally destroyed automatically when the commands that created them exit; however, you might want to explicitly destroy your Kerberos tickets when you are through with them, just to be sure. Tickets can be stolen, and if this happens, the person who has them can use them until they expire (although stolen tickets must be decrypted).

To destroy your tickets, use the `kdestroy` command.

```
% /usr/bin/kdestroy
```

`kdestroy` destroys *all* your tickets. You cannot use it to selectively destroy a particular ticket.

If you are going to be away from your system and are concerned about an intruder using your permissions, you should either use `kdestroy` or a screensaver that locks the screen.

Note – One way to help ensure that tickets are always destroyed is to add the `kdestroy` command to the `.logout` file in your home directory.

In cases where the PAM module has been configured (the default and usual case), tickets are destroyed automatically upon logout, so adding a call to `kdestroy` to your `.login` file is not necessary. However, if the PAM module has not been configured, or if you don't know whether it has or not, you might want to add `kdestroy` to your `.login` file to be sure that tickets are destroyed when you exit your system.

Password Management

With SEAM installed, you now have two passwords: your regular Solaris password, and a Kerberos password. You can make both passwords the same or they can be different.

Non-Kerberized commands, such as `login`, are typically set up through PAM to authenticate with both Kerberos and UNIX. If you have different passwords, you must provide both passwords to log on with the appropriate authentication. However, if both passwords are the same, the first password you enter for UNIX is also accepted by Kerberos.

Unfortunately, using the same password for both can compromise security. That is, if someone discovers your Kerberos password, then your UNIX password is no longer a secret. However, using the same passwords for UNIX and Kerberos is still more secure than a site without Kerberos, because passwords in a Kerberos environment are not sent across the network. Usually, your site will have a policy to help you determine your options.

Your Kerberos password is the only way Kerberos has of verifying your identity. If someone discovers your Kerberos password, Kerberos security becomes meaningless, for that person can masquerade as you — send email that comes from "you," read, edit, or delete your files, or log into other hosts as you — and no one will be able to tell the difference. For this reason, it is vital that you choose a good password and keep it secret. You should *never* reveal your password to anyone else, not even your system administrator. Additionally, you should change your password frequently, particularly any time you believe someone might have discovered it.

Advice on Choosing a Password

Your password can include almost any character you can type (the main exceptions being control keys and the Return key). A good password is one that you can remember readily, but which no one else can easily guess. Examples of bad passwords include:

- Words that can be found in a dictionary
- Any common or popular name
- The name of a famous person or character
- Your name or username in any form (for example: backward, repeated twice, and so forth.)
- A spouse's, child's, or pet's name
- Your birth date or a relative's birth date

- Social Security number, driver's license number, passport number, or similar identifying number
- Any sample password that appears in this or any other manual

A good password is at least eight characters long. Moreover, a password should include a mix of characters, such as upper- and lower-case letters, numbers, and punctuation marks. Examples of passwords that would be good if they didn't appear in this manual include:

- Acronyms, such as "I2LMHinSF" (recalled as "I too left my heart in San Francisco")
- Easy-to-pronounce nonsense words, like "WumpaBun" or "WangDangdoodle!"
- Deliberately misspelled phrases, such as "6o'cluck" or "RrriotGrrrlsRrrule!"



Caution – Don't use these examples. Passwords that appear in manuals are the first ones an intruder will try.

Changing Your Password

You can change your Kerberos password in two ways:

- With the usual UNIX `passwd` command. With SEAM installed, the Solaris `passwd` command also automatically prompts for a new Kerberos password.

The advantage of using `passwd` instead of `kpasswd` is that you can set both passwords (UNIX and Kerberos) at the same time. However, generally you do not *have* to change both passwords with `passwd`; often you can change only your UNIX password and leave the Kerberos password untouched, or vice-versa.

Note – The behavior of `passwd` depends on how the PAM module is configured. You may be required to change both passwords in some configurations. For some sites the UNIX password must be changed, while others require the Kerberos password to change.

- With the `kpasswd` command. `kpasswd` is very similar to `passwd`. One difference is that `kpasswd` changes only Kerberos passwords — you must use `passwd` if you want to change your UNIX password.

Another difference is that `kpasswd` can change a password for a Kerberos principal that is not a valid UNIX user. For example, `david/admin` is a Kerberos principal, but not an actual UNIX user, so you must use `kpasswd` instead of `passwd`.

After you change your password, it takes some time for the change to propagate through a system (especially over a large network). Depending on how your system is

set up, this might be anywhere from a few minutes to an hour or more. If you need to get new Kerberos tickets shortly after changing your password, try the new password first. If the new password doesn't work, try again using the old one.

Kerberos V5 allows system administrators to set criteria about allowable passwords for each user. Such criteria is defined by the *policy* set for each user (or by a default policy)— see “Administering Policies” on page 158 for more on policies. For example, suppose that jennifer's policy (call it `jenpol`) mandates that passwords be at least eight letters long and include a mix of at least two kinds of characters. `kpasswd` will therefore reject an attempt to use "sloth" as a password:

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:      <jennifer enters her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <jennifer enters 'sloth'>
New password (again):      <jennifer re-enters 'sloth'>
kpasswd: New password is too short.
Please choose a password which is at least 4 characters long.
```

Here jennifer uses “slothrop49” as a password. ‘slothrop49’ meets the criteria, because it is over eight letters long and contains two different kinds of characters (numbers and lowercase letters):

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:      <jennifer enters her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <jennifer enters 'slothrop49'>
New password (again):      <jennifer re-enters 'slothrop49'>
Kerberos password changed.
```

Examples — Changing Your Password

The following example shows david changing both his UNIX and Kerberos passwords with `passwd`.

```
% passwd
passwd: Changing password for david
Enter login (NIS+) password:      <enter the current UNIX password>
New password:                      <enter the new UNIX password>
Re-enter password:                  <confirm the new UNIX password>
Old KRB5 password:                  <enter the current Kerberos password>
```

```
New KRB5 password:                <enter the new Kerberos password>
Re-enter new KRB5 password:       <confirm the new Kerberos password>
```

In the above example `passwd` asks for both the UNIX and Kerberos password; however, if `try_first_pass` is set in the PAM module, the Kerberos password is automatically set to be the same as the UNIX password. (That is the default configuration.) In that case, `david` must use `kpasswd` to set his Kerberos password to something else, as shown next.

This example shows him changing only his Kerberos password with `kpasswd`:

```
% kpasswd
kpasswd: Changing password for david@ENG.EXAMPLE.COM.
Old password:                <enter the current Kerberos password>
New password:                <enter the new Kerberos password>
New password (again):       <confirm the new Kerberos password>
Kerberos password changed.
```

In this example, `david` changes the password for the Kerberos principal `david/admin` (which is not a valid UNIX user). To do this he must use `kpasswd`.

```
% kpasswd david/admin
kpasswd: Changing password for david/admin.
Old password:                <enter the current Kerberos password>
New password:                <enter the new Kerberos password>
New password (again):       <confirm the new Kerberos password>
Kerberos password changed.
```

SEAM Reference

This chapter lists many of the files, commands, and daemons that are part of the SEAM product. In addition, this chapter provides detailed information about how the Kerberos authentication system works.

This is a list of the reference information in this chapter.

- “SEAM Files” on page 187
- “SEAM Commands” on page 189
- “SEAM Daemons” on page 190
- “SEAM Terminology” on page 190
- “How the Authentication System Works” on page 196
- “Gaining Access to a Service Using SEAM” on page 196

SEAM Files

TABLE 12-1 SEAM Files

File Name	Description
<code>~/ .gkadmin</code>	Default values for creating new principals in the SEAM Administration Tool
<code>~/ .k5login</code>	List of principals to grant access to a Kerberos account
<code>/etc/init.d/kdc</code>	init script to start or stop krb5kdc
<code>/etc/init.d/kdc.master</code>	init script to start or stop kadmind

TABLE 12-1 SEAM Files (Continued)

File Name	Description
/etc/krb5/kadm5.acl	Kerberos access control list file; includes principal names of KDC administrators and their Kerberos administration privileges
/etc/krb5/kadm5.keytab	Keytab for kadmin service on master KDC
/etc/krb5/kdc.conf	KDC configuration file
/etc/krb5/kpropd.acl	Kerberos database propagation configuration file
/etc/krb5/krb5.conf	Kerberos realm configuration file
/etc/krb5/krb5.keytab	Keytab for network application servers
/etc/krb5/warn.conf	Kerberos warning configuration file
/etc/pam.conf	PAM configuration file
/tmp/krb5cc_uid	Default credentials cache (<i>uid</i> is the decimal UID of the user)
/tmp/ovsec_admin.xxxxx	Temporary credentials cache for the lifetime of the password changing operation (<i>xxxxxx</i> is a random string)
/var/krb5/.k5.REALM	KDC stash file; contains encrypted copy of the KDC master key
/var/krb5/kadmin.log	Log file for kadmind
/var/krb5/kdc.log	Log file for the KDC
/var/krb5/principal.db	Kerberos principal database
/var/krb5/principal.kadm5	Kerberos administrative database; contains policy information
/var/krb5/principal.kadm5.lock	Kerberos administrative database lock file
/var/krb5/principal.ok	Kerberos principal database initialization file; created when the Kerberos database is initialized successfully
/var/krb5/slave_datatrans	Backup file of the KDC that the <i>kprop_script</i> uses for propagation

PAM Configuration File

The default PAM configuration file includes entries for the authentication service, account management, session management, and password management modules.

For the authentication module, the new entries are for `rlogin`, `login`, and `dtlogin`. An example of these entries is shown below. All of these services use the new PAM library, `/usr/lib/security/pam_krb5.so.1`, to provide Kerberos authentication.

The first three entries employ the `try_first_pass` option, which requests authentication using the user's initial password. Using the initial password means that the user is not prompted for another password even if multiple mechanisms are listed.

```
# cat /etc/pam.conf
:
:
rlogin auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
login auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
dtlogin auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
other auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
```

For the account management, `dtlogin` has a new entry that uses the Kerberos library, as shown below. An other entry is included to provide a default rule. Currently no actions are taken by the other entry.

```
dtlogin account optional /usr/lib/security/pam_krb5.so.1
other account optional /usr/lib/security/pam_krb5.so.1
```

The last two entries in the `/etc/pam.conf` file are shown below. The other entry for session management destroys user credentials. The new other entry for password management selects the Kerberos library.

```
other session optional /usr/lib/security/pam_krb5.so.1
other password optional /usr/lib/security/pam_krb5.so.1 try_first_pass
```

SEAM Commands

This section lists some of the commands included in the SEAM product.

TABLE 12-2 SEAM Commands

File Name	Description
<code>/usr/lib/krb5/kprop</code>	Kerberos database propagation program
<code>/usr/sbin/gkadmin</code>	Kerberos database administration GUI program; used to manage principals and policies

TABLE 12-2 SEAM Commands (Continued)

File Name	Description
<code>/usr/sbin/kadmin</code>	Remote Kerberos database administration program (run with Kerberos authentication); used to manage principals, policies, and keytab files
<code>/usr/sbin/kadmin.local</code>	Local Kerberos database administration program (run without Kerberos authentication; must be run on master KDC); used to manage principals, policies, and keytab files
<code>/usr/sbin/kdb5_util</code>	Creates Kerberos databases and stash files

SEAM Daemons

The daemons that are used by the SEAM product are listed in the following table.

TABLE 12-3 SEAM Daemons

File Name	Description
<code>/usr/lib/krb5/kadmind</code>	Kerberos database administration daemon
<code>/usr/lib/krb5/kpropd</code>	Kerberos database propagation daemon
<code>/usr/lib/krb5/krb5kdc</code>	Kerberos ticket processing daemon

SEAM Terminology

The following section presents terms and their definitions that are used throughout the SEAM documentation. In order to follow many of the discussions, a understanding of these terms is essential.

Kerberos-Specific Terminology

Understanding the terms presented in this section, is needed when studying the sections about the administering the KDCs.

The *Key Distribution Center* or *KDC* is the portion of SEAM that is responsible for issuing credentials. These credentials are created using information stored in the KDC database. Each realm will need at least two KDCs, a master and at least one slave. All KDCs generate credentials, but only the master handles any changes to the KDC database.

A *stash file* contains an encrypted copy of the master key for the KDC. This key is used when a server is rebooted to automatically authenticate the KDC before starting `kadmind` and `krb5kdc`. Because this file includes the master key, the file and any backups of the file should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.

Authentication-Specific Terminology

The terms discussed below are necessary for an understanding of the authentication process. Programmers and system administrators should be familiar with these terms.

A *client* is the software running on a user's workstation. The SEAM software running on the client makes many requests during this process, and it is important to differentiate the actions of this software from the user.

The terms *server* and *service* are often used interchangeably. To make things clearer, the term *server* is used to define the physical system that SEAM software is running on. The term *service* corresponds to a particular function that is being supported on a server (for instance, `nfs`). Documentation often mentions servers as part of a service, but using this definition clouds the meaning of the terms; therefore, servers refer to the physical system and service refers to the software.

The SEAM product includes three types of keys. One of them is the *private key*. This key is given to each user principal and is known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. For servers and services, the key is known as a *service key*. This key serves the same purpose as the private key, but is used by servers and services. The third type of key is a *session key*. This is a key generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service.

A *ticket* is an information packet used to securely pass the identity of a user to a server or service. A ticket is good for only a single client and a particular service on a specific server. It contains the principal name of the service, the principal name of the user, the IP address of the user's host, a timestamp, and a value to define the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. After a ticket has been created, it can be reused until the ticket expires.

A *credential* is a packet of information that includes a ticket and a matching session key. Credentials are often encrypted using either a private key or a service key depending on what will be decrypting the credential.

An *authenticator* is another type of information. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user's host, and a timestamp. Unlike a ticket, an authenticator can be used once only, usually when access to a service is requested. An authenticator is encrypted using the session key for that client and that server.

Types of Tickets

Tickets have properties that govern how they can be used. These properties are assigned to the ticket when it is created, although you can modify a ticket's properties later. (For example, a ticket can change from *forwardable* to *forwarded*). You can view ticket properties with the `klist` command (see "How to View Tickets" on page 181).

Tickets can be described by one or more of the following terms:

forwardable/forwarded	A forwardable ticket can be sent from one host to another, obviating the need for a client to reauthenticate itself. For example, if the user <code>david</code> obtains a forwardable ticket while on <code>jennifer</code> 's machine, he can log in to his own machine without having to get a new ticket (and thus authenticate himself again). (See "Example — Creating a Ticket" on page 180 for an example of a forwardable ticket.) Compare a forwardable ticket to a <i>proxiable</i> ticket, below.
initial	An <i>initial</i> ticket is one that is issued directly, not based on a ticket-granting ticket. Some services, such as applications that change passwords, can require tickets to be marked <i>initial</i> in order to assure themselves that the client can demonstrate a knowledge of its secret key — because an <i>initial</i> ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might have been around for a long time).
invalid	An <i>invalid</i> ticket is a postdated ticket that has not yet become usable. (See <i>postdated</i> , below.) It will be rejected by an application server until it becomes validated. To be validated, it must be presented to the KDC by the client in a TGS request, with the <code>VALIDATE</code> flag set, after its start time has passed.
postdatable/postdated	A <i>postdated</i> ticket is one that does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs intended to be run late at night, since the ticket, if stolen, cannot be used until the

batch job is to be run. When a *postdated* ticket is issued, it is issued as *invalid* and remains that way until: its start time has passed, and the client requests validation by the KDC. (See *invalid*, above.) A *postdated* ticket is normally valid until the expiration time of the ticket-granting ticket; however, if it is marked *renewable*, its lifetime is normally set to be equal to the duration of the full life of the ticket-granting ticket. See *renewable*, below.

proxiabile/proxy

At times it can be necessary for a principal to allow a service to perform an operation on its behalf. (An example might be when a principal requests a service to run a print job on a third host.) The service must be able to take on the identity of the client, but need only do so for that single operation. In that case, the server is said to be acting as a *proxy* for the client. The principal name of the proxy must be specified when the ticket is created.

A *proxiabile* ticket is similar to a *forwardable* ticket, except that it is valid only for a single service, whereas a *forwardable* ticket grants the service the complete use of the client's identity. A *forwardable* ticket can therefore be thought of as a sort of super-proxy.

renewable

Because it is a security risk to have tickets with very long lives, tickets can be designated as *renewable*. A *renewable* ticket has two expiration times: the time at which the current instance of the ticket expires, and the maximum lifetime for any ticket. If a client wants to continue to use a ticket, it renews it before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of ten hours. If the client holding the ticket wants to keep it for more than an hour, the client must renew it within that hour. When a ticket reaches the maximum ticket lifetime (10 hours), it automatically expires and cannot be renewed.

For information on how to view tickets to see what their attributes are, see "How to View Tickets" on page 181.

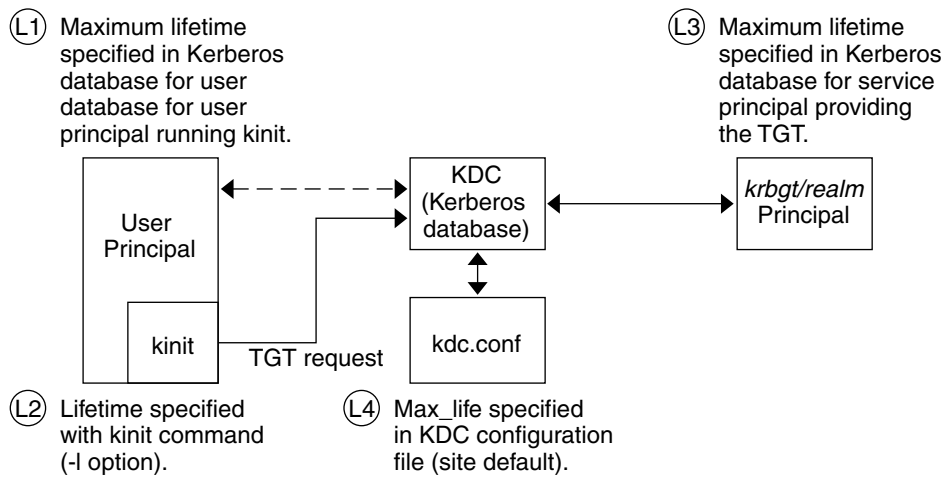
Ticket Lifetimes

Any time a principal obtains a ticket, including a ticket-granting ticket, the ticket's lifetime is set as the smallest of the following lifetime values:

- The lifetime value specified by the `-l` option of `kinit`, if `kinit` is used to get the ticket

- The maximum lifetime value (`max_life`) specified in the `kdc.conf` file
- The maximum lifetime value specified in the Kerberos database for the service principal providing the ticket. (In the case of `kinit`, the service principal is `krbtgt/realm`)
- The maximum lifetime value specified in the Kerberos database for the user principal requesting the ticket.

Figure 12–1 shows how a TGT’s lifetime is determined and illustrates where the four lifetime values come from. Even though Figure 12–1 shows how a TGT’s lifetime is determined, basically the same thing happens when any principal obtains a ticket. The only differences are that `kinit` doesn’t provide a lifetime value, and the service principal providing the ticket provides a maximum lifetime value (instead of the `krbtgt/realm` principal).



Ticket lifetime = Minimum value of L1, L2, L3, and L4

FIGURE 12–1 How a TGT’s Lifetime is Determined

The *renewable* ticket lifetime is also determined from the minimum of four values, but renewable lifetime values are used instead:

- The renewable lifetime value specified by the `-r` option of `kinit`, if `kinit` is used to obtain or renew the ticket
- The maximum renewable lifetime value (`max_renewable_life`) specified in the `kdc.conf` file
- The maximum lifetime renewable value specified in the Kerberos database for the service principal providing the ticket (In the case of `kinit`, the service principal is `krbtgt/realm`)

- The maximum lifetime renewable value specified in the Kerberos database for the user principal requesting the ticket

Principal Names

Each ticket is identified by a principal name. The principal name can identify a user or a service. Here are examples of several of the principal names.

TABLE 12-4 Examples of Principal Names

Principal Name	Description
<code>root/boston.example.com@EXAMPLE.COM</code>	A principal associated with the <code>root</code> account on an NFS client. This is called a <code>root</code> principal and is needed for authenticated NFS-mounting to succeed.
<code>host/boston.example.com@EXAMPLE.COM</code>	A principal used by the Kerberized applications (<code>klist</code> and <code>kprop</code> for example) . This is called a <code>host</code> or <code>service</code> principal.
<code>username@EXAMPLE.COM</code>	A principal for a user
<code>username/admin@EXAMPLE.COM</code>	An <code>admin</code> principal that can be used to administer the KDC database
<code>nfs/boston.example.com@EXAMPLE.COM</code>	A principal used by the NFS service. This can be used instead of a <code>host</code> principal.
<code>K/M@EXAMPLE.COM</code>	The master key name principal. There is one of these associated with each master KDC.
<code>kadmin/history@EXAMPLE.COM</code>	A principal which includes a key used to keep password histories for other principals. There is one of these for each master KDC.
<code>kadmin/kdc1.example.com@EXAMPLE.COM</code>	A principal for the master KDC server that allows access to the KDC using <code>kadmin</code>
<code>changepw/kdc1.example.com@EXAMPLE.COM</code>	A principal for the master KDC server that allows access to the KDC when changing passwords
<code>krbtgt/EXAMPLE.COM@EXAMPLE.COM</code>	This principal is used when generating a ticket granting ticket.

How the Authentication System Works

Applications allow you to log on to a remote system if you can provide a ticket that proves your identity and a matching session key. The session key contains information that is specific to the user and the service being accessed. A ticket and session key are created by the KDC for all users when they first log in. The ticket and matching session key form a credential. While using multiple networking services, a user can gather many credentials. The user needs to have a credential for each service running on a particular server. For instance, access to the `ftp` service on a server named `boston` requires one credential, and access to the `ftp` service on another server requires its own credential.

The process of creating and storing the credentials is transparent. Credentials are created by the KDC that sends the credential to the requestor. When received, the credential is stored in a credential cache.

Gaining Access to a Service Using SEAM

In order for a user to access a specific service on a specific server, the user must obtain two things. The first is a credential for the ticket-granting service (known as the TGT). Once the ticket-granting service has decrypted this credential, the service creates a second credential for the server that the user is requesting access to. This second credential can then be used to request access to the service on the server. After the server has successfully decrypted the second credential, then the user is given access. This process is described in more detail below.

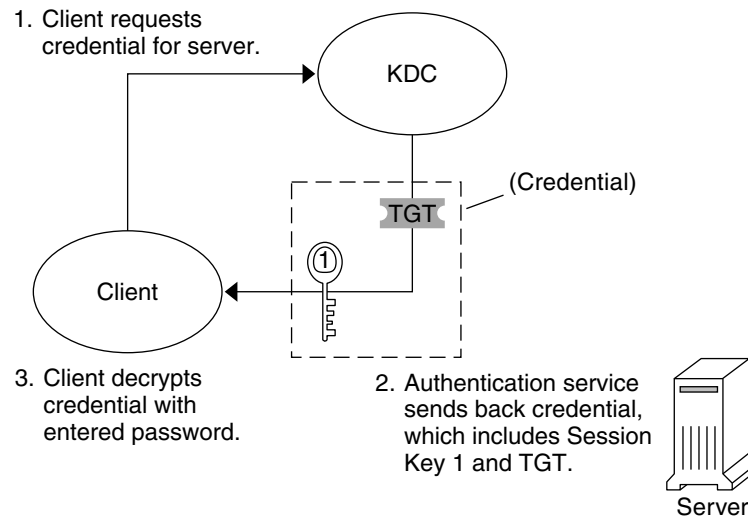
Obtaining a Credential for the Ticket-Granting Service

1. To start the authentication process, the client sends a request to the authentication server for a specific user principal. This request is sent without encryption. There is no secure information included in the request, so it is not necessary to use encryption.
2. When the request is received by the authentication service, the principal name of the user is looked up in the KDC database. If a principal matches, the authentication service obtains the private key for that principal. The authentication service then generates a session key to be used by the client and the ticket-granting

service (call it session key 1) and a ticket for the ticket-granting service (ticket 1). This ticket is also known as the ticket-granting ticket (TGT). Both the session key and the ticket are encrypted using the user's private key and the information is sent back to the client.

3. The client uses this information to decrypt session key 1 and ticket 1, using the private key for the user principal. Since the private key should only be known by the user and the KDC database, the information in the packet should be safe. The client stores the information in the credentials cache.

Normally during this a user is prompted for her password. If the password she enters is the same as the one used to build the private key stored in the KDC database, then the client can successfully decrypt the information sent by the authentication service. Now the client has a credential to be used with the ticket-granting service. The client is ready to request a credential for a server.



TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 12-2 Obtaining a Credential for the Ticket-Granting Service

Obtaining a Credential for a Server

1. To request access to a specific server, a client must first have obtained a credential for that server from the authentication service (see "Obtaining a Credential for the Ticket-Granting Service" on page 196). The client then sends a request to the ticket-granting service, which includes the service principal name, ticket 1, and an authenticator encrypted with session key 1. Ticket 1 was originally encrypted by

the authentication service using the service key of the ticket-granting service.

2. Because the service key of the ticket-granting service is known to the ticket-granting service, ticket 1 can be decrypted. The information included in ticket 1 includes session key 1, so the ticket-granting service can decrypt the authenticator. At this point, the user principal is authenticated with the ticket-granting service.
3. Once the authentication is successful, the ticket-granting service generates a session key for the user principal and the server (session key 2) and a ticket for the server (ticket 2). Session key 2 and ticket 2 are then encrypted using session key 1. Since session key 1 is known only to the client and the ticket-granting service, this information is secure and can be safely set over the net.
4. When the client receives this information packet, it decrypts the information using session key 1, which it had stored in the credential cache. The client has obtained a credential to be used with the server. Now the client is ready to request access to a particular service on that server.

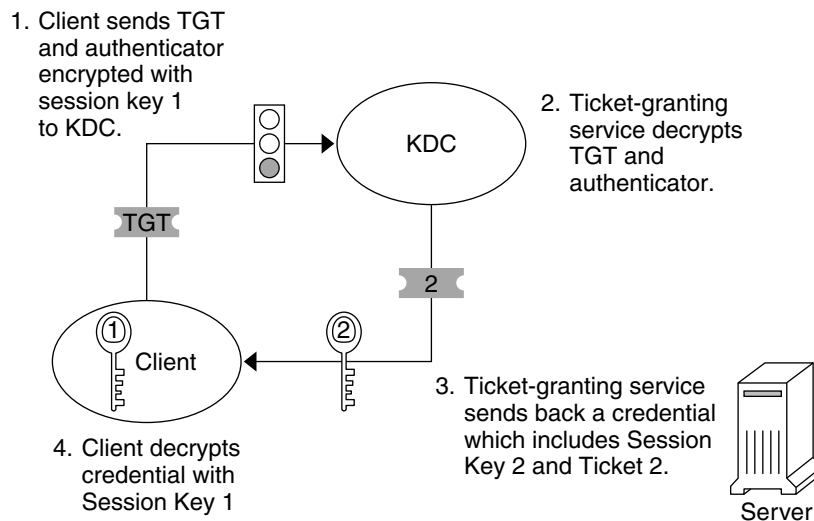


FIGURE 12-3 Obtaining a Credential for a Server

Obtaining Access to a Specific Service

1. To request access to a specific service, the client must first have obtained a credential for the ticket-granting service from the authentication server, and a server credential from the ticket-granting service (see “Obtaining a Credential for the Ticket-Granting Service” on page 196 and “Obtaining a Credential for a Server” on page 197). The client can send a request to the server including ticket 2 and another authenticator. The authenticator is encrypted using session key 2.

2. Ticket 2 was encrypted by the ticket-granting service with the service key for the service. Since the service key is known by the service principal, the service can decrypt ticket 2 and get session key 2. Session key 2 can then be used to decrypt the authenticator. If the authenticator is successfully decrypted, the client is given access to the service.

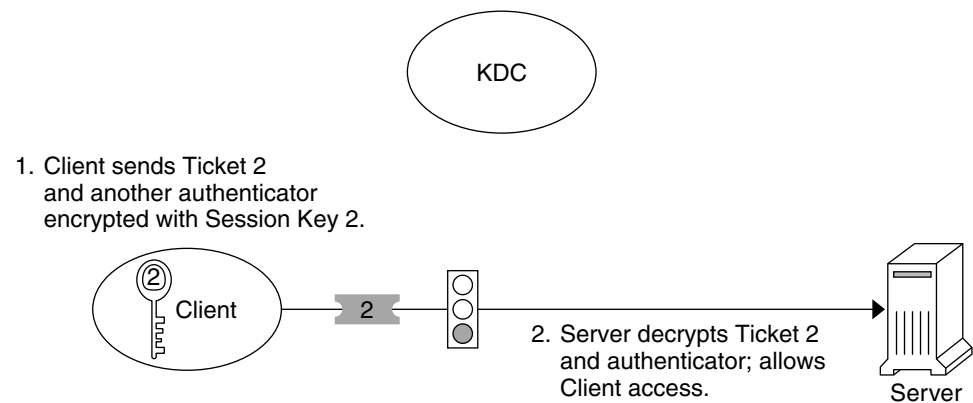


FIGURE 12-4 Obtaining Access to a Specific Service

Using the `gsscred` Table

The `gsscred` table is used by an NFS server when the server is trying to identify a SEAM user. The NFS services use UNIX IDs to identify users and these IDs are not part of a user principal or credential. The `gsscred` table provides a mapping from UNIX UIDs (from the password file) to principal names. The table must be created and administered after the KDC database is populated.

When a client request comes in, the NFS services try to map the principal name to a UNIX ID. If the mapping fails, the `gsscred` table is consulted. With the `kerberos_v5` mechanism, a `root/hostname` principal is automatically mapped to UID 0, and the `gsscred` table is not consulted. This means that there is no way to do special remappings of `root` through the `gsscred` table.

Managing System Security Topics

This section provides instructions for managing system security in the Solaris environment. This section contains these chapters.

Chapter 14	Provides overview information about file, system, and network security.
Chapter 15	Provides step-by-step instructions to display file information, change file ownership and permissions, and set special permissions.
Chapter 16	Provides step-by-step instructions to check login status, set up dial-up passwords, restrict root access, and monitor root access and <code>su</code> attempts.
Chapter 17	Provides overview information for using role-based access control.
Chapter 18	Provides step-by-step instructions for using role-based access control.
Chapter 19	Provides reference information about role-based access control.
Chapter 3	Provides step-by-step instructions for setting up login authentication and Pluggable Authentication Module (PAM).
Chapter 6	Provides overview information about SEAM.
Chapter 7	Provides a list of information or issues that need to be resolved before installing SEAM.
Chapter 8	Provides step-by-step instructions for configuring SEAM.
Chapter 9	Provides a list of SEAM error messages, how to fix the problem that generates the message and how to troubleshoot some error conditions.

Chapter 10	Provides step-by-step instructions for administering principles and policies in SEAM with <code>gkadmin</code> and at the command line.
Chapter 11	Provides user instructions for SEAM.
Chapter 12	Provides additional information about SEAM.
Chapter 20	Provides overview information about Automated Security Enhancement Tool (ASET) and step-by-step instructions to run ASET interactively or periodically (by using a <code>cron</code> job). It also includes information about collecting client ASET reports on a server.

Managing System Security (Overview)

Keeping a system's information secure is an important system administration responsibility. This chapter provides overview information about managing system security at the file, system, and network level.

This is a list of the overview information in this chapter.

- "Controlling Access to a Computer System" on page 204
- "File Security" on page 207
- "System Security" on page 208
- "Network Security" on page 212

Where to Find System Security Tasks

Use these references to find step-by-step instructions for setting up system security.

- Chapter 15
- Chapter 16
- Chapter 18
- Chapter 3
- Chapter 8
- Chapter 20
- Chapter 24

Controlling Access to a Computer System

At the file level, the SunOS operating system provides some standard security features that you can use to protect files, directories, and devices. At the system and network levels, the security issues are mostly the same. In the workplace, a number of systems connected to a server can be thought of as one large multifaceted system. The system administrator is responsible for the security of this larger system or network. Not only is it important to defend the network from outsiders trying to gain access to the network, but it is also important to ensure the integrity of the data on the systems within the network.

The first line of security defense is to control access to your system. You can control and monitor system access by:

- Maintaining physical site security
- Maintaining login control
- Restricting access to data in files
- Maintaining network control
- Monitoring system usage
- Setting the path variable correctly
- Securing files
- Installing a firewall
- Reporting security problems

Maintaining Physical Site Security

To control access to your system, you must maintain the physical security of your computer environment. For instance, if a system is logged in and left unattended, anyone who can use that system can gain access to the operating system and the network. You need to be aware of your computer's surroundings and physically protect it from unauthorized access.

Maintaining Login and Access Control

You also must restrict unauthorized logins to a system or the network, which you can do through password and login control. All accounts on a system should have a password. An account without a password makes your entire network accessible to anyone who can guess a user name.

Solaris software restricts control of certain system devices to the user login account. Only a process running as superuser or console user can access a system mouse, keyboard, frame buffer, or audio device unless `/etc/logindevperm` is edited. See `logindevperm(4)` for more information.

Restricting Access to Data in Files

After you have established login restrictions, you can control access to the data on your system. You might want to allow some users to read some files, and give other users permission to change or delete some files. You might have some data that you do not want anyone else to see. Chapter 15 discusses how to set file permissions.

Maintaining Network Control

Computers are often part of a configuration of systems called a *network*. A network allows connected systems to exchange information and access data and other resources available from systems connected to the network. Networking has created a powerful and sophisticated way of computing. However, networking has also jeopardized computer security.

For instance, within a network of computers, individual systems are open to allow sharing of information. Also, because many people have access to the network, there is more chance for allowing unwanted access, especially through user error (for example, through a poor use of passwords).

Monitoring System Usage

As system administrator, you need to monitor system activity, being aware of all aspects of your systems, including the following:

- What is the normal load?
- Who has access to the system?
- When do individuals access the system?

With this kind of knowledge, you can use the available tools to audit system use and monitor the activities of individual users. Monitoring is very useful when there is a suspected breach in security.

Setting the Correct Path

It is important to set your path variable correctly; otherwise, you can accidentally run a program introduced by someone else that harms your data or your system. This kind of program, which creates a security hazard, is referred to as a “Trojan horse.” For example, a substitute `su` program could be placed in a public directory where you, as system administrator, might run it. Such a script would look just like the regular `su` command; since it removes itself after execution, it is hard to tell that you have actually run a Trojan horse.

The path variable is automatically set at login time through the startup files: `.login`, `.profile`, and `.cshrc`. Setting up the user search path so that the current directory (`.`) comes last prevents you or your users from running this type of Trojan horse. The path variable for superuser should not include the current directory at all. The ASET utility examines the startup files to ensure that the path variable is set up correctly and that it does not contain a dot (`.`) entry.

Securing Files

Since the SunOS operating system is a multiuser system, file system security is the most basic, and important, security risk on a system. You can use both the traditional UNIX file protection or the more secure access control lists (ACLs) to protect your files.

Also, many executable programs have to be run as root (that is, as superuser) to work properly. These executables run with the user ID set to 0 (`setuid=0`). Anyone running these programs runs them with the root ID, which creates a potential security problem if the programs are not written with security in mind.

Except for the executables shipped with `setuid` to root, you should disallow the use of `setuid` programs, or at least restrict and keep them to a minimum.

Installing a Firewall

Another way to protect your network is to use a firewall or secure gateway system. A firewall is a dedicated system separating two networks, each of which approaches the other as untrusted. You should consider this setup as mandatory between your internal network and any external networks, such as the Internet, with which you want internal network users to communicate.

A firewall can also be useful between some internal networks. For example, the firewall or secure gateway computer will not send a packet between two networks unless the gateway computer is the origin or the destination address of the packet. A firewall should also be set up to forward packets for particular protocols only. For

example, you can allow packets for transferring mail, but not those for `telnet` or `rlogin`. The `ASET` utility, when run at high security, disables the forwarding of Internet Protocol (IP) packets.

Reporting Security Problems

If you experience a suspected security breach, you can contact the Computer Emergency Response Team/Coordination Center (CERT/CC), which is a Defense Advanced Research Projects Agency (DARPA) funded project located at the Software Engineering Institute at Carnegie Mellon University. It can assist you with any security problems you are having. It can also direct you to other Computer Emergency Response Teams that might be more appropriate to your particular needs. You can call CERT/CC at its 24-hour hotline: (412) 268-7090, or contact the team by email at `cert@cert.sei.cmu.edu`.

File Security

The SunOS operating system is a multiuser system, which means that all the users logged in to a system can read and use files belonging to one another, as long as they have permission to do so. The table below describes file system administration commands. See Chapter 15 for step-by-step instructions on securing files.

File Administration Commands

This table describes the file administration commands for monitoring and securing files and directories.

TABLE 14-1 File Administration Commands

Command	Description
<code>ls(1)</code>	Lists the files in a directory and information about them.
<code>chown(1)</code>	Changes the ownership of a file.
<code>chgrp(1)</code>	Changes the group ownership of a file.
<code>chmod(1)</code>	Changes permissions on a file. You can use either symbolic mode (letters and symbols) or absolute mode (octal numbers) to change permissions on a file.

File Encryption

Placing a sensitive file into an inaccessible directory (700 mode) and making the file unreadable by others (600 mode) will keep it secure in most cases. However, someone who guesses your password or the root password can read and write to that file. Also, the sensitive file is preserved on backup tapes every time you back up the system files to tape.

Fortunately, an additional layer of security is available to all SunOS system software users in the United States—the optional file encryption kit. The encryption kit includes the `crypt(1)` command which scrambles the data to disguise the text.

Access Control Lists (ACLs)

ACLs (ACLs, pronounced “ackkls”) can provide greater control over file permissions when the traditional UNIX file protection in the SunOS operating system is not enough. The traditional UNIX file protection provides read, write, and execute permissions for the three user classes: owner, group, and other. An ACL provides better file security by enabling you to define file permissions for the owner, owner’s group, others, specific users and groups, and default permissions for each of those categories. See “Using Access Control Lists (ACLs)” on page 232 for step-by-step instructions on using ACLs.

The table below lists the commands for administering ACLs on files or directories.

TABLE 14-2 ACL Commands

Command	Description
<code>setfacl(1)</code>	Sets, adds, modifies, and deletes ACL entries
<code>getfacl(1)</code>	Displays ACL entries

System Security

This section describes how to safeguard your system against unauthorized access, such as how to prevent an intruder from logging in to your system, how to maintain the password files, and how to prevent unauthorized superuser access to sensitive system files and programs.

You can set up two security barriers on a system. The first security barrier is the login program. To cross this barrier and gain access to a system, a user must supply a user name and a corresponding password known by the local system or by the name service (NIS or NIS+).

The second security barrier is ensuring that the system files and programs can be changed or removed by superuser only. A would-be superuser must supply the root user name and its correct password.

Login Access Restrictions

When a user logs in to a system, the login program consults the appropriate database according to the information listed in the `/etc/nsswitch.conf` file. The entries in this file can include `files` (designating the `/etc` files), `nis` (designating the NIS database), and `nisplus` (designating the NIS+ database). See the *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* or `nsswitch.conf(4)` for a description of this file.

The login program verifies the user name and password entered. If the user name is not in the password file or the password is not correct for the user name, the login program denies access to the system. When the user supplies a name from the password file and the correct password for the name, the system grants the user access to the system.

Special Logins

There are two common ways to access a system—by using a conventional user login, or by using the root login. In addition, a number of special *system* logins allow a user to perform administrative commands without using the root account. The administrator assigns passwords to these login accounts.

The table below lists the system login accounts and their uses. The system logins perform special functions, and each has its own group identifier number (GID). Each of these logins should have its own password, which should be distributed on a need-to-know basis.

TABLE 14-3 System Logins

Login Account	GID	Use
root	0	Has almost no restrictions and overrides all other logins, protections, and permissions. The root account has access to the entire system. The password for the root login should be very carefully protected. Owns most of the Solaris commands.
daemon	1	Controls background processing.
bin	2	Owns some of the Solaris commands.
sys	3	Owns many system files.
adm	4	Owns certain administrative files.
lp	71	Owns the object and spooled data files for the printer.
uucp	5	Owns the object and spooled data files for UUCP, the UNIX-to-UNIX copy program.
nuucp	9	Is used by remote systems to log in to the system and start file transfers.

You should also set the security of the `eeprom` command to require a password. See `eeprom(1M)` for more information.

Managing Password Information

When logging in to a system, users must enter both a user name and a password. Although logins are publicly known, passwords must be kept secret, known only to users. You should ask your users to choose their passwords carefully, and they should change them often.

Passwords are initially created when you set up a user account. To maintain security on user accounts, you can set up password aging to force users to routinely change their passwords, and you can also disable a user account by locking the password. See “Managing User Accounts and Groups (Overview)” in *System Administration Guide: Basic Administration* and `passwd(1)` for detailed information about setting up and maintaining passwords.

NIS+ Password File

If your network uses NIS+, the password information is kept in the NIS+ database. Information in the NIS+ database can be protected by restricting access to authorized users. You can use the `passwd` command to change a user’s NIS+ password.

NIS Password File

If your network uses NIS, the password information is kept in the NIS password map. NIS does not support password aging. You can use the `passwd` command to change a user's NIS password.

/etc Files

If your network uses `/etc` files, the password information is kept in the system's `/etc/passwd` and `/etc/shadow` files. The user name and other information are kept in the password file `/etc/passwd`, while the encrypted password itself is kept in a separate *shadow* file, `/etc/shadow`. This is a security measure that prevents a user from gaining access to the encrypted passwords. While the `/etc/passwd` file is available to anyone who can log in to a machine, only superuser can read the `/etc/shadow` file. You can use the `passwd` command to change a user's password on a local system.

Using the Restricted Shell

The standard shell allows a user to open files, execute commands, and so on. The restricted shell can be used to limit the ability of a user to change directories and execute commands. The restricted shell (`rsh`) is located in the `/usr/lib` directory. (Note that this is not the remote shell, which is `/usr/sbin/rsh`.) The restricted shell differs from the normal shell in these ways:

- The user is limited to the home directory (can't use `cd` to change directories).
- The user can use only commands in the `PATH` set by the system administrator (can't change the `PATH` variable).
- The user can access only files in the home directory and its subdirectories (can't name commands or files using a complete path name).
- The user cannot redirect output with `>` or `>>`.

The restricted shell allows the system administrator to limit a user's ability to stray into the system files, and is intended mainly to set up a user who needs to perform specific tasks. The `rsh` is not completely secure, however, and is only intended to keep unskilled users from getting into (or causing) trouble.

See `rsh(1M)` for information about the restricted shell.

Tracking Superuser (Root) Login

Your system requires a root password for superuser mode. In the default configuration, a user cannot remotely log in to a system as root. When logging in remotely, a user must log in as himself and then use the `su` command to become root. This enables you to track who is using superuser privileges on your system.

Monitoring Who is Becoming Superuser or Other Users

You have to use the `su` command to change to another user, for example, if you want to become superuser. For security reasons, you can monitor who has been using the `su` command, especially those users who are trying to gain superuser access.

See “How to Monitor Who Is Using the `su` Command” on page 250 for detailed instructions.

Network Security

The more available access is across a network, the more advantageous it is for networked systems. However, free access and sharing of data and resources create security problems. Network security is usually based on limiting or blocking operations from remote systems. The figure below describes the security restrictions you can impose on remote operations.

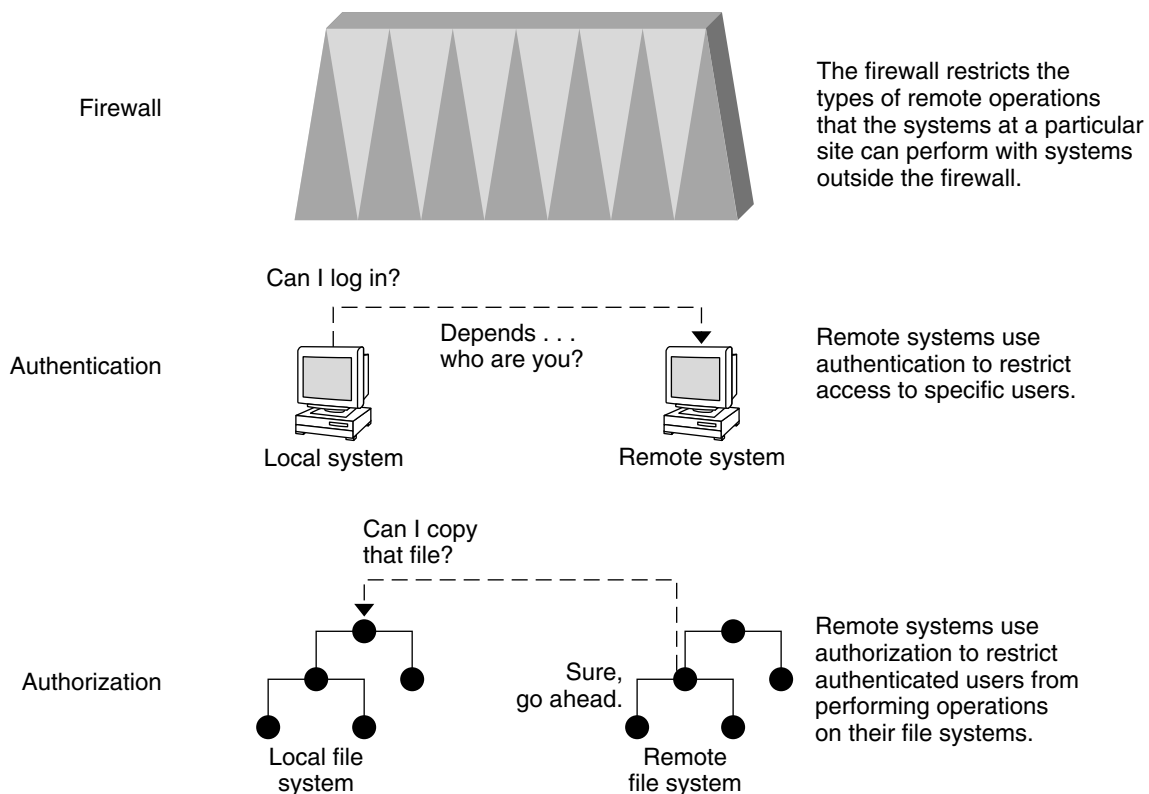


FIGURE 14-1 Security Restrictions for Remote Operations

Firewall Systems

You can set up a firewall system to protect the resources in your network from outside access. A *firewall system* is a secure host that acts as a barrier between your internal network and outside networks.

The firewall has two functions. It acts as a gateway which passes data between the networks, and it acts as a barrier which blocks the free passage of data to and from the network. The firewall requires a user on the internal network to log in to the firewall system to access hosts on remote networks. Similarly, a user on an outside network must log in to the firewall system before being granted access to a host on the internal network.

In addition, all electronic mail sent from the internal network is sent to the firewall system for transfer to a host on an external network. The firewall system receives all incoming electronic mail, and distributes it to the hosts on the internal network.



Caution – A firewall prevents unauthorized users from accessing hosts on your network. You should maintain strict and rigidly enforced security on the firewall, but security on other hosts on the network can be more relaxed. However, an intruder who can break into your firewall system can then gain access to all the other hosts on the internal network.

A firewall system should not have any *trusted hosts*. (A trusted host is one from which a user can log in without being required to type in a password.) It should not share any of its file systems, or mount any file systems from other servers.

ASET can be used to make a system into a firewall, and to enforce high security on a firewall system, as described in Chapter 20.

Packet Smashing

Most local area networks transmit data between computers in blocks called packets. Through a procedure called *packet smashing*, unauthorized users can harm or destroy data. Packet smashing involves capturing packets before they reach their destination, injecting arbitrary data into the contents, then sending the packets back on their original course. On a local area network, packet smashing is impossible because packets reach all systems, including the server, at the same time. Packet smashing is possible on a gateway, however, so make sure all gateways on the network are protected.

The most dangerous attacks are those that affect the integrity of the data. Such attacks involve changing the contents of the packets or impersonating a user. Attacks that involve eavesdropping—recording conversations and replaying them later without impersonating a user—do not compromise data integrity. These attacks do affect privacy, however. You can protect the privacy of sensitive information by encrypting data that goes over the network.

Authentication and Authorization

Authentication is a way to restrict access to specific users when accessing a remote system, which can be set up at both the system or network level. Once a user gains access to a remote system, *authorization* is a way to restrict operations that the user can perform on the remote system. The table below lists the types of authentications and authorizations that can help protect your systems on the network against unauthorized use.

TABLE 14-4 Types of Authentication and Authorization

Type	Description	Where to Find Information
NIS+	The NIS+ name service can provide both authentication and authorization at the network level.	<i>System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)</i>
Remote Login Programs	The remote login programs (<code>rlogin</code> , <code>rcp</code> , <code>ftp</code>) enable users to log in to a remote system over the network and use its resources. If you are a “trusted host,” authentication is automatic; otherwise, you are asked to authenticate yourself.	“Accessing Remote Systems (Tasks)” in <i>System Administration Guide: Resource Management and Network Services</i>
Secure RPC	Secure RPC improves the security of network environments by authenticating users who make requests on remote systems. You can use either the UNIX, DES, or Kerberos authentication system for Secure RPC.	“Overview of Secure RPC” on page 29
	Secure RPC can also be used to provide additional security to the NFS environment, called Secure NFS.	“NFS Services and Secure RPC” on page 30
DES Encryption	The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt a secret key.	“DES Encryption” on page 30
Diffie-Hellman Authentication	This authentication method is based on the ability of the sending system to use the common key to encrypt the current time, which the receiving system can decrypt and check against its current time.	“Diffie-Hellman Authentication” on page 31
Kerberos	Kerberos uses DES encryption to authenticate a user when logging in to the system.	Chapter 3

Sharing Files

A network file server can control which files are available for sharing. It can also control which clients have access to the files, and what type of access is permitted to those clients. In general, the file server can grant read/write or read-only access either to all clients or to specific clients. Access control is specified when resources are made available with the `share` command.

A server can use the `/etc/dfs/dfstab` file to list the file systems it makes available to clients on the network. See “Automatic File-System Sharing” in *System Administration Guide: Resource Management and Network Services* for more information about sharing file systems.

Restricting Superuser (Root) Access

In general, superuser is not allowed root access to file systems shared across the network. Unless the server specifically grants superuser privileges, a user who is logged in as superuser on a client cannot gain root access to files that are remotely mounted on the client. The NFS system implements this by changing the user ID of the requester to the user ID of the user name, `nobody`; this is generally 60001. The access rights of user `nobody` are the same as those given to the public (or a user without credentials) for a particular file. For example, if the public has only execute permission for a file, then user `nobody` can only execute that file.

An NFS server can grant superuser privileges on a shared file system on a per-host basis, using the `root=hostname` option to the `share` command.

Using Privileged Ports

If you do not want to run Secure RPC, a possible substitute is the Solaris “privileged port” mechanism. A privileged port is built up by the superuser with a port number of less than 1024. After a client system has authenticated the client’s credential, it builds a connection to the server via the privileged port. The server then verifies the client credential by examining the connection’s port number.

Non-Solaris clients, however, might not be able to communicate via the privileged port. If they cannot, you will see error messages such as these:

```
"Weak Authentication  
NFS request from unprivileged port"
```

Using Automated Security Enhancement Tool (ASET)

The ASET security package provides automated administration tools that enable you to control and monitor your system’s security. You specify a security level—low, medium, or high—at which ASET will run. At each higher level, ASET’s file-control functions increase to reduce file access and tighten your system security.

See Chapter 20 for more information.

Securing Files (Tasks)

This chapter describes the procedures for securing files. This is a list of the step-by-step instructions in this chapter.

- “How to Display File Information” on page 221
- “How to Change the Owner of a File” on page 223
- “How to Change Group Ownership of a File” on page 224
- “How to Change Permissions in Absolute Mode” on page 227
- “How to Change Special Permissions in Absolute Mode” on page 228
- “How to Change Permissions in Symbolic Mode” on page 229
- “How to Find Files With `setuid` Permissions” on page 230
- “How to Disable Programs From Using Executable Stacks” on page 232
- “How to Set an ACL on a File” on page 235
- “How to Copy an ACL” on page 237
- “How to Check If a File Has an ACL” on page 237
- “How to Modify ACL Entries on a File” on page 238
- “How to Delete ACL Entries From a File” on page 238
- “How to Display ACL Entries for a File” on page 239

File Security Features

This section describes the features that constitute a file’s security.

User Classes

For each file, there are three classes of users that specify the levels of security:

- The file or directory owner—usually the user who created the file. The owner of a file can decide who has the right to read it, to write to it (make changes to it), or, if it is a command, to execute it.
- Members of a group.
- All others who are not the file or group owner.

Only the owner of the file or root can assign or modify file permissions.

File Permissions

The table below lists and describes the permissions you can give to each user class for a file.

TABLE 15-1 File Permissions

Symbol	Permission	Means Designated Users ...
r	Read	Can open and read the contents of a file
w	Write	Can write to the file (modify its contents), add to it, or delete it
x	Execute	Can execute the file (if it is a program or shell script), or run it with one of the <code>exec(1)</code> system calls
-	Denied	Cannot read, write, or execute the file

These file permissions apply to special files such as devices, sockets, and named pipes (FIFOs), as they do to regular files.

For a symbolic link, the permissions that apply are those of the file the link points to.

Directory Permissions

The table below lists and describes the permissions you can give to each user class for a directory.

TABLE 15-2 Directory Permissions

Symbol	Permission	Means Designated Users Can ...
r	Read	List files in the directory.
w	Write	Add or remove files or links in the directory.

TABLE 15-2 Directory Permissions (Continued)

Symbol	Permission	Means Designated Users Can ...
x	Execute	Open or execute files in the directory. Also can make the directory and the directories beneath it current.

You can protect the files in a directory (and in its subdirectories) by disallowing access to that directory. Note, however, that superuser has access to all files and directories on the system.

Special File Permissions (setuid, setgid and Sticky Bit)

Three special types of permissions are available for executable files and public directories. When these permissions are set, any user who runs that executable file assumes the user ID of the owner (or group) of the executable file.

You must be extremely careful when setting special permissions, because special permissions constitute a security risk. For example, a user can gain superuser permission by executing a program that sets the user ID to root. Also, all users can set special permissions for files they own, which constitutes another security concern.

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions to gain superuser privileges. See “How to Find Files With `setuid` Permissions” on page 230 to search for the file systems and print out a list of all programs using these permissions. A suspicious listing would be one that grants ownership of such a program to a user rather than to `root` or `bin`.

setuid Permission

When set-user identification (`setuid`) permission is set on an executable file, a process that runs this file is granted access based on the owner of the file (usually `root`), rather than the user who is running the executable file. This allows a user to access files and directories that are normally only available to the owner. For example, the `setuid` permission on the `passwd` command makes it possible for a user to change passwords, assuming the permissions of the root ID:

```
-r-sr-sr-x  3 root    sys      104580 Sep 16 12:02 /usr/bin/passwd
```

This presents a security risk, because some determined users can find a way to maintain the permissions granted to them by the `setuid` process even after the process has finished executing.

Note – Using `setuid` permissions with the reserved UIDs (0-100) from a program might not set the effective UID correctly. Use a shell script instead or avoid using the reserved UIDs with `setuid` permissions.

setgid Permission

The set-group identification (`setgid`) permission is similar to `setuid`, except that the process's effective group ID (GID) is changed to the group owner of the file, and a user is granted access based on permissions granted to that group. The `/usr/bin/mail` program has `setgid` permissions:

```
-r-x--s--x  1 root    mail      63628 Sep 16 12:01 /usr/bin/mail
```

When `setgid` permission is applied to a directory, files created in this directory belong to the group to which the directory belongs, not the group to which the creating process belongs. Any user who has write and execute permissions in the directory can create a file there—however, the file belongs to the group owning the directory, not to the user's group ownership.

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions to gain superuser privileges. See "How to Find Files With `setuid` Permissions" on page 230 to search for the file systems and print out a list of all programs using these permissions. A suspicious listing would be one that grants ownership of such a program to a user rather than to `root` or `bin`.

Sticky Bit

The *sticky bit* is a permission bit that protects the files within a directory. If the directory has the sticky bit set, a file can be deleted only by the owner of the file, the owner of the directory, or by `root`. This prevents a user from deleting other users' files from public directories such as `/tmp`:

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

Be sure to set the sticky bit manually when you set up a public directory on a TMPFS file system.

Default umask

When you create a file or directory, it has a default set of permissions. These default permissions are determined by the value of `umask(1)` in the system file `/etc/profile`, or in your `.cshrc` or `.login` file. By default, the system sets the

permissions on a text file to 666, granting read and write permission to user, group, and others, and to 777 on a directory or executable file.

The value assigned by `umask` is subtracted from the default. This has the effect of denying permissions in the same way that `chmod` grants them. For example, while the command `chmod 022` grants write permission to group and others, `umask 022` denies write permission for group and others.

The table below shows some typical `umask` settings, and the effect on an executable file.

TABLE 15-3 `umask` Settings for Different Security Levels

Level of Security	<code>umask</code>	Disallows
Permissive (744)	022	w for group and others
Moderate (740)	027	w for group, rwx for others
Moderate (741)	026	w for group, rw for others
Severe (700)	077	rwx for group and others

Displaying File Information

This section describes how to display file information.

▼ How to Display File Information

Display information about all the files in a directory by using the `ls` command.

```
$ ls -la
```

- l Displays the long format.
- a Displays all files, including hidden files that begin with a dot (.).

Each line in the display has the following information about a file:

- Type of file
A file can be one of seven types. The table below lists the possible file types.

TABLE 15-4 File Types

Symbol	Type
-	Text or program
D	Door
d	Directory
b	Block special file
c	Character special file
p	Named pipe (FIFO)
l	Symbolic link
s	Socket

- Permissions; see Table 15-1 and Table 15-2 for descriptions
- Number of hard links
- Owner of the file
- Group of the file
- Size of the file, in bytes
- Date the file was created or last date it was changed
- Name of the file

Example—Displaying File Information

The following example displays the partial list of the files in the `/sbin` directory.

```
$ cd /sbin
$ ls -la
total 13456
drwxr-xr-x  2 root    sys      512 Sep  1 14:11 .
drwxr-xr-x 29 root    root     1024 Sep  1 15:40 ..
-r-xr-xr-x  1 root    bin     218188 Aug 18 15:17 autopush
lrwxrwxrwx  1 root    root      21 Sep  1 14:11 bpgetfile -> ...
-r-xr-xr-x  1 root    bin    505556 Aug 20 13:24 dhcpagent
-r-xr-xr-x  1 root    bin    456064 Aug 20 13:25 dhcpinfo
-r-xr-xr-x  1 root    bin    272360 Aug 18 15:19 fdisk
-r-xr-xr-x  1 root    bin    824728 Aug 20 13:29 hostconfig
-r-xr-xr-x  1 root    bin    603528 Aug 20 13:21 ifconfig
-r-xr-xr-x  1 root    sys    556008 Aug 20 13:21 init
-r-xr-xr-x  2 root    root    274020 Aug 18 15:28 jsh
-r-xr-xr-x  1 root    bin    238736 Aug 21 19:46 mount
-r-xr-xr-x  1 root    sys     7696 Aug 18 15:20 mountall
.
.
```

Changing File Ownership

This section describes how to change the ownership of a file.

▼ How to Change the Owner of a File

1. Become superuser.

By default, the owner cannot use the `chown` command to change the owner of a file or directory. However, you can enable the owner to use `chown` by adding the following line to the system's `/etc/system` file and rebooting the system.

```
set rstchown = 0
```

See `chown(1)` for more details. Also, be aware that there can be other restrictions on changing ownership on NFS-mounted file systems.

2. Change the owner of a file by using the `chown` command.

```
# chown newowner filename
```

newowner Specifies the user name or UID of the new owner of the file or directory.

filename Specifies the file or directory.

3. Verify the owner of the file is changed.

```
# ls -l filename
```

Example—Changing the Owner of a File

The following example sets the ownership on `myfile` to the user `rimmer`.

```
# chown rimmer myfile  
# ls -l myfile  
-rw-r--r--  1 rimmer  scifi  112640 May 24 10:49 myfile
```

▼ How to Change Group Ownership of a File

1. Become superuser.

By default, the owner can only use the `chgrp` command to change the group of a file to a group in which the owner belongs. For example, if the owner of a file only belongs to the `staff` and `sysadm` groups, the owner can only change the group of a file to `staff` or `sysadm` group.

However, you can enable the owner to change the group of a file to a group in which the owner doesn't belong by adding the following line to the system's `/etc/system` file and rebooting the system.

```
set rstchown = 0
```

See `chgrp(1)` for more details. Also, be aware that there can be other restrictions on changing groups on NFS-mounted file systems.

2. Change the group owner of a file by using the `chgrp` command.

```
$ chgrp group filename
```

<i>group</i>	Specifies the group name or GID of the new group of the file or directory.
--------------	--

<i>filename</i>	Specifies the file or directory.
-----------------	----------------------------------

See "Managing User Accounts and Groups (Overview)" in *System Administration Guide: Basic Administration* for information on setting up groups.

3. Verify the group owner of the file is changed.

```
$ ls -l filename
```

Example—Changing Group Ownership of a File

The following example sets the group ownership on `myfile` to the group `scifi`.

```
$ chgrp scifi myfile  
$ ls -l myfile  
-rwxr-- 1 rimmer scifi 12985 Nov 12 16:28 myfile
```

Changing File Permissions

The `chmod` command enables you to change the permissions on a file. You must be superuser or the owner of a file or directory to change its permissions.

You can use the `chmod` command to set permissions in either of two modes:

- **Absolute Mode** - Use numbers to represent file permissions (the method most commonly used to set permissions). When you change permissions by using the absolute mode, represent permissions for each triplet by an octal mode number.
- **Symbolic Mode** - Use combinations of letters and symbols to add or remove permissions.

The table below lists the octal values for setting file permissions in absolute mode. You use these numbers in sets of three to set permissions for owner, group, and other (in that order). For example, the value 644 sets read/write permissions for owner, and read-only permissions for group and other.

TABLE 15-5 Setting File Permissions in Absolute Mode

Octal Value	File Permissions Set	Permissions Description
0	---	No permissions
1	--x	Execute permission only
2	-w-	Write permission only
3	-wx	Write and execute permissions
4	r--	Read permission only
5	r-x	Read and execute permissions
6	rw-	Read and write permissions
7	rwx	Read, write, and execute permissions

You can set special permissions on a file in absolute or symbolic modes. However, you cannot set or remove `setuid` permissions on a directory by using absolute mode, you must use symbolic mode. In absolute mode, you set special permissions by adding a new octal value to the left of the permission triplet. The table below lists the octal values to set special permissions on a file.

TABLE 15-6 Setting Special Permissions in Absolute Mode

Octal Value	Special Permissions Set
1	Sticky bit
2	setgid
4	setuid

The table below lists the symbols for setting file permissions in symbolic mode. Symbols can specify whose permissions are to be set or changed, the operation to be performed, and the permissions being assigned or changed.

TABLE 15-7 Setting File Permissions in Symbolic Mode

Symbol	Function	Description
u	Who	User (owner)
g	Who	Group
o	Who	Others
a	Who	All
=	Operation	Assign
+	Operation	Add
-	Operation	Remove
r	Permission	Read
w	Permission	Write
x	Permission	Execute
l	Permission	Mandatory locking, setgid bit is on, group execution bit is off
s	Permission	setuid or setgid bit is on
S	Permission	suid bit is on, user execution bit is off
t	Permission	Sticky bit is on, execution bit for others is on
T	Permission	Sticky bit is on, execution bit for others is off

The *who operator permission* designations in the function column specifies the symbols that change the permissions on the file or directory.

who Specifies whose permissions are changed.

<i>operator</i>	Specifies the operation to perform.
<i>permissions</i>	Specifies what permissions are changed.

▼ How to Change Permissions in Absolute Mode

1. If you are not the owner of the file or directory, become superuser.

Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. Change permissions in absolute mode by using the `chmod` command.

```
$ chmod mmn filename
```

<i>mmn</i>	Specifies the octal values that represent the permissions for the file owner, file group, and others, in that order. See Table 15-5 for the list of valid octal values.
------------	---

<i>filename</i>	Specifies the file or directory.
-----------------	----------------------------------

Note – If you use `chmod` to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the effective permissions for additional users and groups who have ACL entries on the file. Use the `getfacl(1)` command to make sure the appropriate permissions are set for all ACL entries.

3. Verify the permissions of the file have changed.

```
$ ls -l filename
```

Example—Changing Permissions in Absolute Mode

The following example shows changing the permissions of a public directory from 744 (read/write/execute, read-only, and read-only) to 755 (read/write/execute, read/execute, and read/execute).

```
$ ls -ld public_dir
drwxr--r-- 1 ignatz  staff    6023 Aug  5 12:06 public_dir
$ chmod 755 public_dir
$ ls -ld public_dir
drwxr-xr-x 1 ignatz  staff    6023 Aug  5 12:06 public_dir
```

The following example shows changing the permissions of an executable shell script from read/write to read/write/execute.

```
$ ls -l my_script
-rw----- 1 ignatz  staff    6023 Aug  5 12:06 my_script
$ chmod 700 my_script
$ ls -l my_script
-rwx----- 1 ignatz  staff    6023 Aug  5 12:06 my_script
```

▼ How to Change Special Permissions in Absolute Mode

1. **If you are not the owner of the file or directory, become superuser.**

Only the current owner or superuser can use the `chmod` command to change the special permissions on a file or directory.

2. **Change special permissions in absolute mode by using the `chmod` command.**

```
$ chmod nnnn filename
```

<i>nnnn</i>	Specifies the octal values that change the permissions on the file or directory. The first octal value on the left sets the special permissions on the file. See Table 15-6 for the list of valid octal values for the special permissions.
-------------	---

<i>filename</i>	Specifies the file or directory.
-----------------	----------------------------------

Note – If you use `chmod` to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the effective permissions for additional users and groups who have ACL entries on the file. Use the `getfacl(1)` command to make sure the appropriate permissions are set for all ACL entries.

3. **Verify the permissions of the file have changed.**

```
$ ls -l filename
```

Examples—Setting Special Permissions in Absolute Mode

The following example sets `setuid` permission on the `dbprog` file.

```
$ chmod 4555 dbprog
$ ls -l dbprog
-r-sr-xr-x  1 db      staff      12095 May  6 09:29 dbprog
```

The following example sets setgid permission on the dbprog2 file.

```
$ chmod 2551 dbprog2
$ ls -l dbprog2
-r-xr-s--x  1 db      staff      24576 May  6 09:30 dbprog2
```

The following example sets sticky bit permission on the public_dir directory.

```
$ chmod 1777 public_dir
$ ls -ld public_dir
drwxrwxrwt  2 ignatz  staff      512 May 15 15:27 public_dir
```

▼ How to Change Permissions in Symbolic Mode

1. **If you are not the owner of the file or directory, become superuser.**
Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. **Change permissions in symbolic mode by using the `chmod` command.**

```
$ chmod who operator permission filename
```

who operator permission *who* specifies whose permissions are changed, *operator* specifies the operation to perform, and *permission* specifies what permissions are changed. See Table 15–7 for the list of valid symbols.

filename Specifies the file or directory.

3. **Verify the permissions of the file have changed.**

```
$ ls -l filename
```

Examples—Changing Permissions in Symbolic Mode

The following example takes away read permission from others.

```
$ chmod o-r filea
```

The following example adds read and execute permissions for user, group, and others.

```
$ chmod a+rx fileb
```

The following example assigns read, write, and execute permissions to group.

```
$ chmod g=rwx filec
```

Searching for Special Permissions

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions to gain superuser privileges. A suspicious listing would be one that grants ownership of such a program to a user rather than to `root` or `bin`.

▼ How to Find Files With `setuid` Permissions

1. Become superuser.
2. Find files with `setuid` permissions set by using the `find` command.

```
# find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/filename
```

<code>find <i>directory</i></code>	Checks all mounted paths starting at the specified <i>directory</i> , which can be <code>root (/)</code> , <code>sys</code> , <code>bin</code> , or <code>mail</code> .
<code>-user root</code>	Displays files only owned by <code>root</code> .
<code>-perm -4000</code>	Displays files only with permissions set to 4000.
<code>-exec ls -ldb</code>	Displays the output of the <code>find</code> command in <code>ls -ldb</code> format.
<code>>/tmp/<i>filename</i></code>	Writes results to this file.

3. Display the results in `/tmp/filename`.

If you need background information about `setuid` permissions, see “`setuid` Permission” on page 219.

Example—Finding Files With `setuid` Permissions

```
# find / -user root -perm -4000 -exec ls -ldb {} \; > /tmp/ckprm
# cat /tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
```

```
-r-sr-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
#
```

An unauthorized user (`rar`) has made a personal copy of `/usr/bin/sh`, and has set the permissions as `setuid` to root. This means that `rar` can execute `/usr/rar/bin/sh` and become the privileged user. If you want to save this output for future reference, move the file out of the `/tmp` directory.

Executable Stacks and Security

A number of security bugs are related to default executable stacks when their permissions are set to read, write, and execute. While stacks with execute permissions set are mandated by the SPARC ABI and Intel ABI, most programs can function correctly without using executable stacks.

The `noexec_user_stack` variable (available starting in the Solaris 2.6 release) enables you to specify whether stack mappings are executable or not. By default, the variable is set to zero, except on 64-bit applications, which provides ABI-compliant behavior. If the variable is set to non-zero, the system marks the stack of every process in the system as readable and writable, but not executable.

Once this variable is set, programs that attempt to execute code on their stack are sent a `SIGSEGV` signal, which usually results in the program terminating with a core dump. Such programs also generate a warning message that includes the name of the offending program, the process ID, and real UID of the user who ran the program. For example:

```
a.out[347] attempt to execute code on stack by uid 555
```

The message is logged by the `syslogd(1M)` daemon when the `syslog kern` facility is set to `notice` level. This logging is set by default in the `syslog.conf(4)` file, which means the message is sent to both the console and to the `/var/adm/messages` file.

This message is useful for observing potential security problems, as well as to identify valid programs that depend upon executable stacks which have been prevented from correct operation by setting this variable. If the administrator does not want any messages logged, then the `noexec_user_stack_log` variable can be set to zero to

disable it in the `/etc/system` file, though the `SIGSEGV` signal can continue to cause the executing program to core dump.

You can use `mprotect(2)` if you want programs to explicitly mark their stack as executable.

Because of hardware limitations, the capability of catching and reporting executable stack problems is only available on `sun4m`, `sun4d` and `sun4u` platforms.

▼ How to Disable Programs From Using Executable Stacks

1. **Become superuser.**
2. **Edit the `/etc/system` file and add the following line.**

```
set noexec_user_stack=1
```

3. **Reboot the system.**

```
# init 6
```

▼ How to Disable Executable Stack Message Logging

1. **Become superuser.**
2. **Edit the `/etc/system` file and add the following line.**

```
set noexec_user_stack_log=0
```

3. **Reboot the system.**

```
# init 6
```

Using Access Control Lists (ACLs)

Traditional UNIX file protection provides read, write, and execute permissions for the three user classes: file owner, file group, and other. An ACL provides better file security by enabling you to define file permissions for the file owner, file group, other, specific users and groups, and default permissions for each of those categories.

For example, if you wanted everyone in a group to be able to read a file, you would simply give group read permissions on that file. Now, assume you wanted only one person in the group to be able to write to that file. Standard UNIX doesn't provide that level of file security. However, this dilemma is perfect for ACLs.

ACL entries are the way to define an ACL on a file, and they are set through the `setfacl(1)` command. ACL entries consist of the following fields separated by colons:

entry_type: [*uid* | *gid*] :*perms*

<i>entry_type</i>	Type of ACL entry on which to set file permissions. For example, <i>entry_type</i> can be <code>user</code> (the owner of a file) or <code>mask</code> (the ACL mask).
<i>uid</i>	User name or identification number.
<i>gid</i>	Group name or identification number.
<i>perms</i>	Represents the permissions that are set on <i>entry_type</i> . <i>perms</i> can be indicated by the symbolic characters <code>rxw</code> or a number (the same permissions numbers used with the <code>chmod</code> command).

The following example shows an ACL entry that sets read/write permissions for the user `nathan`.

```
user:nathan:rw-
```



Caution – UFS file system attributes such as ACLs are supported in UFS file systems only. This means that if you restore or copy files with ACL entries into the `/tmp` directory, which is usually mounted as a TMPFS file system, the ACL entries will be lost. Use the `/var/tmp` directory for temporary storage of UFS files.

ACL Entries for Files

The table below lists the valid ACL entries. The first three ACL entries provide the basic UNIX file protection.

TABLE 15-8 ACL Entries for Files

ACL Entry	Description
<code>u[ser] : :perms</code>	File owner permissions.
<code>g[roup] : :perms</code>	File group permissions.
<code>o[ther] :perms</code>	Permissions for users other than the file owner or members of file group.

TABLE 15-8 ACL Entries for Files (Continued)

ACL Entry	Description
<code>m[ask] :perms</code>	The ACL mask. The mask entry indicates the maximum permissions allowed for users (other than the owner) and for groups. The mask is a quick way to change permissions on all the users and groups. For example, the <code>mask:r--</code> mask entry indicates that users and groups cannot have more than read permissions, even though they might have write/execute permissions.
<code>u[ser] :uid:perms</code>	Permissions for a specific user. For <i>uid</i> , you can specify either a user name or a numeric UID.
<code>g[roup] :gid:perms</code>	Permissions for a specific group. For <i>gid</i> , you can specify either a group name or a numeric GID.

ACL Entries for Directories

In addition to the ACL entries described in Table 15-8, you can set default ACL entries on a directory. Files or directories created in a directory that has default ACL entries will have the same ACL entries as the default ACL entries. The table below lists the default ACL entries for directories.

When you set default ACL entries for specific users and groups on a directory for the first time, you must also set default ACL entries for the file owner, file group, others, and the ACL mask (these are required and are the first four default ACL entries in the table below).

TABLE 15-9 Default ACL Entries for Directories

Default ACL Entry	Description
<code>d[efault] :u[ser] ::perms</code>	Default file owner permissions.
<code>d[efault] :g[roup] ::perms</code>	Default file group permissions.
<code>d[efault] :o[ther] :perms</code>	Default permissions for users other than the file owner or members of the file group.
<code>d[efault] :m[ask] :perms</code>	Default ACL mask.
<code>d[efault] :u[ser] :uid:perms</code>	Default permissions for a specific user. For <i>uid</i> , you can specify either a user name or a numeric UID.

TABLE 15-9 Default ACL Entries for Directories (Continued)

Default ACL Entry	Description
<code>d[efault]:g[roup]:gid:perms</code>	Default permissions for a specific group. For <i>gid</i> , you can specify either a group name or a numeric GID.

▼ How to Set an ACL on a File

1. Set an ACL on a file by using the `setfacl` command.

```
$ setfacl -s user::perms,group::perms,other:perms,mask:perms,acl_entry_list filename ...
```

<code>-s</code>	Sets an ACL on the file. If a file already has an ACL, it is replaced. This option requires at least the file owner, file group, and other entries.
<code>user::perms</code>	Specifies the file owner permissions.
<code>group::perms</code>	Specifies the file group permissions.
<code>other:perms</code>	Specifies the permissions for users other than the file owner or members of the file group.
<code>mask:perms</code>	Specifies the permissions for the ACL mask. The mask indicates the maximum permissions allowed for users (other than the owner) and for groups.
<code>acl_entry_list</code>	Specifies the list of one or more ACL entries to set for specific users and groups on the file or directory. You can also set default ACL entries on a directory. Table 15-8 and Table 15-9 show the valid ACL entries.
<code>filename</code>	Specifies one or more files or directories on which to set the ACL.

2. To verify that an ACL was set on the file, see “How to Check If a File Has an ACL” on page 237. To verify which ACL entries were set on the file, use the `getfacl` command.

```
$ getfacl filename
```



Caution – If an ACL already exists on the file, the `-s` option will replace the entire ACL with the new ACL.

Examples—Setting an ACL on a File

The following example sets the file owner permissions to read/write, file group permissions to read only, and other permissions to none on the `ch1.doc` file. In addition, the user `george` is given read/write permissions on the file, and the ACL mask permissions are set to read/write, which means no user or group can have execute permissions.

```
$ setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:george:rw- ch1.doc
$ ls -l
total 124
-rw-r-----+ 1 nathan  sysadmin   34816 Nov 11 14:16 ch1.doc
-rw-r--r--   1 nathan  sysadmin   20167 Nov 11 14:16 ch2.doc
-rw-r--r--   1 nathan  sysadmin    8192 Nov 11 14:16 notes
$ getfacl ch1.doc
# file: ch1.doc
# owner: nathan
# group: sysadmin
user::rw-
user:george:rw-   #effective:rw-
group::r--        #effective:r--
mask:rw-
other:---
```

The following example sets the file owner permissions to read/write/execute, file group permissions to read only, other permissions to none, and the ACL mask permissions to read on the `ch2.doc` file. In addition, the user `george` is given read/write permissions; however, due to the ACL mask, the effective permissions for `george` are read only.

```
$ setfacl -s u::7,g::4,o:0,m:4,u:george:7 ch2.doc
$ getfacl ch2.doc
# file: ch2.doc
# owner: nathan
# group: sysadmin
user::rwx
user:george:rwx   #effective:r--
group::r--        #effective:r--
mask:r--
other:---
```

▼ How to Copy an ACL

Copy a file's ACL to another file by redirecting the `getfacl` output.

```
$ getfacl filename1 | setfacl -f - filename2
```

filename1 Specifies the file from which to copy the ACL.

filename2 Specifies the file on which to set the copied ACL.

Example—Copying an ACL

The following example copies the ACL on `ch2.doc` to `ch3.doc`.

```
$ getfacl ch2.doc | setfacl -f - ch3.doc
```

▼ How to Check If a File Has an ACL

Check if a file has an ACL by using the `ls` command.

```
$ ls -l filename
```

filename Specifies the file or directory.

A plus sign (+) to the right of the mode field indicates the file has an ACL.

Note – Unless you have added ACL entries for additional users or groups on a file, a file is considered to be a “trivial” ACL and the + will not display.

Example—Checking If a File Has an ACL

The following example shows that `ch1.doc` has an ACL, because the listing has a '+' to the right of the mode field.

```
$ ls -l ch1.doc
-rwxr-----+ 1 nathan sysadmin 167 Nov 11 11:13 ch1.doc
```

▼ How to Modify ACL Entries on a File

1. Modify ACL entries on a file by using the `setfacl` command.

```
$ setfacl -m acl_entry_list filename1 [filename2 ...]
```

<code>-m</code>	Modifies the existing ACL entry.
<code>acl_entry_list</code>	Specifies the list of one or more ACL entries to modify on the file or directory. You can also modify default ACL entries on a directory. Table 15–8 and Table 15–9 show the valid ACL entries.
<code>filename ...</code>	Specifies one or more files or directories.

2. To verify that the ACL entries were modified on the file, use the `getfacl` command.

```
$ getfacl filename
```

Examples—Modifying ACL Entries on a File

The following example modifies the permissions for the user `george` to read/write.

```
$ setfacl -m user:george:6 ch3.doc
$ getfacl ch3.doc
# file: ch3.doc
# owner: nathan
# group: staff
user::rw-
user::george:rw-          #effective:r--
group::r-                #effective:r--
mask:r--
other:r-
```

The following example modifies the default permissions for the group `staff` to read and the default ACL mask permissions to read/write on the `book` directory.

```
$ setfacl -m default:group:staff:4,default:mask:6 book
```

▼ How to Delete ACL Entries From a File

1. Delete ACL entries from a file by using the `setfacl` command.

```
$ setfacl -d acl_entry_list filename1 ...
```

<code>-d</code>	Deletes the specified ACL entries.
-----------------	------------------------------------

<i>acl_entry_list</i>	Specifies the list of ACL entries (without specifying the permissions) to delete from the file or directory. You can only delete ACL entries and default ACL entries for specific users and groups. Table 15-8 and Table 15-9 show the valid ACL entries.
<i>filename ...</i>	Specifies one or more files or directories.

Alternately, you can use the `setfacl -s` command to delete all the ACL entries on a file and replace them with the new ACL entries specified.

2. To verify that the ACL entries were deleted from the file, use the `getfacl` command.

```
$ getfacl filename
```

Example—Deleting ACL Entries on a File

The following example deletes the user `george` from the `ch4.doc` file.

```
$ setfacl -d user:george ch4.doc
```

▼ How to Display ACL Entries for a File

Display ACL entries for a file by using the `getfacl` command.

```
$ getfacl [-a | -d] filename1 ...
```

<code>-a</code>	Displays the file name, file owner, file group, and ACL entries for the specified file or directory.
<code>-d</code>	Displays the file name, file owner, file group, and default ACL entries for the specified directory.
<i>filename ...</i>	Specifies one or more files or directories.

If you specify multiple file names on the command line, the ACL entries are separated by a blank line.

Examples—Displaying ACL Entries for a File

The following example shows all the ACL entries for the `ch1.doc` file. The `#effective:` note beside the user and group entries indicates what the permissions are after being modified by the ACL mask.

```
$ getfacl ch1.doc

# file: ch1.doc
# owner: nathan
# group: sysadmin
user::rw-
user:george:r--          #effective:r--
group::rw-              #effective:rw-
mask:rw-
other:---
```

The following example shows the default ACL entries for the book directory.

```
$ getfacl -d book

# file: book
# owner: nathan
# group: sysadmin
user::rwx
user:george:r-x          #effective:r-x
group::rwx              #effective:rwx
mask:rwx
other:---
default:user::rw-
default:user:george:r--
default:group::rw-
default:mask:rw-
default:other:---
```


Securing Systems (Tasks)

This chapter describes the procedures for securing systems. This is a list of the step-by-step instructions in this chapter.

- “How to Display a User’s Login Status” on page 241
- “How to Display Users Without Passwords” on page 243
- “How to Temporarily Disable User Logins” on page 244
- “How to Save Failed Login Attempts” on page 244
- “How to Create a Dial-up Password” on page 247
- “How to Temporarily Disable Dial-up Logins” on page 248
- “How to Restrict Superuser (root) Login to the Console” on page 249
- “How to Monitor Who Is Using the `su` Command” on page 250
- “How to Display Superuser (root) Access Attempts to the Console” on page 250
- “How to Disable or Enable a System’s Abort Sequence” on page 251

For overview information about securing systems, see “System Security” on page 208.

Displaying Security Information

This section describes how to display user login information.

▼ How to Display a User’s Login Status

1. **Become superuser.**

2. Display a user's login status by using the `logins` command.

```
# logins -x -l username
```

<code>-x</code>	Displays an extended set of login status information.
<code>-l <i>username</i></code>	Displays login status for the specified user. <i>username</i> is a user's login name. Multiple login names must be specified in a comma-separated list.

The `logins(1M)` command uses the local `/etc/passwd` file and the NIS or NIS+ password databases to obtain a user's login status.

Example—Displaying a User's Login Status

The following example displays login status for the user `rimmer`.

```
# logins -x -l rimmer
rimmer      500      staff          10   Arnold J. Rimmer
            /export/home/rimmer
            /bin/sh
            PS 010170 10 7 -1
```

<code>rimmer</code>	Identifies the user's login name.
<code>500</code>	Identifies the UID (user ID).
<code>staff</code>	Identifies the user's primary group.
<code>10</code>	Identifies the GID (group ID).
<code>Arnold J. Rimmer</code>	Identifies the comment.
<code>/export/home/rimmer</code>	Identifies the user's home directory.
<code>/bin/sh</code>	Identifies the login shell.
<code>PS 010170 10 7 -1</code>	Specifies the password aging information: <ul style="list-style-type: none">■ Last date password was changed■ Number of days required between changes■ Number of days allowed before a change is required■ Warning period

▼ How to Display Users Without Passwords

You should make sure that all users have a valid password.

1. **Become superuser.**
2. **Display users who have no passwords by using the `logins` command.**

```
# logins -p
```

```
-p
```

Displays a list of users with no passwords.

The `logins` command uses the local `/etc/passwd` file and the NIS or NIS+ password databases to obtain a user's login status.

Example—Displaying Users Without Passwords

The following example displays that the user `pmorph` does not have a password.

```
# logins -p
pmorph      501      other          1      Polly Morph
#
```

Temporarily Disabling User Logins

You can temporarily disable user logins by:

- Creating the `/etc/nologin` file.
- Bringing the system to run level 0 (single-user mode). See “Shutting Down a System (Tasks)” in *System Administration Guide: Basic Administration* for information on bringing the system to single-user mode.

Creating the `/etc/nologin` File

Create this file to disallow user logins and notify users when a system will be unavailable for an extended period of time due to a system shutdown or routine maintenance.

If a user attempts to log in to a system where this file exists, the contents of the `nologin(4)` file is displayed, and the user login is terminated. Superuser logins are not affected.

▼ How to Temporarily Disable User Logins

1. **Become superuser.**
2. **Create the `/etc/nologin` file using an editor.**

```
# vi /etc/nologin
```
3. **Include a message regarding system availability.**
4. **Close and save the file.**

Example—Disabling User Logins

This example shows how to notify users of system unavailability.

```
# vi /etc/nologin
(Add system message here)

# cat /etc/nologin
***No logins permitted.***

***The system will be unavailable until 12 noon.***
```

Saving Failed Login Attempts

You can save failed login attempts by creating the `/var/adm/loginlog` file with read and write permission for root only. After you create the `loginlog` file, all failed login activity will be written to this file automatically after five failed attempts. See “How to Save Failed Login Attempts” on page 244 for detailed instructions.

The `loginlog` file contains one entry for each failed attempt. Each entry contains the user’s login name, tty device, and time of the failed attempt. If a person makes fewer than five unsuccessful attempts, none of the attempts are logged.

The `loginlog` file may grow quickly. To use the information in this file and to prevent the file from getting too large, you must check and clear its contents occasionally. If this file shows a lot of activity, it may suggest an attempt to break into the computer system. For more information about this file, see `loginlog(4)`.

▼ How to Save Failed Login Attempts

1. **Become superuser.**

2. Create the loginlog file in the /var/adm directory.

```
# touch /var/adm/loginlog
```

3. Set read and write permissions for root on the loginlog file.

```
# chmod 600 /var/adm/loginlog
```

4. Change group membership to sys on the loginlog file.

```
# chgrp sys /var/adm/loginlog
```

5. Make sure the log works by attempting to log into the system five times with the wrong password after the loginlog file is created. Then display the /var/adm/loginlog file.

```
# more /var/adm/loginlog
rimmer:/dev/pts/4:Mon Jul 12 13:52:15 1999
rimmer:/dev/pts/4:Mon Jul 12 13:52:23 1999
rimmer:/dev/pts/4:Mon Jul 12 13:52:31 1999
rimmer:/dev/pts/4:Mon Jul 12 13:52:39 1999
#
```

Password Protection Using Dial-up Passwords

You can add a layer of security to your password mechanism by requiring a *dial-up password* for users who access a system through a modem or dial-up port. A dial-up password is an additional password that a user must enter before being granted access to the system.

Only superuser can create or change a dial-up password. To ensure the integrity of the system, the password should be changed about once a month. The most effective use of this mechanism is to require a dial-up password to gain access to a gateway system.

Two files are involved in creating a dial-up password, /etc/dialups and /etc/d_passwd. The first contains a list of ports that require a dial-up password, and the second contains a list of shell programs that require an encrypted password as the additional dial-up password.

The dialups(4) file is a list of terminal devices, for example:

```
/dev/term/a
/dev/term/b
```

The d_passwd(4) file has two fields. The first is the login shell that will require a password, and the second is the encrypted password. The /etc/dialups and /etc/d_passwd files work like this:

When a user attempts to log in on any of the ports listed in /etc/dialups, the login program looks at the user's login entry stored in /etc/passwd, and compares the

login shell to the entries in `/etc/d_passwd`. These entries determine whether the user will be required to supply the dial-up password.

```
/usr/lib/uucp/uucico:encrypted_password:
/usr/bin/csh:encrypted_password:
/usr/bin/ksh:encrypted_password:
/usr/bin/sh:encrypted_password:
```

The basic dial-up password sequence is shown in the figure below.

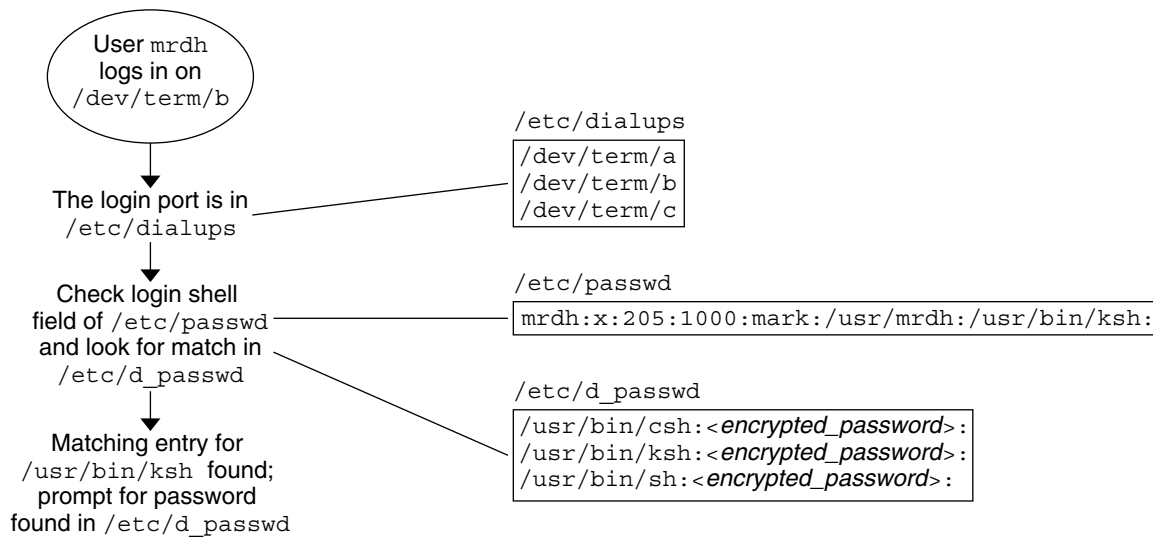


FIGURE 16-1 Basic Dial-up Password Sequence

The `/etc/d_passwd` File

Because most users will be running a shell when they log in, all shell programs should have entries in `/etc/d_passwd`. Such programs include `uucico`, `sh`, `ksh`, and `csh`. If some users run something else as their login shell, include that login shell in the file, too.

If the user's login program (as specified in `/etc/passwd`) is not found in `/etc/d_passwd`, or if the login shell field in `/etc/passwd` is null, the password entry for `/usr/bin/sh` is used.

- If the user's login shell in `/etc/passwd` matches an entry in `/etc/d_passwd`, the user must supply a dial-up password.
- If the user's login shell in `/etc/passwd` is not found in `/etc/d_passwd`, the user must supply the default password. The default password is the entry for `/usr/bin/sh`.

- If the login shell field in `/etc/passwd` is empty, the user must supply the default password (the entry for `/usr/bin/sh`).
- If `/etc/d_passwd` has no entry for `/usr/bin/sh`, then those users whose login shell field in `/etc/passwd` is empty or does not match any entry in `/etc/d_passwd` will not be prompted for a dial-up password.
- Dial-up logins are disabled if `/etc/d_passwd` has only the following entry:
`/usr/bin/sh:*:`

▼ How to Create a Dial-up Password



Caution – When you first establish a dial-up password, be sure to remain logged in on at least one terminal while testing the password on a different terminal. If you make a mistake while installing the extra password and log off to test the new password, you might not be able to log back on. If you are still logged in on another terminal, you can go back and fix your mistake.

1. **Become superuser.**
2. **Create an `/etc/dialups` file containing a list of terminal devices, including all the ports that will require dial-up password protection.**

The `/etc/dialups` file should look like this:

```
/dev/term/a
/dev/term/b
/dev/term/c
```

3. **Create an `/etc/d_passwd` file containing the login programs that will require a dial-up password, and the encrypted dial-up password.**

Include shell programs that a user could be running at login, for example, `uucico`, `sh`, `ksh`, and `csh`. The `/etc/d_passwd` file should look like this:

```
/usr/lib/uucp/uucico:encrypted_password:
/usr/bin/csh:encrypted_password:
/usr/bin/ksh:encrypted_password:
/usr/bin/sh:encrypted_password:
```

4. Set ownership to `root` on the two files.

```
# chown root /etc/dialups /etc/d_passwd
```

5. Set group ownership to `root` on the two files.

```
# chgrp root /etc/dialups /etc/d_passwd
```

6. Set read and write permissions for `root` on the two files.

```
# chmod 600 /etc/dialups /etc/d_passwd
```

7. Create the encrypted passwords.

a. Create a temporary user.

```
# useradd user-name
```

b. Create a password for the temporary user.

```
# passwd user-name
```

c. Capture the encrypted password.

```
# grep user-name /etc/shadow > user-name.temp
```

d. Edit the `user-name.temp` file.

Delete all fields except the encrypted password (the second field).

For example, in the following line, the encrypted password is `U9gp9SyA/J1Sk`.

```
temp:U9gp9SyA/J1Sk:7967::::::7988:
```

e. Delete the temporary user.

```
# userdel user-name
```

8. Copy the encrypted password from `user-name.temp` file into the `/etc/d_passwd` file.

You can create a different password for each login shell, or use the same one for each.

▼ How to Temporarily Disable Dial-up Logins

1. Become superuser.

2. Put the following entry by itself into the `/etc/d_passwd` file:

```
/usr/bin/sh:*:
```


Restricting Superuser (root) Access on the Console

The superuser account is used by the operating system to accomplish basic functions, and has wide-ranging control over the entire operating system. It has access to and can execute essential system programs. For this reason, there are almost no security restraints for any program that is run by superuser.

You can protect the superuser account on a system by restricting access to a specific device through the `/etc/default/login` file. For example, if superuser access is restricted to the console, you can log in to a system as superuser only from the console. If anybody remotely logs in to the system to perform an administrative function, they must first log in with their user login and then use the `su(1M)` command to become superuser. See the section below for detailed instructions.

Note – Restricting superuser login to the console is set up by default when you install a system.

▼ How to Restrict Superuser (root) Login to the Console

1. **Become superuser.**
2. **Edit the `/etc/default/login` file.**
3. **Uncomment the following line.**

```
CONSOLE=/dev/console
```

Any users who try to remotely log in to this system must first log in with their user login, and then use the `su` command to become superuser.

4. **Attempt to log in remotely as superuser to this system, and verify that the operation fails.**

Monitoring Who Is Using the `su` Command

You can start monitoring `su` attempts through the `/etc/default/su` file. Through this file, you can enable the `/var/adm/sulog` file to monitor each time the `su` command is used to change to another user. See “How to Monitor Who Is Using the `su` Command” on page 250 for step-by-step instructions.

The `sulog` file lists all uses of the `su` command, not only those used to switch user to superuser. The entries show the date and time the command was entered, whether or

not it was successful (+ or -), the port from which the command was issued, and finally, the name of the user and the switched identity.

Through the `/etc/default/su` file, you can also set up the system to display on the console each time an attempt is made to use the `su` command to gain superuser access from a remote system. This is a good way to immediately detect someone trying to gain superuser access on the system you are currently working on. See the section below for detailed instructions.

▼ How to Monitor Who Is Using the `su` Command

1. **Become superuser.**
2. **Edit the `/etc/default/su` file.**
3. **Uncomment the following line.**

```
SULOG=/var/adm/sulog
```

4. **After modifying the `/etc/default/su` file, use the `su` command several times and display the `/var/adm/sulog` file. You should see an entry for each time you used the `su` command.**

```
# more /var/adm/sulog
SU 12/20 16:26 + pts/0 nathan-root
SU 12/21 10:59 + pts/0 nathan-root
SU 01/12 11:11 + pts/0 root-joebob
SU 01/12 14:56 + pts/0 pmorph-root
SU 01/12 14:57 + pts/0 pmorph-root
```

▼ How to Display Superuser (root) Access Attempts to the Console

1. **Become superuser.**
2. **Edit the `/etc/default/su` file.**
3. **Uncomment the following line.**

```
CONSOLE=/dev/console
```

Use the `su` command to become root, and verify that a message is printed on the system console.

Modifying a System's Abort Sequence

Use the following procedure to disable or enable a system's abort sequence. The default system behavior is that a system's abort sequence is enabled.

Some server systems have a key switch that if set in the secure position, overrides the software keyboard abort settings, so any changes you make with the following procedure may not be implemented.

▼ How to Disable or Enable a System's Abort Sequence

1. **Become superuser.**
2. **Select one of the following to disable or enable a system's abort sequence:**
 - a. **Remove the pound sign (#) from the following line in the `/etc/default/kbd` file to disable a system's abort sequence:**

```
#KEYBOARD_ABORT=disable
```
 - b. **Add the pound sign (#) to the following line in the `/etc/default/kbd` file to enable a system's abort sequence:**

```
KEYBOARD_ABORT=disable
```
3. **Update the keyboard defaults.**

```
# kbd -i
```

Role-Based Access Control (Overview)

This chapter describes role-based access control (RBAC), a security feature for controlling access to tasks that would normally be restricted to superuser. The topics in this chapter are:

- “RBAC: Replacing the Superuser Model” on page 253
- “Solaris RBAC Elements” on page 254
- “Privileged Applications” on page 256
- “Roles” on page 257
- “Authorizations” on page 258
- “Rights Profiles” on page 258
- “Management Scope” on page 259

For information on RBAC tasks, see Chapter 18. For detailed information on the RBAC elements and tools, see Chapter 19.

RBAC: Replacing the Superuser Model

In conventional UNIX systems, `root` (also referred to as superuser) is all-powerful, with the ability to read and write to any file, run all programs, and send kill signals to any process. Effectively, anyone who can become superuser can modify a site’s firewall, alter the audit trail, read payroll and other confidential records, and shut down the entire network.

Role-based access control (RBAC) is an alternative to the all-or-nothing superuser model. RBAC uses the security principle of least privilege, which is that no user should be given more privilege than necessary for performing his or her job. RBAC allows an organization to separate superuser’s capabilities and assign them to special user accounts that are called *roles*. Roles can be assigned to specific individuals according to their job needs.

The flexibility in setting up roles enables a variety of security policies. Three recommended roles that can be easily configured are available:

- **Primary Administrator** – A powerful role equivalent to `root`
- **System Administrator** – A less strong role for administration that is not related to security
- **Operator** – A junior administrator role for operations such as backups and restores and printer management

There is no requirement that these specific roles be implemented. Roles are a function of an organization's security needs. Roles can be set up for special-purpose administrators in such areas as security, networking, or firewall administration. Another strategy is to create a single strong administrator role along with an advanced user role for those users who are permitted to fix portions of their own systems.

Solaris RBAC Elements

In the RBAC model in the Solaris operating environment, users log in as themselves and assume roles that enable them to run restricted administration tools and utilities. The RBAC model introduces these elements to the Solaris operating environment:

- **Privileged Application** — An application that can override system controls and checks for specific UIDs, GIDs, or authorizations (see “Privileged Applications” on page 256).
- **Role** — A special identity for running privileged applications that can be assumed by assigned users only.
- **Authorization** — A permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions that are otherwise prohibited by security policy.
- **Rights Profile** - A collection of overrides that can be assigned to a role or user. A rights profile can consist of authorizations, commands with set UIDs or GIDs (referred to as security attributes), and other rights profiles.

The following figure shows how the RBAC elements work together.

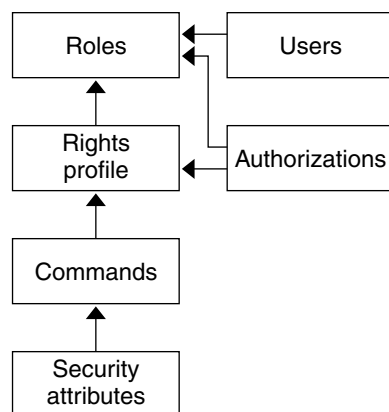


FIGURE 17-1 Solaris RBAC Element Relationships

In RBAC, users are assigned to roles. Roles get their capabilities from rights profiles and authorizations. Authorizations are generally assigned to the rights profiles with which they are logically associated but can be assigned directly to roles.

Note – Rights profiles and authorizations can also be assigned directly to users. This practice is discouraged because it enables users to make mistakes through inadvertent use of their privileges.

Commands with security attributes, that is, real or effective UIDs or GIDs, can be assigned to rights profiles.

The following figure uses the Operator role and the Printer Management rights profiles as examples to demonstrate RBAC relationships.

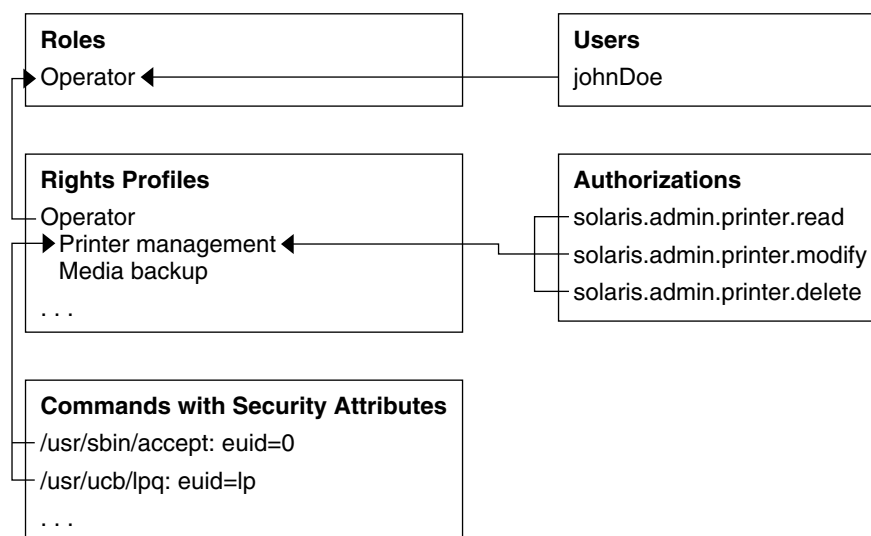


FIGURE 17-2 How Solaris RBAC Elements Relate

The Operator role is for maintaining printers and performing media backup. The user johnDoe is assigned to the Operator role and can assume it by supplying the Operator password.

The Operator rights profile has been assigned to the Operator role. The Operator profile has two supplementary profiles assigned to it, Printer Management and Media Backup, which reflect the Operator role's chief tasks.

The Printer Management rights profile is for managing printers, print daemons, and spoolers. Three authorizations are assigned to the Printer Management rights profile: `solaris.admin.printer.read`, `solaris.admin.printer.delete`, and `solaris.admin.printer.modify` for manipulating information in the printer queue. The Printer Management profile also has a number of commands with security attributes assigned to it, such as `/usr/sbin/accept` with `euid=0` and `/usr/ucb/lpq` with `euid=lp`.

Privileged Applications

Applications that can override system controls are considered to be *privileged applications*.

Applications Checking UIDs and GIDs

Privileged applications that check for `root` or some other special UID or GID have long existed in UNIX. The RBAC rights profile mechanism enables you to set the UID or GID for specific commands. Instead of setting the ID on a command that anyone can access, you can isolate the command with the set ID to a rights profile. A user or role with that rights profile can then run the program without having to become `root`.

IDs can be specified as real or effective. Assigning effective IDs is preferred over assigning real IDs. Effective IDs are equivalent to the `setuid` functionality in the file permission bits and identify the user's ID for auditing. However, because some shell scripts and programs require a real UID of `root`, real IDs can be set as well. For example, the `pkgadd` command requires a real rather than an effective UID.

Applications Checking Authorizations

RBAC additionally provides commands that check authorizations. By definition, `root` has all authorizations and thus can run any application. Currently, the applications that check for authorizations include the following:

- The entire Solaris Management Console suite of tools
- The batch job-related commands (`at(1)`, `atq(1)`, `batch(1)`, and `crontab(1)`)
- Device-oriented commands `allocate(1)`, `deallocate(1)`, `list_devices(1)`, and `cdrw(1)`.

Profile Shell

Authorized users can obtain privileged applications from the Solaris Management Console launcher or on the command line from a *profile shell*. A profile shell is a special kind of shell that enables access to the privileged applications that are assigned to the profile. Profile shells are launched when the user runs `su` to assume a role. The profile shells are `pfsh`, `pfcsh`, and `pfksh`, and they correspond to Bourne shell (`sh(1)`), C shell (`csh(1)`), and Korn shell (`ksh(1)`) respectively.

Roles

A *role* is a special type of user account from which you can run privileged applications. Roles are created in the same general manner as user accounts, with a home directory, groups, password, and so on. The capabilities of a role are a function of the rights

profiles and authorizations that are assigned to it. Roles do not have inheritance. When a user assumes a role, the role's attributes replace all user attributes. Role information is stored in the `passwd(4)`, `shadow(4)`, `user_attr(4)`, and `audit_user(4)` databases. For detailed information on setting up roles, see "Configuring Suggested Roles" on page 287, "Creating Roles" on page 274, and "Changing Role Properties" on page 277.

All users who can assume the same role have the same role home directory, operate in the same environment, and have access to the same files. Users can assume roles from the command line by running `su` and supplying the role name and password. A user can also assume a role when opening a Solaris Management Console tool. Users cannot log in directly to a role. For this reason, it is useful to make `root` a role to prevent anonymous login. See "Making Root a Role" on page 270. Users must log in to their user account first. A user cannot assume a role directly from another role. A user's real UID can always be audited.

No predefined roles are shipped with the Solaris 9 software. As stated earlier in this chapter, three roles that can be easily configured are available.

Authorizations

An *authorization* is a discrete right that can be granted to a role or user. RBAC-compliant applications can check a user's authorizations prior to granting access to the application or specific operations within it. This check replaces the check in conventional UNIX applications for `UID=0`. For more information on authorizations, see "Authorizations" on page 292, "auth_attr Database" on page 297, and "Commands Requiring Authorizations" on page 303.

Rights Profiles

A *rights profile* is a collection of system overrides that can be assigned to a role or user. A rights profile can contain commands with set effective or real UIDs or GIDs, authorizations, and other rights profiles. Rights profile information is split between the `prof_attr(4)` and `exec_attr(4)` databases. For more information on rights profiles, see "Contents of Rights Profiles" on page 288, "prof_attr Database" on page 299, and "exec_attr Database" on page 300.

Management Scope

Management scope is an important concept for understanding RBAC. The scope in which a role can operate might apply to an individual host or to all hosts that are served by a name service such as NIS, NIS+, or LDAP. The precedence of local configuration files versus distributed databases is specified in the file `/etc/nsswitch.conf`. A lookup stops at the first match. For example, if a profile exists in two scopes, only the entries in the first scope are used.

Role-Based Access Control (Tasks)

This chapter covers tasks that you can use to manage RBAC elements. To find the tasks for the initial setup of RBAC, see “Configuring RBAC (Task Map)” on page 262. For general management of the RBAC elements, see “Managing RBAC Information (Task Map)” on page 271.

The topics that are covered in this chapter include the following:

- “Planning for RBAC” on page 262
- “First-Time Use of the User Tool Collection” on page 264
- “Initial User Setup” on page 266
- “Initial Role Setup” on page 268
- “Making Root a Role” on page 270
- “Using Privileged Applications” on page 272
- “Creating Roles” on page 274
- “Changing Role Properties” on page 277
- “Creating or Changing a Rights Profile” on page 279
- “Modifying a User’s RBAC Properties” on page 283
- “Securing Legacy Applications” on page 285

The preferred approach for performing RBAC-related tasks is through the Solaris Management Console tool suite. The console tools for managing the RBAC elements are all contained in the User tool collection.

You can also operate on local files with the Solaris Management Console command-line interfaces and other command-line interfaces. The Solaris Management Console commands require authentication to connect to the server and as a result are not practical in scripts. The other commands require superuser or a role, and cannot be applied to databases in a name service.

Configuring RBAC (Task Map)

Task	Description	For Instructions, Go To ...
1. Plan for RBAC	Learn the principles behind RBAC, examine your site's security needs, and plan how to integrate RBAC into your operation.	"How to Plan Your RBAC Implementation" on page 262
2. Start the User tools from the Solaris Management Console launcher	All RBAC tasks can be performed by the User tools.	"How to Run the User Tool Collection" on page 264
3. Install initial users if needed	One or more existing users must be available for assignment to the first role.	"How to Create Initial Users Using the User Accounts Tool" on page 266
4. Install the first role	The first role, typically Primary Administrator, needs to be installed by root user.	"How to Run the User Tool Collection" on page 264
5. Make root a role (optional)	To eliminate anonymous root login, root can be made a role.	"How to Make Root a Role" on page 270

Planning for RBAC

RBAC can be an integral part of how an organization manages its information resources. Planning requires a thorough knowledge of the RBAC capabilities as well as the security requirements of the organization.

▼ How to Plan Your RBAC Implementation

1. Learn the basic RBAC concepts.

Read Chapter 17. Using RBAC to administer a system is very different from conventional UNIX. You should be familiar with the RBAC concepts before you start your implementation. For greater detail, see Chapter 19.

2. Examine your security policy.

Your organization's security policy should detail the potential threats to your system, measure the risk of each threat, and have a strategy to counter these threats. Isolating security-relevant tasks through RBAC can be a part of the strategy. Although you can install the suggested roles and their configurations as is, you might need to customize your RBAC configuration to coordinate with your security policy.

3. Decide how much RBAC your organization needs.

Depending on your security needs, you can use varying degrees of RBAC, as follows:

- **No RBAC** — You can perform all tasks as `root` user. In this instance, you log in as yourself and when you select a console tool, you type `root` as the user.
- **Root as a Role** — This technique eliminates anonymous `root` logins by preventing all users from logging in as `root`. Instead they must log in as normal users prior to assuming the `root` role. See "Making Root a Role" on page 270.
- **Single Role Only** — This approach adds the Primary Administrator role only and is similar to the superuser model.
- **Suggested Roles** — Three suggested roles that can be easily configured are available: Primary Administrator, System Administrator, and Operator. These roles are suitable for organizations with administrators at different levels of responsibility whose job capabilities fit the suggested roles.
- **Custom Roles** — You can create your own roles to meet the security requirements of your organization. The new roles can be based on existing or customized rights profiles.

4. Decide which suggested roles are appropriate for your organization.

Review the capabilities of the suggested roles and default rights profiles. Three rights profiles are available for configuring the suggested roles:

- **Primary Administrator rights profile** — for creating a role that can perform all administrative tasks, granting rights to others, and editing rights that are associated with administrative roles. A user in this role can assign the Primary Administrator role and the ability to grant rights to other users.
- **System Administrator rights profile** — for creating a role that can perform most nonsecurity administrative tasks. For example, the System Administrator can add new user accounts but cannot set passwords and cannot grant rights to other users.
- **Operator rights profile** — for creating a role that can perform simple administrative tasks, such as backup and restore and printer maintenance.

These rights profiles enable administrators to configure the suggested roles by using a single rights profile instead of having to mix and match rights profiles. To further examine these and other rights profiles, use the Rights tool to display the contents. You can also refer to "Contents of Rights Profiles" on page 288 for a summary of some major rights profiles. With the console tools, you can customize the roles and rights profiles that are provided to meet the needs of your organization.

5. Decide if any additional roles or rights profiles are appropriate for your organization.

Look for other applications or families of applications at your site that might benefit from restricted access. Applications that affect security, that can cause denial-of-service problems, or that require special administrator training are good candidates for RBAC.

a. Determine which commands are needed for the new task.

b. Decide which rights profile is appropriate for this task.

Check if an existing rights profile can handle this task or if a separate rights profile needs to be created.

c. Determine which role is appropriate for this rights profile.

Decide if the rights profile for this task should be assigned to an existing role or if a new role should be created. If you use an existing role, check that the other rights profiles are appropriate for users who are assigned to this role.

6. Decide which users should be assigned to the available roles.

According to the principle of least privilege, you should assign users to roles appropriate to their level of trust. Keeping users away from tasks they do not need to use reduces potential problems.

First-Time Use of the User Tool Collection

To install the initial users to be assigned roles, you first log in as yourself. When you authenticate yourself to the Solaris Management Console interface, specify `root` user.

▼ How to Run the User Tool Collection

1. Log in as a normal user and start the console.

```
%whoami  
johnDoe  
% /usr/sadm/bin/smc&
```


2. Navigate to the User tool collection and click the icon, as follows:

- a. Find the icon that is labeled **This Computer** under **Management Tools** in the navigation pane and click the turner icon to its left.

The turner icon is shaped like a lever. When the lever is horizontal, the contents of the folder are hidden. When the lever is vertical the contents are displayed. Clicking the turner icon toggles the folder between the hidden and displayed states.

- b. Click the turner icon next to the **System Configuration** folder to display its contents.

- c. Click the **User** icon to open the **User** tool collection.

The user login dialog box is displayed.

3. Type **root** and the root password in the user login dialog box and click **OK**.

Generally you should type your user name here and then assume a role, but the first time you need to be **root** user because no roles exist yet. This step opens the Tool collection (see the following figure).

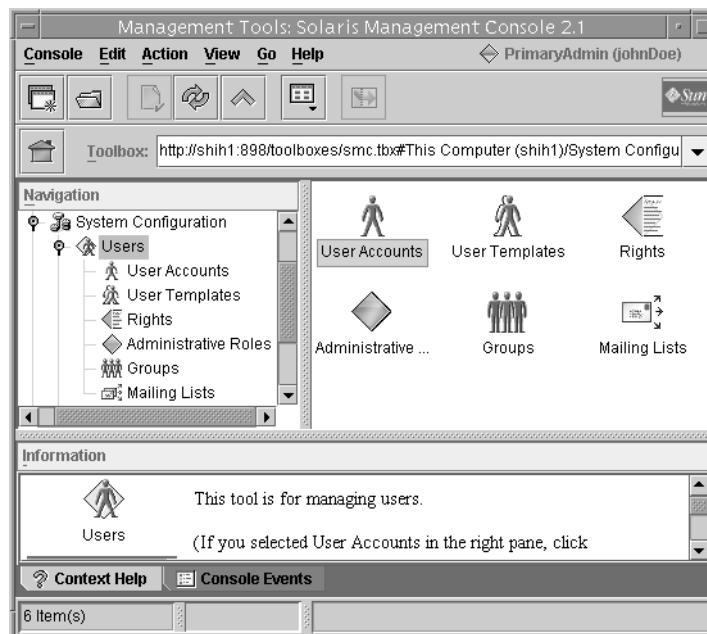


FIGURE 18–1 The User Tool Collection

Initial User Setup

If all users who are assigned to roles are already installed on this system, you can skip this task and go to "Initial Role Setup" on page 268.

▼ How to Create Initial Users Using the User Accounts Tool

1. **Click the User Accounts Tool icon in either the navigation pane or the view pane of the User Tool Collection.**

The User Accounts tool is started and the Action menu now provides options for this tool.

2. **Select Add User->With Wizard from the Action menu.**

This step starts the Add User wizard, a series of dialog boxes that request information necessary for configuring a user. Use the Next and Back buttons to navigate between dialog boxes. Note that the Next button does not become active until all required fields have been filled in. The last dialog box is for reviewing the entered data, at which point you can go back to change entries or click Finish to save the new role. The first dialog box "Step 1: Enter a user name" is shown in the following figure.

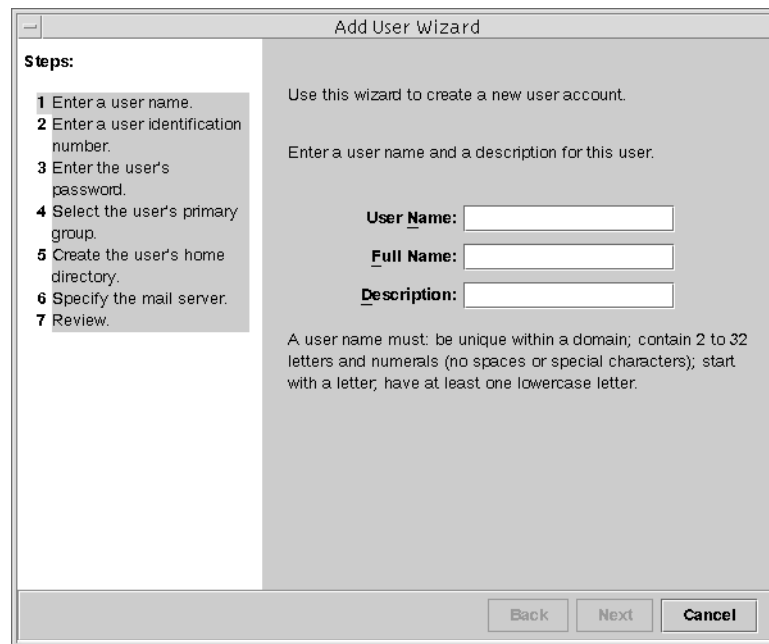


FIGURE 18-2 Add User Wizard

3. **Type the name of the first user and the other identification information.**
4. **In the Step 2: Enter a User Identification Number dialog box, type the user's UID.**
This should match the existing UID for the user.
5. **In the Step 3: Enter the User's Password dialog box, indicate whether you or the user is to set the password.**
If you are setting up this account for yourself, click the second option, and type and confirm your password.
6. **In the Step 4: Select the User's Primary Group dialog box, select the appropriate group.**
7. **In the Step 5: Create the User's Home Directory dialog box, specify the path for the home directory.**
8. **In the Step 6: Specify the Mail Server dialog box, check out the default mail server and mailbox.**
You can change these settings later in the User Properties dialog box.

9. **Check the information in the Review dialog box. Click Finish to save or Back to reenter information.**

If you discover missing or incorrect information, click the Back button successively to display the dialog box where the incorrect information is displayed. Then click Next repeatedly to return to the Review dialog box.

Initial Role Setup

The first role to be created is the one responsible for managing users and roles, typically the Primary Administrator. You should install the users and the roles first on your local host. After you have set up a toolbox for the name service scope, you need to create the same users and roles in the name service. See “Using the Solaris Management Console Tools in a Name Service Environment (Task Map)” in *System Administration Guide: Basic Administration*. After the first role is established and assigned to you, you can then run the console tools by assuming a role instead of becoming `root`.

▼ How to Create the First Role (Primary Administrator) Using the Administrative Roles Tool

To install the first role, you should log in as yourself. When you authenticate yourself to the Solaris Management Console interface, specify `root` user. You should install the role first on your local host. After the first role is established and assigned to you, you will be able to run the console tools assuming a role instead of as `root` user.

1. **Type `root` and the root password in the user login dialog box and click OK.**
2. **Click the Administrative Roles icon in either the navigation pane and the view pane of the User Tool Collection.**

The Administrative Roles tool is started and the Action menu now provides options for this tool.

3. **Select Add Administrative Role from the Action menu.**

This step starts the Add Administrative Role wizard, a series of dialog boxes that request information necessary for configuring a role. Use the Next and Back buttons to navigate between dialog boxes. Note that the Next button does not become active until all required fields have been filled in. The last dialog box is for reviewing the entered data, at which point you can go back to change entries or click Finish to save

the new role. The first dialog box Step 1: Enter a Role Name follows.

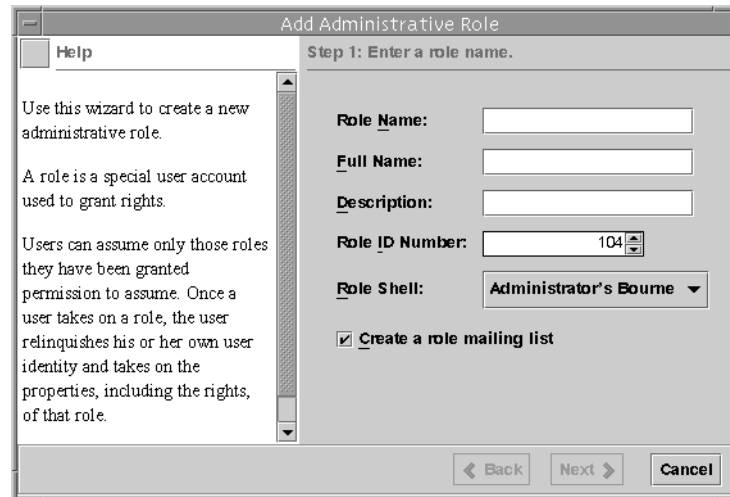


FIGURE 18-3 Add Administrative Role Wizard

4. **Type `primaryadmin` or whatever role name you are using and the other identification information.**
If you select mailing list, you can create an alias of users who can assume this role.
5. **In the Step 2: Enter a Role Password dialog box, type the password for the new role in the Role Password field and again in the Confirm Password field.**
Confirmation helps prevent a misspelled password from being saved.
6. **In the Step 3: Enter Role Rights dialog box, select the Primary Administrator rights profile.**
Double-click the Primary Administrator rights profile in the Available Rights column (on the left). The rights profiles in the Granted Rights column (on the right) are the ones that are assigned to this role. In this instance, only the Primary Administrator rights profile is needed.
7. **In the Step 4: Select a Home Directory dialog box, specify the server and path for the home directory.**
8. **In the Step 5: Assign Users to This Role dialog box, type the login names for any users to be assigned to the Primary Administrator role.**
Any users you add must be defined in the same scope in which you are working. If you selected the email alias in the Step 1: Enter a Role Name dialog box, these users will receive email that is addressed to the Primary Administrator role.

9. Check the information in the Review dialog box. Click Finish to save or Back to reenter information.

If you discover missing or incorrect information, click the Back button successively to display the dialog box where the incorrect information is displayed. Then click Next repeatedly to return to the Review dialog box.

Making Root a Role

This procedure shows how to change `root` from a user to a role within a local scope. Changing `root` to a role prevents users from logging in to that server directly as `root`. Users must first log in as themselves so their UIDs are available for auditing.



Caution – If you make `root` a role without assigning it to a valid user or without a currently existing role equivalent to `root`, no one can become `root`.

▼ How to Make Root a Role

1. Log in to the target server.
2. Become superuser.
3. Edit the `/etc/user_attr` file.

Here is an excerpt from a typical `user_attr` file.

```
root:::type=normal;auths=solaris.*,solaris.grant;profiles=All
johnDoe:::type=normal
```

4. Check that your name is in the file.
5. Add `root` to the roles that are assigned to your record.

Assign the `root` role to any applicable users. If you intend to use `primaryadmin` as your most powerful role, you do not have to assign `root` to any users.

```
johnDoe:::type=normal;roles=root
```

6. Go to the `root` record in the file and change `type=normal` to `type=root`.

```
root:::type=role;auths=solaris.*,solaris.grant;profiles=All
```

7. Save the file.

Managing RBAC Information (Task Map)

The following table shows where to obtain information for performing specific RBAC tasks.

Task	Description	For Instructions, Go To ...
Using privileged applications	To run applications that can affect security or system operations requires becoming superuser or assuming a role.	"How to Assume a Role at the Command Line" on page 272
		"How to Assume a Role in the Console Tools" on page 273
Creating roles	To add new roles, that is, special identities for running privileged applications.	"How to Create a Role Using the Administrative Roles Tool" on page 274
		"How to Create a Role From the Command Line" on page 275
Changing role properties	To change the properties of a role, that is, the assigned users, rights profiles, and authorizations that are assigned to a role.	"How to Change a Role Using the Administrative Roles Tool" on page 277
		"How to Change a Role From the Command Line" on page 278
Creating or changing rights profiles	To add or change a rights profile, including the assignment of authorizations, commands with security attributes, and supplementary rights profiles.	"How to Create or Change a Rights Profile Using the Rights Tool" on page 279
		"How to Change Rights Profiles From the Command Line" on page 283
Changing a user's RBAC properties	To change the roles, rights profiles, or authorizations that are assigned to a user.	"How to Modify a User's RBAC Properties Using the User Accounts Tool" on page 284
		"How to Modify a User's RBAC Properties From the Command Line" on page 284

Task	Description	For Instructions, Go To ...
Securing legacy applications	Legacy applications can be run with a set ID. Scripts can contain commands with set IDs. Legacy applications can check for authorizations, if appropriate.	"How to Add Security Attributes to a Legacy Application" on page 285
		"How to Add Security Attributes to Commands in a Script" on page 285
		"How to Check for Authorizations in a Script or Program" on page 285

These procedures are for managing the elements that are used in role-based access control (RBAC). For user management procedures, refer to "Setting Up User Accounts and Groups (Tasks)" in *System Administration Guide: Basic Administration*.

Using Privileged Applications

To run privileged applications, you must first become superuser or assume a role. Although running privileged applications as a normal user is possible, it is discouraged to avoid errors that are caused by users inadvertently exercising privilege.

▼ How to Assume a Role at the Command Line

1. Use the `su(1M)` command as follows:

```
%su my-role
Password: my-role-password
#
```

Typing `su` by itself lets you become superuser. Typing `su` with a role name lets you assume that role (if it has been assigned to you). You must supply the appropriate password. Assuming a role switches the command line to the profile shell for that role. The profile shell has been modified to run commands with the security attributes that are assigned in the role's rights profiles.

2. Type the command in the shell.

The command is executed with any assigned security attributes and `setuid` or `setgid` permissions.

▼ How to Assume a Role in the Console Tools

1. Start the Solaris Management Console launcher.

Use one of the following methods:

- **Type `smc` at the command line.**
- **Click the Solaris Management Console icon in the Tools subpanel.**
- **Double-click the Solaris Management Console icon in the Application Manager.**

All Solaris Management Console tools have extensive context-sensitive help that document each field. In addition, you can access various help topics that from the Help menu. Note that it does not matter whether you are logged in as `root` or as a normal user when you start the console.

2. Select the toolbox for your task.

Navigate to the toolbox that contains the tool or collection in the appropriate scope and click the icon. The scopes are files (local), NIS, NIS+, and LDAP. If the appropriate toolbox is not displayed in the navigation pane, choose Open Toolbox from the Console menu and load the relevant toolbox.

3. Select the tool.

Navigate to the tool or collection to be used and click the icon. The tools for managing the RBAC elements are all part of the User tool collection.

4. Authenticate yourself in the Login: User Name dialog box.

Your choices are the following:

- **Type your user name and password to assume a role or to operate as a normal user.**
- **Type `root` and the root password to operate as superuser.**

Note that if you have not yet set up any roles or if the roles that are set up cannot perform the appropriate tasks, you need to log in as `root`. If you authenticate yourself as `root` (or as a user with no roles assigned), the tools are loaded into the console and you can proceed to step 6.

5. Authenticate yourself in the Login: Role dialog box.

The Role option menu in the dialog box displays the roles that are assigned to you. Choose a role and type the role password. If you are to operate as a normal user, type your user name and password.

6. Navigate to the tool to be run and click the icon.

Creating Roles

To create a role, you must either assume a role that has the Primary Administrator rights profile assigned to it or run as `root` user. See “Roles” on page 257 and “Configuring Suggested Roles” on page 287 to learn more about roles.

▼ How to Create a Role Using the Administrative Roles Tool

1. Start the Administrative Roles tool.

To run the Administrative Roles tool, start the Solaris Management Console launcher as described in “How to Assume a Role in the Console Tools” on page 273, open the User tool collection, and click the Administrative Roles icon.

2. Start the Add Administrative Role wizard.

Select Add Administrative Role from the Action menu to start the Add Administrative Role wizard series for configuring roles.

3. Fill in the fields in the series of dialog boxes and click Finish when done.

Use the Next and Back buttons to navigate between dialog boxes. Note that the Next button does not become active until all required fields have been filled in. The last dialog box is for reviewing the entered data, at which point you can go back to change entries or click Finish to save the new role. The dialog boxes are summarized in the following table.

Dialog Box	Fields	Field Description
Step 1. Enter a role name.	Role Name	The short name of the role.
	Full Name	Long version of the name.
	Description	Description of the role.
	Role ID Number	UID for the role, automatically incremented.
	Role Shell	The profile shells available to roles: Administrator's C, Administrator's Bourne, or Administrator's Korn shell.
	Create a role mailing list	Makes a mailing list for users who are assigned to this role.

Dialog Box	Fields	Field Description
Step 2. Enter a role password	Role Password	*****
	Confirm Password	*****
Step 3. Select role rights.	Available Rights / Granted Rights	<p>Assigns or removes a role's rights profiles.</p> <p>Note that the system does not prevent you from typing multiple occurrences of the same command. The attributes that are assigned to the first occurrence of a command in a rights profile have precedence and all subsequent occurrences are ignored. Use the Up and Down arrows to change the order.</p>
Step 4. Select a home directory.	Server	Server for the home directory.
	Path	Home directory path.
Step 5. Assign users to this role.	Add	Users who can assume this role. Must be in the same scope.
	Delete	For deleting users who are assigned to this role.

▼ How to Create a Role From the Command Line

1. **Become superuser or assume a role capable of creating other roles.**
2. **Select a method for creating a role:**
 - **For roles in the local scope, use the `roleadd` command to specify a new local role and its attributes.**
 - **Alternatively, for roles in the local scope, edit the `user_attr` file to add a user with `type=role`.**

This method is recommended for emergencies only, as it is easy to make mistakes while typing.
 - **For roles in a name service, use the `smrole` command to specify the new role and its attributes.**

This command requires authentication by superuser or a role capable of creating other roles. You can apply the `smrole(1M)` to all name services. This command runs as a client of the Solaris Management Console server.

EXAMPLE 18-1 Creating a Custom Operator Role Using `smrole(1M)`

The following sequence demonstrates how a role is created with the `smrole` command. In this instance, a new version of the operator role is created that has the standard Operator rights profile assigned and additionally the Media Restore rights profile.

```
% su primaryadmin
# /usr/sadm/bin/smrole add -H myHost -- -c "Custom Operator" -n oper2 -a johnDoe \
-d /export/home/oper2 -F "Backup/Restore Operator" -p "Operator" -p "Media Restore"
Authenticating as user: primaryadmin

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::      <enter primaryadmin password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadmin was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::      <enter oper2 password>
```

To view the newly created role (and any other roles), use `smrole` with `list` subcommand, as follows:

```
# /usr/sadm/bin/smrole list --
Authenticating as user: primaryadmin

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::      <enter primaryadmin password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadmin was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.
root                0                Super-User
primaryadmin        100              Most powerful role
sysadmin            101              Performs non-security admin tasks
oper2                102              Backup/Restore Operator
```

Changing Role Properties

To change a role, you must either assume a role that has the Primary Administrator rights profile that is assigned to it, or run the User tool collection as `root` user if roles have not yet been set up.

▼ How to Change a Role Using the Administrative Roles Tool

1. Start the Administrative Roles tool.

To run the Administrative Roles tool, you need to start the Solaris Management Console launcher as described in “How to Assume a Role in the Console Tools” on page 273, open the User tool collection, and click the Administrative Roles icon.

The icons for the existing roles are displayed in the view pane after the Administrative Roles tool starts.

2. Click the role to be changed and select the appropriate item from the Action menu, as follows:

■ To change users who are assigned to a role, select Assign Administrative Role.

The Assign Administrative Role dialog box is displayed. The Assign Administrative Role dialog box is a modified version of the Role Properties dialog box that has a Users tab only. Use the Add field to assign a user in the current scope to this role. Use the Delete field to remove a user’s role assignment. Click OK to save.

■ To change rights that are assigned to a role, select Assign Rights to Role.

The Assign Rights to Role dialog box is displayed. The Assign Rights to Role dialog box is a modified version of the Role Properties dialog box that has a Rights tab only. Use the Available Rights and Granted Rights columns to add or remove rights profiles for the selected role. Click OK to save.

■ To change any of the role’s properties, select Properties (or simply double-click the role icon).

The Role Properties dialog box is displayed, which provides access to all role properties (see the following figure and table). Use the tabs to navigate to any information to be changed, make your changes, and click OK to save.

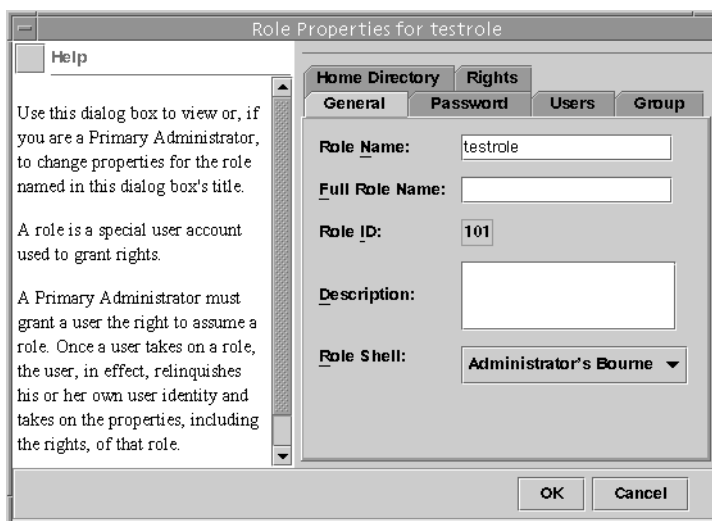


FIGURE 18-4 Role Properties Dialog Box

TABLE 18-1 Role Properties Summary

Tab	Tab Description
General	Specifies the role identification information and the default login shell.
Password	Specifies the role password.
Users	Specifies users who are assigned to the role.
Group	Sets the role's primary and secondary groups for the purpose of accessing and creating files and directories.
Home Directory	Specifies the role's home directory, home directory server, automounting, and directory access.
Rights	Allows rights profiles to be assigned to the role. The precedence of the assigned rights profiles can be changed.

▼ How to Change a Role From the Command Line

1. Become superuser or assume a role capable of changing other roles.
2. Use the command appropriate for the task:
 - Use the `rolemod` command to modify the attributes of a role that are defined locally.
 - Use the `roledel` command to delete a role that is defined locally.

- **Edit the `user_attr` file to change the authorizations or rights profiles that are assigned to a local role.**

This method is recommended for emergencies only, as it is easy to make a mistake while typing.

- **Use the `smrole` command to modify the attributes of a role in a name service.**

This command requires authentication as superuser or as a role capable of changing other roles. The `smrole` command runs as a client of the Solaris Management Console server.

Creating or Changing a Rights Profile

To create or change a rights profile, you must either assume a role that has the Primary Administrator rights profile assigned to it, or run the User tool collection as `root` user if roles have not yet been set up. See “Roles” on page 257 and “Configuring Suggested Roles” on page 287 to learn more about rights profiles.

▼ How to Create or Change a Rights Profile Using the Rights Tool

1. Start the Rights tool.

To run the Rights tool, you need to start the Solaris Management Console launcher as described in “How to Assume a Role in the Console Tools” on page 273, open the User tool collection, and click the Rights icon.

The icons for the existing rights profiles are displayed in the view pane after the Rights tool starts.

2. Take the appropriate action for creating or changing a rights profile:

- **To create a new rights profile, select Add Right from the Action menu.**
- **To change an existing rights profile, click the rights profile icon and select Properties from the Action menu (or simply double-click the rights profile icon).**

Both actions display a version of the Rights Properties dialog box. The Add Right version (which follows) has a writable Name field. The standard Rights Properties dialog box has a read-only Name field because the name of rights profile cannot be changed after it has been defined.

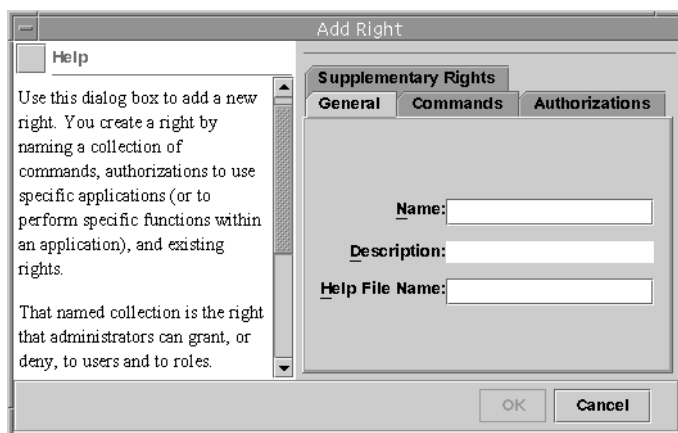


FIGURE 18-5 Add Right Dialog Box

3. Type the new information and click OK to save the rights profile.

The tabs and fields in the Right Properties dialog box are listed in the following table.

Tab	Field	Field Description
General	Name	Name of the new rights profile.
	Description	Description of the new rights profile.
	Help File Name	Name of the HTML help file for the new rights profile.
Commands	Add Directory	Opens a dialog box for adding directories not already in the Commands Denied or Commands Permitted columns.
	Commands Denied / Commands Permitted	Assigns or removes a rights profile's commands.

Tab	Field	Field Description
	Set Security Attributes	<p>Opens a dialog box for assigning or removing a command's security attributes, that is, real or effective UIDs or GIDs (see Figure 18-6).</p> <p>Note – Assigning effective IDs is preferred over assigning real IDs. Use real IDs only where required by the command, such as <code>pkgadd(1M)</code>.</p>
	Find (command)	Searches the two command lists for the particular string.
Authorizations	Authorizations Excluded / Authorizations Included	Assigns or removes a rights profile's authorizations.
Supplementary Rights	Rights Excluded / Rights Included	Assigns or removes a rights profile's supplementary rights profiles.

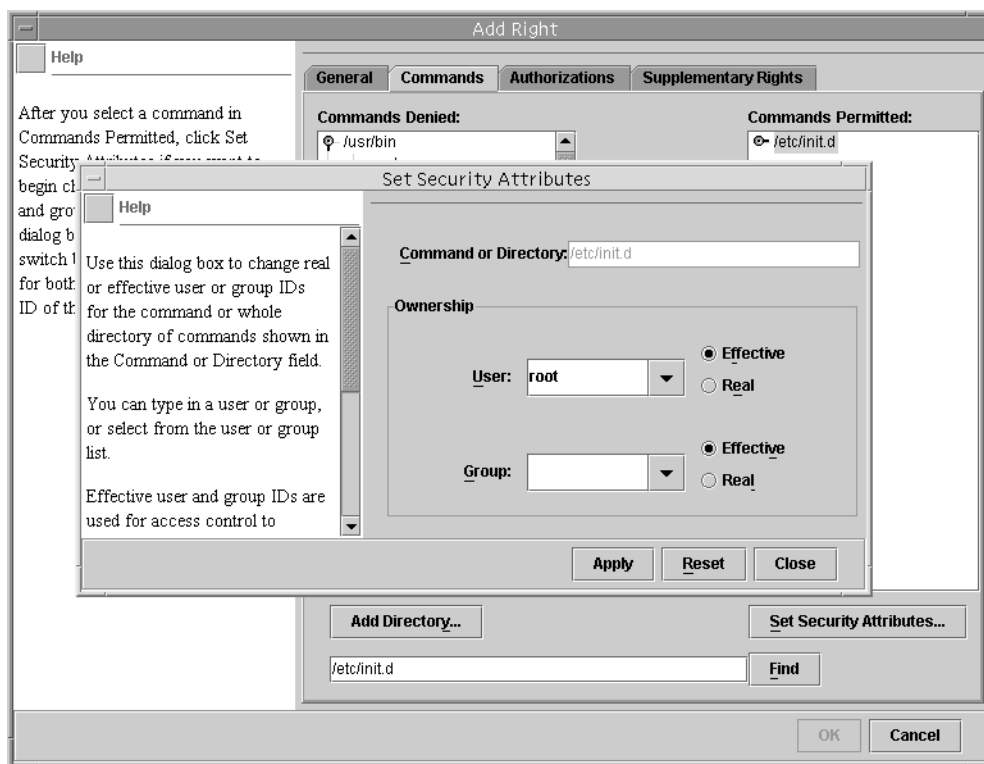


FIGURE 18-6 Adding Security Attributes to Commands

EXAMPLE 18-2 Creating a New Rights Profile With the Rights Tool

The data in the following table show how a rights profile that is called “Restart” could be created. The example rights profile, Restart, has the commands in the subdirectory `/etc/init.d` assigned to it. These commands have an effective UID of 0. This rights profile would be useful for administrators who are permitted to stop and start the daemons in `/etc/init.d`.

Tab	Field	Example
General	Name	Restart
	Description	For starting and stopping daemons in <code>/etc/init.d</code> .
	Help File Name	Restart.html

EXAMPLE 18–2 Creating a New Rights Profile With the Rights Tool (Continued)

Tab	Field	Example
Commands	Add Directory	Click Add Directory, type <code>/etc/init.d</code> in the dialog box, and click OK.
	Commands Denied / Commands Permitted	Select <code>/etc/init.d</code> and click Add to move the command to the Commands Permitted column.
	Set Security Attributes	Select <code>/etc/init.d</code> , click Set Security Attributes, and set Effective UID = root (see Figure 18–6).
	Find (command)	
Authorizations	Authorizations Excluded / Authorizations Included	
Supplementary Rights	Rights Excluded / Rights Included	

▼ How to Change Rights Profiles From the Command Line

1. **Become superuser or assume a role with the PrimaryAdmin rights profile.**
2. **Use the subcommand of `smprofile(1M)` appropriate for the task.**

This command requires authentication. You can apply the command to all name services. `smprofile` runs as a client of the Solaris Management Console server.

 - **To add a new profile, use `smprofile` with the `add` subcommand.**
 - **To change an existing profile, use `smprofile` with the `modify` subcommand.**

Modifying a User's RBAC Properties

To modify a user's properties, you must either be running the User tool collection as root user or assume a role that has the Primary Administrator rights profile assigned to it.

▼ How to Modify a User's RBAC Properties Using the User Accounts Tool

1. Start the User Accounts tool.

To run the User Accounts tool, you need to start the Solaris Management Console launcher as described in "How to Assume a Role in the Console Tools" on page 273, open the User tool collection, and click the User Accounts icon.

The icons for the existing user accounts are displayed in the view pane after the User Accounts tool starts.

2. Click the user account icon to be changed and select Properties from the Action menu (or simply double-click the user account icon).

3. Click the appropriate tab in the dialog box for the property to be changed, as follows:

- To change the roles that are assigned to the user, click the Roles tab and move the role assignment to be changed to the appropriate column: Available Roles or Assigned Roles.
- To change the rights profiles assigned to the user, click the Rights tab and move it to the appropriate column: Available Rights or Assigned Rights.

Note that it is not good practice to assign rights profiles directly to users. The preferred approach is to force users to assume roles in order to perform privilege applications. This strategy avoids the possibility of normal users abusing privileges.

▼ How to Modify a User's RBAC Properties From the Command Line

1. Become superuser or assume a role that can modify user files.

2. Use the appropriate command:

- To change the authorizations, roles, or rights profiles that are assigned to a user who is defined in the local scope, use the `usermod` command.
- Alternatively, to change the authorizations, roles, or rights profiles that are assigned to a user who is defined in the local scope, edit the `user_attr` file.

This method is recommended for emergencies only, as it is easy to make a mistake while typing.

- To change the authorizations, roles, or rights profiles that are assigned to a user who is defined in a name service, use the `smuser` command.

This command requires authentication as superuser or as a role capable of changing user files. You can apply `smuser` to all name services. `smuser` runs as a

client of the Solaris Management Console server.

Securing Legacy Applications

This section discusses how to make legacy applications more secure. To add legacy applications, see “Adding Tools to the Solaris Management Console” in *System Administration Guide: Basic Administration*.

▼ How to Add Security Attributes to a Legacy Application

You add security attributes to a legacy application in the same way as you would for any command. You need to add the command (or its directory) to the Commands Denied column in the Commands tab of the Rights property dialog box. Then move the command to the Commands Permitted column.

▼ How to Add Security Attributes to Commands in a Script

If a command in a script needs set ID to run, simply add the security attributes to that command in the same rights profile. See “How to Create or Change a Rights Profile Using the Rights Tool” on page 279.

▼ How to Check for Authorizations in a Script or Program

To have a script for authorizations, you need to add a test that is based on the `auths(1)` command (see the `auths(1)` man page). For example, the following line would test if the user has the authorization entered as the `$1` argument:

```
if [ ` /usr/bin/auths|/usr/xpg4/bin/grep $1 ` ]; then
    echo Auth granted
else
    echo Auth denied
fi
```

To be more complete, the test should include logic that checks for other authorizations that use wildcards. For example, to test if the user has the `solaris.admin.usermgr.write` authorization, you need to check for the strings: `solaris.admin.usermgr.write`, `solaris.admin.usermgr.*`, `solaris.admin.*`, and `solaris.*`.

If you are writing a program, use the `getauthattr(3SECDB)` function to test for the authorization.

Role-Based Access Control (Reference)

This chapter provides additional information that supplements Chapter 17.

The topics in the reference section are:

- “Configuring Suggested Roles” on page 287
- “Contents of Rights Profiles” on page 288
- “Authorizations” on page 292
- “Overview of Databases Supporting RBAC” on page 293
- “user_attr Database” on page 296
- “auth_attr Database” on page 297
- “prof_attr Database” on page 299
- “exec_attr Database” on page 300
- “Command-Line Applications for Managing RBAC” on page 301
- “Commands Requiring Authorizations” on page 303

For information on RBAC tasks, see Chapter 18.

RBAC Elements: Reference Information

Configuring Suggested Roles

No predefined roles are shipped with the Solaris 9 software. Management at a customer site must decide what types of roles should be set up. However, the three recommended roles can be readily configured by assigning the appropriate predefined rights profile to the corresponding roles:

- **Primary Administrator rights profile** — For creating a role that can perform all administrative tasks, granting rights to others, and editing rights that are associated with administrative roles. Can assign the Primary Administrator role and the ability to grant rights to other users.
- **System Administrator rights profile** — For creating a role that can perform most non-security administrative tasks. For example, the System Administrator can add new user accounts but cannot set passwords and cannot grant rights to other users.
- **Operator rights profile** — For creating a role that can perform simple administrative tasks, such as backup and restore and printer maintenance.

These rights profiles enable administrators to configure the suggested roles by using a single rights profile instead of having to mix and match rights profiles.

Those sites that customize roles should closely check the order of the rights profiles that are assigned to the role. The system does not prevent someone from typing multiple occurrences of the same command. The attributes that are assigned to the first occurrence of a command in a rights profile take precedence and all subsequent occurrences are ignored.

Note – You can also set up `root` as a role through a manual process. This approach prevents users from logging in directly as `root`, forcing them to log in as themselves first. See “Making Root a Role” on page 270.

Contents of Rights Profiles

This section describes some typical rights profiles to demonstrate the following:

- The All rights profile provides a role access to commands without security attributes.
- The Primary Administrator rights profile is designed specifically for the Primary Administrator role. The Primary Administrator rights profile demonstrates the use of wildcards.
- The System Administrator rights profile is designed specifically for the System Administrator role. The System Administrator rights profile uses discrete supplementary profiles to create a powerful role.
- The Operator rights profile is designed specifically for the Operator role. The Operator rights profile uses a few discrete supplementary profiles to create a simple role.
- The Basic Solaris User rights profile shows how the `policy.conf` file can be used to assign users tasks that are not related to security.
- The Printer Management rights profile is an example of a profile that is dedicated to a single area of administration.

The contents of these rights profiles are displayed in tables in this section. The tables label the purpose, authorizations, commands, supplementary rights profiles, and help files that are assigned. Help files are in HTML and can be readily customized if required. These files reside in the `/usr/lib/help/auths/locale/C` directory. The Solaris Management Console Rights tool provides another way of inspecting the contents of rights profiles.

All Rights Profile

The All rights profile uses the wildcard to include all commands but with no security attributes. This profile is intended to provide a role access to all commands that are not explicitly assigned in other rights profiles. Without the All rights profile or some other rights profiles that use wildcards, a role has access to explicitly assigned commands only, which is not very practical.

Because commands in rights profiles are interpreted in the order in which they occur, any wildcard settings should be positioned last so that explicit attribute assignments are not inadvertently overridden. The All profile, if used, should be the final profile that is assigned.

TABLE 19-1 All Rights Profile Contents

Purpose	Contents
Execute any command as the user or role	Commands: * Help File: RtAll.html

Primary Administrator Rights Profile

The Primary Administrator rights profile is intended to be assigned to the most powerful role on the system, effectively providing that role with superuser capabilities. The `solaris.*` authorization effectively assigns all of the authorizations that are provided by the Solaris software. The `solaris.grant` authorization lets a role assign any authorization to any rights profile, role, or user. The command assignment `*:uid=0;gid=0` provides the ability to run any command with UID=0 and GID=0. The help file `RtPriAdmin.html` is identified so that a site can modify it if necessary. Help files are stored in the `/usr/lib/help/auths/locale/C` directory. Note also that if the Primary Administrator rights profile is not consistent with a site's security policy, it can be modified or not assigned at all. However, the security capabilities in the Primary Administrator rights profile would need to be handled in one or more other rights profiles.

TABLE 19-2 Primary Administrator Rights Profile Contents

Purpose	Contents
Can perform all administrative tasks	Commands: * Authorizations: solaris.*, solaris.grant Help File: RtPriAdmin.html

System Administrator Rights Profile

The System Administrator rights profile is intended for the System Administrator role. Because the System Administrator does not have the broad powers of the Primary Administrator, no wildcards are used. Instead, discrete administrative rights profiles that do not deal with security are assigned. The commands that are assigned to the supplementary rights profiles are not shown in this example.

Notice that the All rights profile is assigned at the end of the list of supplementary rights profiles that are assigned to the System Administrator.

TABLE 19-3 System Administrator Rights Profile Contents

Purpose	Contents
Can perform most nonsecurity administrative tasks	Supplementary rights profiles: Audit Review, Printer Management, Cron Management, Device Management, File System Management, Mail Management, Maintenance and Repair, Media Backup, Media Restore, Name Service Management, Network Management, Object Access Management, Process Management, Software Installation, User Management, All Help File: RtSysAdmin.html

Operator Rights Profile

The Operator rights profile is a less powerful administrative rights profile that provides the ability to do backups and printer maintenance. The ability to restore files has more security consequences and the default is not to assign it to this rights profile.

TABLE 19-4 Operator Rights Profile Contents

Purpose	Contents
Can perform simple administrative tasks	Supplementary rights profiles: Printer Management, Media Backup, All Help File: RtOperator.html

Basic Solaris User Rights Profile for User

The Basic Solaris User rights profile is assigned by default to all users automatically through the `policy.conf` file. This profile provides basic authorizations useful in normal operation. Note that the convenience offered by the Basic Solaris User rights profile must be balanced against the site security requirements. Those sites that need stricter security may prefer to remove this profile from the `policy.conf` file.

TABLE 19-5 Basic Solaris User Rights Profile Contents

Purpose	Contents
Provides automatically assigned rights to all users	Authorizations: <code>solaris.profmgr.read</code> , <code>solaris.admin.usermgr.read</code> , <code>solaris.admin.logsvc.read</code> , <code>solaris.admin.fsmgr.read</code> , <code>solaris.admin.serialmgr.read</code> , <code>solaris.admin.diskmgr.read</code> , <code>solaris.admin.procmgr.user</code> , <code>solaris.compsys.read</code> , <code>solaris.admin.printer.read</code> , <code>solaris.admin.prodreg.read</code> , <code>solaris.admin.dcmgr.read</code> Supplementary rights profiles: All Help File: RtDefault.html

Printer Management Rights Profile

Printer Management is a typical rights profile that is intended for a specific task area. Both authorizations and commands are assigned to the Printer Management rights profile. The following table shows a partial list of commands.

TABLE 19-6 Printer Management Rights Profile Contents

Purpose	Contents
Manage printers, daemons, and spooling	<p>Authorizations: solaris.admin.printer.delete, solaris.admin.printer.modify, solaris.admin.printer.read</p> <p>Commands: /usr/sbin/accept:euid=lp, /usr/ucb/lpq:euid=0, /etc/init.d/lp:euid=0, /usr/bin/lpstat:euid=0, /usr/lib/lp/lpsched:uid=0, /usr/sbin/lpfilter:euid=lp, ...</p> <p>Help File: RtPrntMngmnt.html</p>

Authorizations

An *authorization* is a discrete right that can be granted to a role or user. Authorizations are checked by RBAC-compliant applications before a user gets access to the application or specific operations within it. This check is used in place of the tests in conventional UNIX applications for UID=0.

Authorization Naming Convention

An authorization has a name that is used internally and in files (for example, `solaris.admin.usermgr.pswd`), and a short description, which appears in the graphical interfaces (for example, Change Passwords). By convention, authorization names consist of the reverse order of the Internet name of the supplier, the subject area, any subareas, and the function, which are all separated by dots. An example would be `com.xyzcorp.device.access`. The exceptions to this convention are authorizations from Sun Microsystems, Inc., which use the prefix `solaris` instead of an Internet name. This convention enables administrators to apply authorizations in a hierarchical fashion by using a wildcard (*) to represent any strings to the right of a dot.

Example of Authorization Granularity

As an example of how authorizations are used, consider the following. A user in the Operator role might be limited to the `solaris.admin.usermgr.read` authorization, which provides read but not write access to users' configuration files. The System Administrator role naturally has the `solaris.admin.usermgr.read` and also `solaris.admin.usermgr.write` authorization for making changes to

users' files, but without the `solaris.admin.usermgr.pswd` authorization, the System Administrator cannot change passwords. The Primary Administrator has all three of these authorizations. The `solaris.admin.usermgr.pswd` authorization is required to make password changes in the Solaris Management Console User Tool. This authorization is also required for using the password modification options in the `smuser`, `smmultiuser`, and `smrole` commands.

Delegating Authorizations

An authorization that ends with the suffix `grant` permits a user or role to delegate to other users any assigned authorizations that begin with the same prefix. For example, a role with the authorizations `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.read` can delegate the `solaris.admin.usermgr.read` authorization to another user. A role with the `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.*` can delegate any of the authorizations with the `solaris.admin.usermgr` prefix to other users.

Databases Supporting RBAC

Overview of Databases Supporting RBAC

Data for the RBAC elements is stored in these four databases:

- `user_attr` (extended user attributes database) — Associates users and roles with authorizations and rights
- `auth_attr` (authorization attributes database) — Defines authorizations and their attributes, and identifies the associated help file
- `prof_attr` (rights profile attributes database) — Defines rights profiles, lists the rights profile's assigned authorizations, and identifies the associated help file
- `exec_attr` (execution attributes database) — Identifies the commands with security attributes that are assigned to specific rights profiles

Note – The commands can also indicate a security policy. Currently, the only security policy available for the Solaris operating environment is `suser` (short for superuser). The `suser` policy is the default and it accommodates both the ID attributes and authorizations. The Trusted Solaris environment, which can interoperate with the Solaris environment, uses a policy called `tsol`. Additional policies might be available in future releases.

The `policy.conf` database is also important to the RBAC implementation. `policy.conf` can contain authorizations and rights profiles to be applied to all users by default.

The following figure illustrates how the RBAC databases work together.

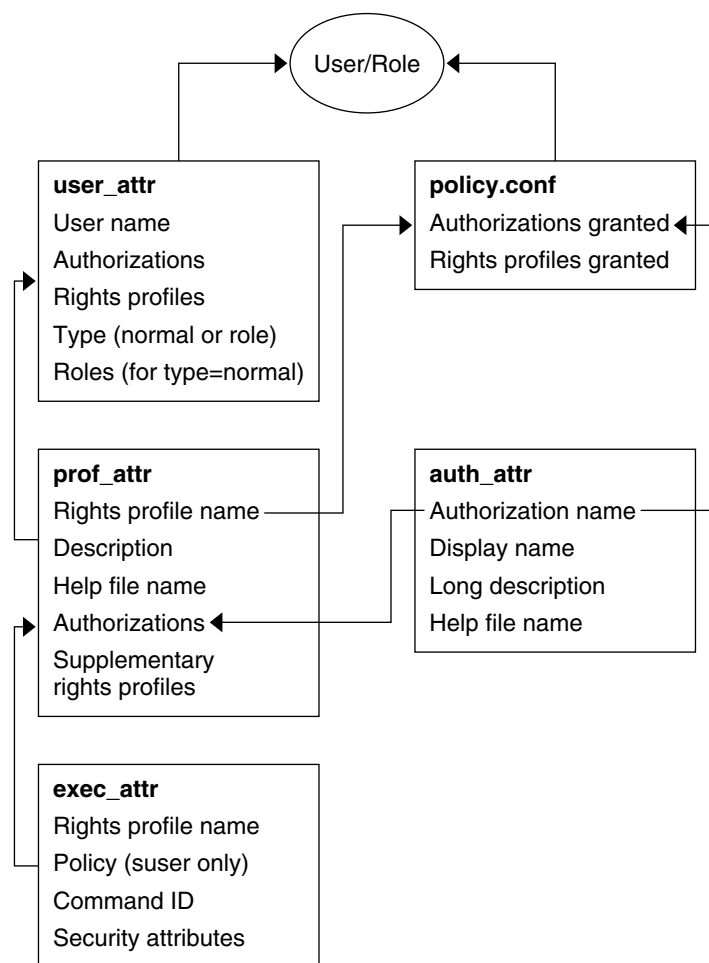


FIGURE 19-1 RBAC Database Relations

The `user_attr` database stores the basic definitions for both users and roles (they are differentiated by the `type` field). `user_attr` contains the attributes that are shown in the figure, which includes a comma-separated list of rights profile names. The definitions of the rights profiles are split between two databases. The `prof_attr` database contains rights profile identification information, authorizations that are assigned to the profile, and supplementary profiles. The `exec_attr` database identifies the policy and contains commands with their associated security attributes. The `auth_attr` database supplies authorization information to the SMC tools. The `policy.conf` database supplies default authorizations and rights profiles to be applied to all users.

Each of these databases uses a key=value syntax for storing attributes. This approach accommodates future expansion of the databases and enables a system to continue if it encounters a key that is unknown to its policy.

The scope of the RBAC databases can apply to individual hosts or to all hosts that are served by a name service such as NIS, NIS+, or LDAP. The precedence of local configuration files versus distributed databases for the `user_attr` database is set by the precedence that is specified for the `passwd` entry in the file `/etc/nsswitch.conf`. The precedence for `prof_attr` and `auth_attr` are individually set in `/etc/nsswitch.conf`. The `exec_attr` file uses the same precedence as `prof_attr`. For example, if a command with security attributes is assigned to a profile that exists in two scopes, only the entry in the first scope is used.

The databases can reside on a local system or can be administered by the NIS, NIS+, or LDAP name service.

The databases can be edited manually or manipulated by the commands that are described in “Command-Line Applications for Managing RBAC” on page 301.

user_attr Database

The `user_attr` database contains user and role information that supplements the `passwd` and `shadow` databases. `user_attr` contains extended user attributes such as authorizations, rights profiles, and assigned roles. The fields in the `user_attr` database are separated by colons, as follows:

```
user:qualifier:res1:res2:attr
```

The fields are described in the following table.

Field Name	Description
<code>user</code>	The name of the user or role as specified in the <code>passwd(4)</code> database.
<code>qualifier</code>	Reserved for future use.
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.

Field Name	Description
attr	<p>An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to be applied when the user runs commands. The four valid keys are <code>auths</code>, <code>profiles</code>, <code>roles</code>, and <code>type</code>.</p> <ul style="list-style-type: none"> ■ <code>type</code> can be set to <code>normal</code>, if this account is for a normal user, or to <code>role</code>, if this account is for a role. ■ <code>auths</code> specifies a comma-separated list of authorization names that are chosen from names that are defined in the <code>auth_attr(4)</code> database. Authorization names can include the asterisk (*) character as a wildcard. For example, <code>solaris.device.*</code> means all of the Solaris device authorizations. ■ <code>profiles</code> specifies an ordered, comma-separated list of rights profile names from the <code>prof_attr(4)</code> file. The order of rights profiles works similarly to UNIX search paths. The first rights profile in the list that contains the command to be executed defines which (if any) attributes are to be applied to the command. ■ <code>roles</code> can be assigned to the user through a comma-separated list of role names. Note that roles are defined in the same <code>user_attr</code> database. Roles are indicated by setting the <code>type</code> value to <code>role</code>. Roles cannot be assigned to other roles.

The following example demonstrates how the operator role is defined in a typical `user_attr` database and how it is assigned to user `johnDoe`. Roles and users are differentiated by the `type` keyword.

```
%grep operator /etc/user_attr
johnDoe:::type=normal;roles=sysadmin,operator
operator:::profiles=Operator;type=role
```

auth_attr Database

All authorizations are stored in the `auth_attr` database. Authorizations can be assigned directly to users (or roles) in the `user_attr` database. Authorizations can also be assigned to rights profiles, which are assigned to users.

The fields in the `auth_attr` database are separated by colons, as follows:

```
authname:res1:res2:short_desc:long_desc:attr
```

The fields are described in the following table.

Field Name	Description
authname	<p>A unique character string that is used to identify the authorization in the format <i>prefix.[suffix]</i>. Authorizations for the Solaris operating environment use <i>solaris</i> as a prefix. All other authorizations should use a prefix that begins with the reverse-order Internet domain name of the organization that creates the authorization (for example, <i>com.xyzcompany</i>). The suffix indicates what is being authorized, typically the functional area and operation.</p> <p>When the <i>authname</i> consists of a prefix and functional area and ends with a period, the <i>authname</i> serves as a heading to be used by applications in their GUIs, rather than as an actual authorization. The <i>authname</i> <i>solaris.printmgr.</i> is an example of a heading.</p> <p>When <i>authname</i> ends with the word "grant," the <i>authname</i> serves as a grant authorization and lets the user delegate authorizations with the same prefix and functional area to other users. The <i>authname</i> <i>solaris.printmgr.grant</i> is an example of a grant authorization. <i>solaris.printmgr.grant</i> gives the user the right to delegate such authorizations as <i>solaris.printmgr.admin</i> and <i>solaris.printmgr.nobanner</i> to other users.</p>
res1	Reserved for future use.
res2	Reserved for future use.
short_desc	A terse name for the authorization suitable for display in user interfaces, such as in a scrolling list in a GUI.
long_desc	A long description. This field identifies the purpose of the authorization, the applications in which it is used, and the type of user who may be interested in using it. The long description can be displayed in the help text of an application.
attr	<p>An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys can be specified.</p> <p>The keyword <i>help</i> identifies a help file in HTML. Help files can be accessed from the <i>index.html</i> file in the <i>/usr/lib/help/auths/locale/Cdirectory</i>.</p>

An *auth_attr* database with some typical values is shown in the following example.

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

Note that *solaris.admin.printer.* is defined to be a heading, because it ends in a dot (.). Headings are used by the graphical user interface to organize families of authorizations.

prof_attr Database

The name, description, help file location, and authorizations that are assigned to rights profiles are stored in the `prof_attr` database. The commands and security attributes that are assigned to rights profiles are stored in the `exec_attr` database (see “`exec_attr` Database” on page 300). The fields in the `prof_attr` database are separated by colons:

```
profname:res1:res2:desc:attr
```

The fields are described in the following table.

Field Name	Description
<code>profname</code>	The name of the profile. Profile names are case-sensitive. This name is also used by the <code>user_attr</code> file to indicate rights profiles that are assigned to roles and users.
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.
<code>desc</code>	A long description. This field should explain the purpose of the profile, including what type of user would be interested in using it. The long description should be suitable for display in the help text of an application.
<code>attr</code>	<p>An optional list of key-value pairs that are separated by semicolons (;) that describe the security attributes to apply to the object on execution. Zero or more keys can be specified. The two valid keys are <code>help</code> and <code>auths</code>.</p> <p>The keyword <code>help</code> identifies a help file in HTML. Help files can be accessed from the <code>index.html</code> file in the <code>/usr/lib/help/auths/locale/C</code> directory.</p> <p><code>auths</code> specifies a comma-separated list of authorization names that are chosen from those names that are defined in the <code>auth_attr(4)</code> database. Authorization names can be specified with the asterisk (*) character as a wildcard.</p>

A typical `prof_attr` database follows this paragraph. Notice the definition of the Printer Management rights profile. Note also that the Printer Management rights profile is a supplementary rights profile that is assigned to the Operator rights profile.

```
% grep 'Printer Management' /etc/security/prof_attr
Printer Management:::Manage printers, daemons, spooling:help=RtPrntAdmin.html; \
auths=solaris.admin.printer.read,solaris.admin.printer.modify,solaris.admin.printer.delete \
Operator:::Can perform simple administrative tasks:profiles=Printer Management,\
Media Backup,All;help=RtOperator.html
...
```

exec_attr Database

An execution attribute is a command that is associated with a set UID or GID and that is assigned to a rights profile. The command with its security attributes can be run by users or roles to whom the profile is assigned.

The definitions of the execution attributes are stored in the `exec_attr` database.

The fields in the `exec_attr` database are separated by colons:

```
name:policy:type:res1:res2:id:attr
```

The fields are described in the following table.

Field Name	Description
<code>name</code>	The name of the profile. Profile names are case-sensitive. The name refers to a rights profile in the <code>prof_attr</code> database.
<code>policy</code>	The security policy that is associated with this entry. Currently, <code>suser</code> (the superuser policy model) is the only valid policy entry.
<code>type</code>	The type of entity that is specified. Currently, the only valid type is <code>cmd</code> (command).
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.
<code>id</code>	A string that identifies the entity. Commands should have the full path or a path with a wildcard. To specify arguments, write a script with the arguments and point the <code>id</code> to the script.
<code>attr</code>	<p>An optional list of semicolon (;) separated key-value pairs that describe the security attributes to apply to the entity on execution. Zero or more keys can be specified. The list of valid key words depends on the policy that is enforced. The four valid keys are <code>eid</code>, <code>uid</code>, <code>egid</code>, and <code>gid</code>.</p> <p><code>eid</code> and <code>uid</code> contain a single user name or a numeric user ID. Commands that are designated with <code>eid</code> run with the effective UID indicated, which is similar to setting the <code>setuid</code> bit on an executable file. Commands that are designated with <code>uid</code> run with both the real and effective UIDs.</p> <p><code>egid</code> and <code>gid</code> contain a single group name or numeric group ID. Commands that are designated with <code>egid</code> run with the effective GID indicated, which is similar to setting the <code>setgid</code> bit on an executable file. Commands that are designated with <code>gid</code> run with both the real and effective GIDs.</p>

Some typical values from an `exec_attr` database are shown in the following example.

```
% grep 'Printer Management' /etc/security/exec_attr
Printer Management:suser:cmd::/usr/sbin/accept:euid=lp
Printer Management:suser:cmd::/usr/ucb/lpq:euid=0
Printer Management:suser:cmd::/etc/init.d/lp:euid=0
.
.
.
```

policy.conf File

The `policy.conf` file provides a way of granting specific rights profiles and authorizations to all users. The two types of entries in the file consist of key-value pairs. They are the following:

- `AUTHS_GRANTED=authorizations`
- `PROFS_GRANTED=right profiles`

authorizations refers to one or more authorizations and *right profiles* refers to one or more rights profiles. Some typical values from a `policy.conf` database are shown in the following example.

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw

# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User
```

RBAC Commands

Command-Line Applications for Managing RBAC

In addition to editing the databases directly, the following tools are available for managing with role-based access control.

TABLE 19-7 RBAC Administration Commands

Command	Description
<code>auths(1)</code>	Displays authorizations for a user.
<code>makedbm(1M)</code>	Makes a dbm file.

TABLE 19-7 RBAC Administration Commands *(Continued)*

Command	Description
nscd(1M)	Name service cache daemon, useful for caching the <code>user_attr</code> , <code>prof_attr</code> , and <code>exec_attr</code> databases.
pam_roles(5)	Role account management module for PAM. Checks for the authorization to assume role.
pfexec(1)	Used by profile shells to execute commands with attributes that are specified in the <code>exec_attr</code> database.
policy.conf(4)	Configuration file for security policy. Lists granted authorizations.
profiles(1)	Displays profiles for a specified user.
roles(1)	Displays roles that are granted to a user.
roleadd(1M)	Adds a role account on the system.
roledel(1M)	Deletes a role's account from the system.
rolemod(1M)	Modifies a role's account information on the system.
smattrpop(1M)	Merges source security attribute database into target database. For use in situations where local databases need to be merged into a naming service and in upgrades where conversion scripts are not supplied.
smexec(1M)	Manages entries in the <code>exec_attr</code> database. Requires authentication.
smmultiuser(1M)	Manages bulk operations on user accounts. Requires authentication.
smuser(1M)	Manages user entries. Requires authentication.
smprofile(1M)	Manages profiles in the <code>prof_attr</code> and <code>exec_attr</code> databases. Requires authentication.
smrole(1M)	Manages roles and users in role accounts. Requires authentication.
useradd(1M)	Adds a user account on the system. The <code>-P</code> option assigns a role to a user's account.
userdel(1M)	Deletes a user's login from the system.
usermod(1M)	Modifies a user's account information on the system.

Commands Requiring Authorizations

The following table provides examples of how authorizations are used to limit command options in the Solaris environment. See also “Authorizations” on page 292.

TABLE 19-8 Commands and Associated Authorizations

Commands	Authorization Requirements
at(1)	solaris.jobs.user required for all options (when neither at.allow nor at.deny files exist)
atq(1)	solaris.jobs.admin required for all options
crontab(1)	solaris.jobs.user required for the option to submit a job (when neither crontab.allow nor crontab.deny files exist) solaris.jobs.admin required for the options to list or modify other users' crontab files
allocate(1) (with BSM enabled only)	solaris.device.allocate (or other authorization as specified in device_allocate(4)) required to allocate device. solaris.device.revoke (or other authorization as specified in device_allocate file) required to allocate device to another user (-F option)
deallocate(1) (with BSM enabled only)	solaris.device.allocate (or other authorization as specified in device_allocate(4)) required to deallocate another user's device. solaris.device.revoke (or other authorization as specified in device_allocate file) required to force deallocation of the specified device (-F option) or all devices (-I option)
list_devices(1) (with BSM enabled only)	solaris.device.revoke required to list another user's devices (-U option)

Using Automated Security Enhancement Tool (Tasks)

This chapter describes how to use the Automated Security Enhancement Tool (ASET) to monitor or restrict access to system files and directories.

This is a list of step-by-step instructions in this chapter.

- “How to Run ASET Interactively” on page 324
- “How to Run ASET Periodically” on page 325
- “How to Stop Running ASET Periodically” on page 325
- “How to Collect ASET Reports on a Server” on page 326

Automated Security Enhancement Tool (ASET)

The Solaris 9 release includes the Automated Security Enhancement Tool (ASET). ASET helps you monitor and control system security by automatically performing tasks that you would otherwise do manually.

The ASET security package provides automated administration tools that enable you to control and monitor your system’s security. You specify a security level—low, medium, or high—at which ASET will run. At each higher level, ASET’s file-control functions increase to reduce file access and tighten your system security.

There are seven tasks involved with ASET, each performing specific checks and adjustments to system files. The ASET tasks tighten file permissions, check the contents of critical system files for security weaknesses, and monitor crucial areas. ASET can safeguard a network by applying the basic requirements of a firewall system to a system that serves as a gateway system. (See “Firewall Setup” on page 309.)

ASET uses master files for configuration. Master files, reports, and other ASET files are in the `/usr/aset` directory. These files can be changed to suit the particular requirements of your site.

Each task generates a report noting detected security weaknesses and changes the task has made to the system files. When run at the highest security level, ASET will attempt to modify all system security weaknesses. If it cannot correct a potential security problem, ASET reports the existence of the problem.

You can initiate an ASET session by using the `/usr/aset` command interactively, or you can also set up ASET to run periodically by putting an entry into the `crontab` file.

ASET tasks are disk-intensive and can interfere with regular activities. To minimize the impact on system performance, schedule ASET to run when system activity level is lowest, for example, once every 24 or 48 hours at midnight.

ASET Security Levels

ASET can be set to operate at one of three security levels: low, medium, or high. At each higher level, ASET's file-control functions increase to reduce file access and heighten system security. These functions range from monitoring system security without limiting users' file access, to increasingly tightening access permissions until the system is fully secured.

The three levels are outlined in the table below.

Security Level	This Level ...
Low Security	Ensures that attributes of system files are set to standard release values. ASET performs several checks and reports potential security weaknesses. At this level, ASET takes no action and does not affect system services.
Medium Security	Provides adequate security control for most environments. ASET modifies some of the settings of system files and parameters, restricting system access to reduce the risks from security attacks. ASET reports security weaknesses and any modifications it makes to restrict access. At this level, ASET does not affect system services.
High Security	Renders a highly secure system. ASET adjusts many system files and parameter settings to minimum access permissions. Most system applications and commands continue to function normally, but at this level, security considerations take precedence over other system behavior.

Note – ASET does not change the permissions of a file to make it less secure, unless you downgrade the security level or intentionally revert the system to the settings that existed prior to running ASET.

ASET Tasks

This section discusses what ASET does. You should understand each ASET task—what its objectives are, what operations it performs, and what system components it affects—to interpret and use the reports effectively.

ASET report files contain messages that describe as specifically as possible any problems discovered by each ASET task. These messages can help you diagnose and correct these problems. However, successful use of ASET assumes that you possess a general understanding of system administration and system components. If you are a new administrator, you can refer to other SunOS 5.8 system administration documentation and related manual pages to prepare yourself for ASET administration.

The `taskstat` utility identifies the tasks that have been completed and the ones that are still running. Each completed task produces a report file. For a complete description of the `taskstat` utility, refer to `taskstat(1M)`.

System Files Permissions Verification

This task sets the permissions on system files to the security level you designate. It is run when the system is installed. If you decide later to alter the previously established levels, run this task again. At low security, the permissions are set to values that are appropriate for an open information-sharing environment. At medium security, the permissions are tightened to produce adequate security for most environments. At high security, they are tightened to severely restrict access.

Any modifications that this task makes to system files permissions or parameter settings are reported in the `tune.rpt` file. “Tune Files” on page 322 shows an example of the files that ASET consults when setting permissions.

System Files Checks

This task examines system files and compares each one with a description of that file listed in a master file. The master file is created the first time ASET runs this task. The master file contains the system file settings enforced by `checklist` for the specified security level.

A list of directories whose files are to be checked is defined for each security level. You can use the default list, or you can modify it, specifying different directories for each level.

For each file, the following criteria are checked:

- Owner and group
- Permission bits
- Size and checksum
- Number of links
- Last modification time

Any discrepancies found are reported in the `cklist.rpt` file. This file contains the results of comparing system file size, permission, and checksum values to the master file.

User/Group Checks

This task checks the consistency and integrity of user accounts and groups as defined in the `passwd` and `group` files. It checks the local, and NIS or NIS+ password files. NIS+ password file problems are reported but not corrected. This task checks for the following violations:

- Duplicate names or IDs
- Entries in incorrect format
- Accounts without a password
- Invalid login directories
- The `nobody` account
- Null group password
- A plus sign (+) in the `/etc/passwd` file on an NIS (or NIS+) server

Discrepancies are reported in the `usrgrp.rpt` file.

System Configuration Files Check

During this task, ASET checks various system tables, most of which are in the `/etc` directory. These files are:

- `/etc/default/login`
- `/etc/hosts.equiv`
- `/etc/inetd.conf`
- `/etc/aliases`
- `/var/adm/utmpx`
- `/.rhosts`
- `/etc/vfstab`
- `/etc/dfs/dfstab`

- `/etc/ftpusers`

ASET performs various checks and modifications on these files, and reports all problems in the `sysconf.rpt` file.

Environment Check

This task checks how the `PATH` and `UMASK` environment variables are set for root, and other users, in the `/.profile`, `/.login`, and `/.cshrc` files.

The results of checking the environment for security are reported in the `env.rpt` file.

EEPROM Check

This task checks the value of the `EEPROM` security parameter to ensure that it is set to the appropriate security level. You can set the `EEPROM` security parameter to `none`, `command`, or `full`.

ASET does not change this setting, but reports its recommendations in the `EEPROM.rpt` file.

Firewall Setup

This task ensures that the system can be safely used as a network relay. It protects an internal network from external public networks by setting up a dedicated system as a firewall, which is described in “Firewall Systems” on page 213. The firewall system separates two networks, each of which approaches the other as untrusted. The firewall setup task disables the forwarding of Internet Protocol (IP) packets and hides routing information from the external network.

The firewall task runs at all security levels, but takes action only at the highest level. If you want to run ASET at high security, but find that your system does not require firewall protection, you can eliminate the firewall task by editing the `asetenv` file.

Any changes made are reported in the `firewall.rpt` file.

ASET Execution Log

ASET generates an execution log whether it runs interactively or in the background. By default, ASET generates the log file on standard output. The execution log confirms that ASET ran at the designated time, and also contains any execution error messages.

The `aset -n` command directs the log to be delivered by electronic mail to a designated user. For a complete list of ASET options, refer to `aset(1M)`.

Example of an ASET Execution Log File

```
ASET running at security level low

Machine=example; Current time = 0325_08:00

aset: Using /usr/aset as working directory

Executing task list...
    firewall
    env
    sysconfig
    usrgrp
    tune
    cklist
    eeprom
All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
    $/usr/aset/util/taskstat    aset_dir
Where aset_dir is ASET's operating directory, currently=/usr/aset

When the tasks complete, the reports can be found in:
    /usr/aset/reports/latest/*.rpt
You can view them by:
more /usr/aset/reports/latest/*.rpt
```

The log first shows the system and time that ASET was run. Then it lists each task as it is started.

ASET invokes a background process for each of these tasks, which are described in “ASET Tasks” on page 307. The task is listed in the execution log when it starts; this does not indicate that it has been completed. To check the status of the background tasks, use the `taskstat` utility.

ASET Reports

All report files generated from ASET tasks are in subdirectories under the `/usr/aset/reports` directory. This section describes the structure of the `/usr/aset/reports` directory, and provides guidelines on managing the report files.

ASET places the report files in subdirectories that are named to reflect the time and date when the reports are generated. This enables you to keep an orderly trail of

records documenting the system status as it varies between ASET executions. You can monitor and compare these reports to determine the soundness of your system's security.

The figure below shows an example of the `reports` directory structure.

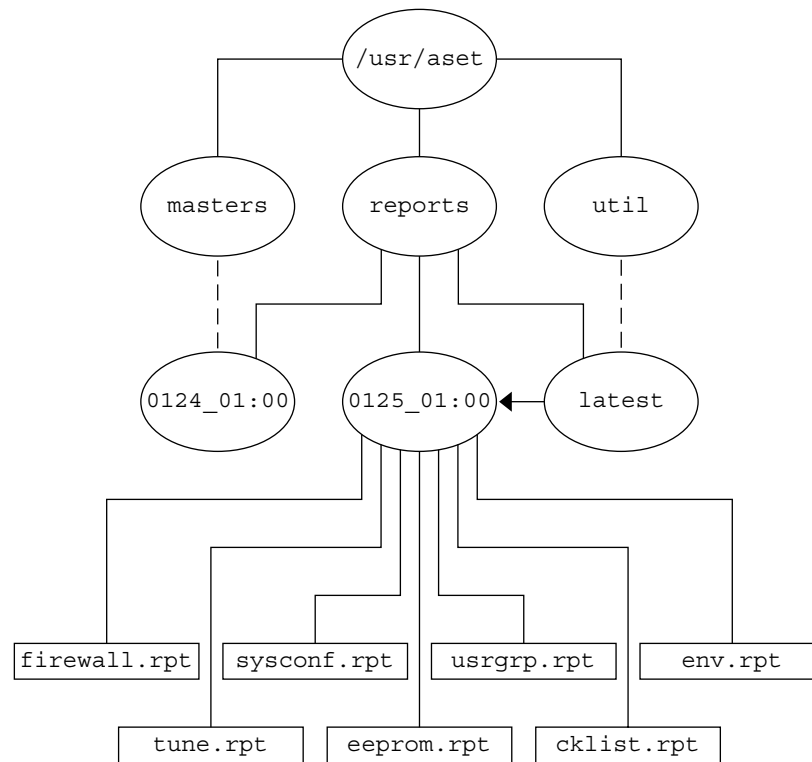


FIGURE 20-1 ASET `reports` Directory Structure

Two report subdirectories are shown in this example:

- 0124_01:00
- 0125_01:00

The subdirectory names indicate the date and time the reports were generated. Each report subdirectory name has the following format:

monthdate_hour:minute

where *month*, *date*, *hour*, and *minute* are all two-digit numbers. For example, 0125_01:00 represents January 25, at 1 a.m.

Each of the two report subdirectories contains a collection of reports generated from one execution of ASET.

The `latest` directory is a symbolic link that always points to the subdirectory that contains the latest reports. Therefore, to look at the latest reports that ASET has generated, you can go to the `/usr/aset/reports/latest` directory. There is a report file in this directory for each task that ASET performed during its most recent execution.

Format of ASET Report Files

Each report file is named after the task that generates it. See the table below for a list of tasks and their reports.

TABLE 20-1 ASET Tasks and Resulting Reports

Tasks	Report
System file permissions tuning (<code>tune</code>)	<code>tune.rpt</code>
System files checklist (<code>cklist</code>)	<code>cklist.rpt</code>
User/group checks (<code>usrgrp</code>)	<code>usrgrp.rpt</code>
System configuration files check (<code>sysconf</code>)	<code>sysconf.rpt</code>
Environment check (<code>env</code>)	<code>env.rpt</code>
EEPROM check (<code>eeeprom</code>)	<code>eeeprom.rpt</code>
Firewall setup (<code>firewall</code>)	<code>firewall.rpt</code>

Within each report file, messages are bracketed by a beginning and an ending banner line. Sometimes a task terminates prematurely; for example, when a component of ASET is accidentally removed or damaged. In most cases, the report file will contain a message near the end that indicates the reason for the premature exit.

The following is a sample report file, `usrgrp.rpt`.

```
*** Begin User and Group Checking ***

Checking /etc/passwd ...
Warning! Password file, line 10, no passwd
:sync::1:1:::/bin/sync
..end user check; starting group check ...
Checking /etc/group...
*** End User And group Checking ***
```


Examining ASET Report Files

After initially running or reconfiguring ASET, you should examine the report files closely. (Reconfiguration includes modifying the `asetenv` file or the master files in the `masters` subdirectory, or changing the security level at which ASET operates.) The reports record any errors introduced when you reconfigured. By watching the reports closely, you can react to, and solve, problems as they arise.

Comparing ASET Report Files

After you monitor the report files for a period during which there are no configuration changes or system updates, you might find that the content of the reports begin to stabilize and that it contains little, if any, unexpected information. You can use the `diff` utility to compare reports.

ASET Master Files

ASET's master files, `tune.high`, `tune.low`, `tune.med`, and `uid_aliases`, are located in the `/usr/aset/masters` directory. ASET uses the master files to define security levels.

Tune Files

The `tune.low`, `tune.med`, and `tune.high` master files define the available ASET security levels. They specify the attributes of system files at each level and are used for comparison and reference purposes.

The `uid_aliases` File

The `uid_aliases` file contains a list of multiple user accounts sharing the same ID. Normally, ASET warns about such multiple user accounts because this practice lessens accountability. You can allow for exceptions to this rule by listing the exceptions in the `uid_aliases` file. ASET does not report entries in the `passwd` file with duplicate user IDs if these entries are specified in the `uid_aliases` file.

Avoid having multiple user accounts (password entries) share the same user ID. You should consider other methods of achieving your objective. For example, if you intend for several users to share a set of permissions, you could create a group account. Sharing user IDs should be your last resort, used only when absolutely necessary and when other methods will not accomplish your objectives.

You can use the `UID_ALIASES` environment variable to specify an alternate aliases file. The default is `/usr/aset/masters/uid_aliases`.

The Checklist Files

The master files used by the systems files checklist are generated when you first execute ASET, or when you run ASET after you change the security level.

The files checked by this task are defined by the following environment variables:

- `CKLISTPATH_LOW`
- `CKLISTPATH_MED`
- `CKLISTPATH_HIGH`

ASET Environment File (`asetenv`)

The environment file, `asetenv`, contains a list of variables that affect ASET tasks. These variables can be changed to modify ASET operation.

Configuring ASET

This section discusses how ASET is configured and the environment under which it operates.

ASET requires minimum administration and configuration, and in most cases, you can run it with the default values. You can, however, fine-tune some of the parameters that affect the operation and behavior of ASET to maximize its benefit. Before changing the default values, you should understand how ASET works, and how it affects the components of your system.

ASET relies on four configuration files to control behavior of its tasks:

- `/usr/aset/asetenv`
- `/usr/aset/masters/tune.low`
- `/usr/aset/masters/tune.med`
- `/usr/aset/masters/tune.high`

Modifying the Environment File (`asetenv`)

The `/usr/aset/asetenv` file has two main sections:

- A user-configurable parameters section
- An internal environment variables section

You can alter the user-configurable parameters section. However, the settings in the internal environment variables section are for internal use only and should not be modified.

You can edit the entries in the user-configurable parameters section to:

- Choose which tasks to run
- Specify directories for checklist task
- Schedule ASET execution
- Specify an aliases file
- Extend checks to NIS+ tables

Choose Which Tasks to Run: TASKS

Each of the tasks ASET performs monitors a particular area of system security. In most system environments, all the tasks are necessary to provide balanced security coverage. However, you might decide to eliminate one or more of the tasks.

For example, the firewall task runs at all security levels, but takes action only at the high security level. You might want to run ASET at the high-security level, but do not require firewall protection.

It's possible to set up ASET to run at the high level without the firewall feature by editing the `TASKS` list of environment variables in the `asetenv` file. By default, the `TASKS` list contains all of the ASET tasks. (An example is shown below.) To delete a task, remove the task setting from the file. In this case, you would delete the `firewall` environment variable from the list. The next time ASET runs, the excluded task will not be performed.

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

Specify Directories for Checklist Task: CKLISTPATH

The system files check checks attributes of files in selected system directories. You define which directories to check by using these checklist path environment variables:

- `CKLISTPATH_LOW`
- `CKLISTPATH_MED`
- `CKLISTPATH_HIGH`

The `CKLISTPATH_LOW` variable defines the directories to be checked at the low security level. `CKLISTPATH_MED` and `CKLISTPATH_HIGH` environment variables function similarly for the medium and high security levels.

The directory list defined by a variable at a lower security level should be a subset of the directory list defined at the next higher level. For example, all directories specified

for `CKLISTPATH_LOW` should be included in `CKLISTPATH_MED`, and all the directories specified for `CKLISTPATH_MED` should be included in `CKLISTPATH_HIGH`.

Checks performed on these directories are not recursive; ASET only checks those directories explicitly listed in the variable. It does not check their subdirectories.

You can edit these variable definitions to add or delete directories that you want ASET to check. Note that these checklists are useful only for system files that do not normally change from day to day. A user's home directory, for example, is generally too dynamic to be a candidate for a checklist.

Schedule ASET Execution: `PERIODIC_SCHEDULE`

When you start ASET, you can start it interactively, or use the `-p` option to request that the ASET tasks run at a scheduled time and period. You can run ASET periodically, at a time when system demand is light. For example, ASET consults `PERIODIC_SCHEDULE` to determine how frequently to execute the ASET tasks, and at what time to run them. For detailed instructions about setting up ASET to run periodically, see "How to Run ASET Periodically" on page 325.

The format of `PERIODIC_SCHEDULE` follows the format of `crontab` entries. See `crontab(1)` for complete information.

Specify an Aliases File: `UID_ALIASES`

The `UID_ALIASES` variable specifies an aliases file that lists shared user IDs. The default is `/usr/aset/masters/uid_aliases`.

Extend Checks to NIS+ Tables: `YPCHECK`

The `YPCHECK` environment variable specifies whether ASET should also check system configuration file tables. `YPCHECK` is a Boolean variable; you can specify only true or false for it. The default value is false, disabling NIS+ table checking.

To understand how this variable works, consider its effect on the `passwd` file. When this variable is set to false, ASET checks the local `passwd` file. When it is set to true, the task also checks the NIS+ `passwd` file for the domain of the system.

Note – Although ASET automatically repairs the local tables, it only reports potential problems in the NIS+ tables; it does not change them.

Modifying the Tune Files

ASET uses the three master tune files, `tune.low`, `tune.med`, and `tune.high`, are used by ASET to ease or tighten access to critical system files. These master files are located in the `/usr/aset/masters` directory, and they can be modified to suit your environment. For additional information, see “Tune Files” on page 322.

The `tune.low` file sets permissions to values appropriate for default system settings. The `tune.med` file further restricts these permissions and includes entries not present in `tune.low`. The `tune.high` file restricts permissions even further.

Note – Modify settings in the tune file by adding or deleting file entries. Setting a permission to a less restrictive value than the current setting has no effect; the ASET tasks do not relax permissions unless you downgrade your system security to a lower level.

Restoring System Files Modified by ASET

When ASET is executed for the first time, it saves and archives the original system files. The `aset.restore` utility reinstates these files. It also deschedules ASET, if it is currently scheduled for periodic execution. The `aset.restore` utility is located in `/usr/aset`, the ASET operating directory.

Changes made to system files are lost when you run `aset.restore`.

You should use `aset.restore`:

- When you want to remove ASET changes and restore the original system. If you want to deactivate ASET permanently, you can remove it from `cron` scheduling if the `aset` command had been added to root’s `crontab` previously. For directions on how to use `cron` to remove automatic execution, see “How to Stop Running ASET Periodically” on page 325.
- After a brief period of experimenting with ASET, to restore the original system state.
- When some major system functionality is not working properly and you suspect that ASET is causing the problem.

Network Operation Using the NFS System

Generally, ASET is used in standalone mode, even on a system that is part of a network. As system administrator for your standalone system, you are responsible for the security of your system and for running and managing ASET to protect your system.

You can also use ASET in the NFS distributed environment. As a network administrator, you are responsible for installing, running, and managing various administrative tasks for all of your clients. To facilitate ASET management across several client systems, you can make configuration changes that are applied globally to all clients, eliminating the need for you to log in to each system to repeat the process.

When deciding how to set up ASET on your networked systems, you should consider how much you want users to control security on their own systems, and how much you want to centralize responsibility for security control.

Providing a Global Configuration for Each Security Level

A case might arise where you want to set up more than one network configuration. For example, you might want to set up one configuration for clients that are designated with low security level, another configuration for those with medium level, and yet another one with high level.

If you need to create a separate ASET network configuration for each security level, you can create three ASET configurations on the server—one for each level. You would export each configuration to the clients with the appropriate security level. Some ASET components that are common to all three configurations could be shared using links.

Collecting ASET Reports

Not only can you centralize the ASET components on a server to be accessed by clients with or without superuser privilege, but you can also set up a central directory on a server to collect all reports produced by tasks running on various clients. For instructions on setting up a collection mechanism, see “How to Collect ASET Reports on a Server” on page 326.

Setting up the collection of reports on a server allows you to review reports for all clients from one location. You can use this method whether a client has superuser privilege or not. Alternatively, you can leave the reports directory on the local system when you want users to monitor their own ASET reports.

ASET Environment Variables

The table below lists the ASET environment variables and the values that they specify.

TABLE 20-2 ASET Environment Variables and Their Meanings

Environment Variable	Specifies ...
ASETDIR (See below)	ASET working directory
ASETSECLEVEL (See below)	Security level
PERIODIC_SCHEDULE	Periodic schedule
TASKS	Tasks to run
UID_ALIASES	Aliases file
YPCHECK	Extends check to NIS and NIS+
CKLISTPATH_LOW	Directory lists for low security
CKLISTPATH_MED	Directory list for medium security
CKLISTPATH_HIGH	Directory list for high security

The environment variables listed below are found in the `/usr/aset/asetenv` file. The `ASETDIR` and `ASETSECLEVEL` variables are optional and can be set only through the shell by using the `aset` command. The other environment variables can be set by editing the file. The variables are described below.

ASETDIR Variable

`ASETDIR` specifies an ASET working directory.

From the C shell, type:

```
% setenv ASETDIR pathname
```

From the Bourne shell or the Korn shell, type:

```
$ ASETDIR=pathname  
$ export ASETDIR
```

Set *pathname* to the full path name of the ASET working directory.

ASETSECLEVEL Variable

The `ASETSECLEVEL` variable specifies a security level at which ASET tasks are executed.

From the C shell, type:

```
% setenv ASETSECLEVEL level
```

From the Bourne shell or the Korn shell, type:

```
$ ASETSECLEVEL=level
export ASETSECLEVEL
```

In the above commands, *level* can be set to one of the following:

low	Low security level
med	Medium security level
high	High security level

PERIODIC_SCHEDULE Variable

The value of `PERIODIC_SCHEDULE` follows the same format as the `crontab` file. Specify the variable value as a string of five fields enclosed in double quotation marks, each field separated by a space:

```
"minutes hours day-of-month month day-of-week"
```

TABLE 20-3 Periodic_Schedule Variable Values

Variable	Value
<i>minutes hours</i>	Specifies start time in number of minutes (0-59) after the hour and the hour (0-23)
<i>day-of-month</i>	Specifies the day of the month when ASET should be run, using values from 1 through 31
<i>month</i>	Specifies the month of the year when ASET should be run, using values from 1 through 12
<i>day-of-week</i>	Specifies the day of the week when ASET should be run, using values from 0 through 6; Sunday is day 0 in this scheme

The following rules apply:

- You can specify a list of values, each delimited by a comma, for any field.
- You can specify a value as a number, or you can specify it as a range; that is, a pair of numbers joined by a hyphen. A range states that the ASET tasks should be executed for every time included in the range.
- You can specify an asterisk (*) as the value of any field. An asterisk specifies all possible values of the field, inclusive.

The default entry for `PERIODIC_SCHEDULE` variable causes ASET to execute at 12:00 midnight every day:

```
PERIODIC_SCHEDULE="0 0 * * *"
```

TASKS Variable

The `TASKS` variable lists the tasks that ASET performs. The default is to list all seven tasks:

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

UID_ALIASES Variable

The `UID_ALIASES` variable specifies an aliases file. If present, ASET consults this file for a list of permitted multiple aliases. The format is `UID_ALIASES=pathname`. *pathname* is the full path name of the aliases file.

The default is:

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

YPCHECK Variable

The `YPCHECK` variable extends the task of checking system tables to include NIS or NIS+ tables. It is a Boolean variable, which can be set to either true or false.

The default is false, confining checking to local system tables:

```
YPCHECK=false
```

CKLISTPATH_level Variable

The three checklist path variables list the directories to be checked by the checklist task. The following definitions of the variables are set by default; they illustrate the relationship between the variables at different levels:

```
CKLISTPATH_LOW=${ASETDIR}/tasks:${ASETDIR}/util:${ASETDIR}/masters:  
/etc  
CKLISTPATH_MED=${CKLISTPATH_LOW}:/usr/bin:/usr/ucb  
CKLISTPATH_HIGH=${CKLISTPATH_MED}:/usr/lib:/sbin:/usr/sbin:/usr/ucb/lib
```

The values for the checklist path environment variables are similar to those of the shell path variables, in that they are lists of directory names separated by colons (:). You use an equal sign (=) to connect the variable name to its value.

ASET File Examples

This section has examples of some of the ASET files, including the tune files and the aliases file.

Tune Files

ASET maintains three tune files. The entry format in all three tune files are described in the table below.

TABLE 20-4 Tune File Entry Format

Entry	Description
<i>pathname</i>	The full path name of the file
<i>mode</i>	A five-digit number that represents the permission setting
<i>owner</i>	The owner of the file
<i>group</i>	The group owner of the file
<i>type</i>	The type of the file

The following rules apply:

- You can use regular shell wildcard characters, such as an asterisk (*) and a question mark (?), in the path name for multiple references. See `sh(1)` for more information.
- *mode* represents the least restrictive value. If the current setting is already more restrictive than the specified value, ASET does not loosen the permission settings. For example, if the specified value is 00777, the permission will remain unchanged, because 00777 is always less restrictive than whatever the current setting is.
This is how ASET handles mode setting, unless the security level is being downgraded or you are removing ASET. When you decrease the security level from what it was for the previous execution, or when you want to restore the system files to the state they were in before ASET was first executed, ASET recognizes what you are doing and decreases the protection level.
- You must use names for *owner* and *group* instead of numeric IDs.
- You can use a question mark (?) in place of *owner*, *group*, and *type* to prevent ASET from changing the existing values of these parameters.
- *type* can be `symlink` (symbolic link), `directory`, or `file` (everything else).
- Higher security level tune files reset file permissions to be at least as restrictive as they are at lower levels. Also, at higher levels, additional files are added to the list.

- A file can match more than one tune file entry. For example, `etc/passwd` matches `etc/pass*` and `/etc/*` entries.
- Where two entries have different permissions, the file permission is set to the most restrictive value. In the following example, the permission of `/etc/passwd` will be set to `00755`, which is the more restrictive of `00755` and `00770`.

```
/etc/pass*      00755      ?   ?   file
/etc/*          00770      ? ?  file
```

- If two entries have different *owner* or *group* designations, the last entry takes precedence. The following example shows the first few lines of the `tune.lpw` file.

```
/ 02755 root root directory
/bin 00777 root bin symlink
/sbin 02775 root sys directory
/usr/sbin 02775 root bin directory
/etc 02755 root sys directory
/etc/chroot 00777 bin bin symlink
```

Aliases File

An aliases file contains a list of aliases that share the same user ID.

Each entry is in this form:

```
uid=alias1=alias2=alias3= . . .
```

<i>uid</i>	Shared user ID.
<i>aliasn</i>	User account sharing the user ID.

For example, the following entry lists the user ID `0` being shared by `sysadm` and `root`:

```
0=root=sysadm
```

Running ASET

This section describes how to run ASET either interactively or periodically.

▼ How to Run ASET Interactively

1. **Become superuser.**
2. **Run ASET interactively by using the `aset` command.**

```
# /usr/aset/aset -l level -d pathname
```

<i>level</i>	Specifies the level of security. Valid values are low, medium, or high. The default setting is low. See “ASET Security Levels” on page 306 for detailed information about security levels.
<i>pathname</i>	Specifies the working directory for ASET. The default is /usr/aset.

3. **Verify ASET is running by viewing the ASET execution log that is displayed on the screen.**

The execution log message identifies which tasks are being run.

Example—Running ASET Interactively

The following example runs ASET at low security with the default working directory.

```
# /usr/aset/aset -l low
===== ASET Execution Log =====
```

```
ASET running at security level low
```

```
Machine = jupiter; Current time = 0111_09:26
```

```
aset: Using /usr/aset as working directory
```

```
Executing task list ...
```

```
  firewall
  env
  sysconf
  usrgrp
  tune
  cklist
  eeprom
```

All tasks executed. Some background tasks may still be running.

```
Run /usr/aset/util/taskstat to check their status:
/usr/aset/util/taskstat [aset_dir]
```

where `aset_dir` is ASET's operating directory, currently=/usr/aset.

When the tasks complete, the reports can be found in:
`/usr/aset/reports/latest/*.rpt`

You can view them by:
`more /usr/aset/reports/latest/*.rpt`

▼ How to Run ASET Periodically

1. Become superuser.

2. If necessary, set up the time when you want ASET to run periodically.

You should have ASET run when system demand is light. The `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file is used to set up the time for ASET to run periodically. By default, the time is set for midnight every 24 hours.

If you want to set up a different time, edit the `PERIODIC_SCHEDULE` variable in the `/usr/aset/asetenv` file. See “`PERIODIC_SCHEDULE` Variable” on page 320 for detailed information about setting the `PERIODIC_SCHEDULE` variable.

3. Add an entry to the crontab file using the `aset` command.

```
# /usr/aset/aset -p
```

`-p`

Inserts a line in the crontab file that starts ASET running at the time determined by the `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file.

4. Display the crontab entry to verify when ASET will run.

```
# crontab -l root
```

▼ How to Stop Running ASET Periodically

1. Become superuser.

2. Edit the crontab file.

```
# crontab -e root
```

3. Delete the ASET entry.

4. Save the changes and exit.

5. Display the crontab entry to verify the ASET entry is deleted.

```
# crontab -l root
```

▼ How to Collect ASET Reports on a Server

1. Become superuser.

2. Set up a directory on the server:

a. Change to the `/usr/aset` directory.

```
mars# cd /usr/aset
```

b. Create a `rptdir` directory.

```
mars# mkdir rptdir
```

c. Change to the `rptdir` directory and create a `client_rpt` directory.

```
mars# cd rptdir
mars# mkdir client_rpt
```

d. This creates a subdirectory (`client_rpt`) for a client. Repeat this step for each client whose reports you need to collect.

The following example creates the directory `all_reports`, and the subdirectories `pluto_rpt` and `neptune_rpt`.

```
mars# cd /usr/aset
mars# mkdir all_reports
mars# cd all_reports
mars# mkdir pluto_rpt
mars# mkdir neptune_rpt
```

3. Add the `client_rpt` directories to the `/etc/dfs/dfstab` file.

The directories should have read/write options.

For example, the following entries in `dfstab` are shared with read/write permissions.

```
share -F nfs -o rw=pluto /usr/aset/all_reports/pluto_rpt
share -F nfs -o rw=neptune /usr/aset/all_reports/neptune_rpt
```

4. Make the resources in the `dfstab` file available to the clients.

```
# shareall
```

5. On each client, mount the client subdirectory from the server at the mount point, `/usr/aset/masters/reports`.

```
# mount server:/usr/aset/client_rpt /usr/aset/masters/reports
```

6. Edit the `/etc/vfstab` file to mount the directory automatically at boot time.

The following sample entry in `/etc/vfstab` on `neptune` lists the directory to be mounted from `mars`, `/usr/aset/all_reports/neptune_rpt`, and the mount point on `neptune`, `/usr/aset/reports`. At boot time, the directories listed in `vfstab` are automatically mounted.

```
mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes  
hard
```

Troubleshooting ASET Problems

This section documents the error messages generated by ASET.

ASET Error Messages

ASET failed: no mail program found.

Cause: ASET is directed to send the execution log to a user, but no mail program can be found.

Solution: Install a mail program.

Usage: `aset [-n user[@host]] in /bin/mail or /usr/ucb/mail.`

Cannot decide current and previous security levels.

Cause: ASET cannot determine what the security levels are for the current and previous invocations.

Solution: Ensure the current security level is set either through the command line option or the `ASETSECLEVEL` environment variable. Also, ensure that the last line of `ASETDIR/archives/asetseclevel.arch` correctly reflects the previous security level. If these values are not set or are incorrect, specify them correctly.

ASET working directory undefined.

To specify, set `ASETDIR` environment variable or use command line option `-d`.

ASET startup unsuccessful.

Cause: The ASET working (operating) directory is not defined, or defined incorrectly.

Solution: Use the `ASETDIR` environment variable or the `-d` command line option to specify it correctly, and restart ASET.

ASET working directory \$ASETDIR missing.

ASET startup unsuccessful.

Cause: The ASET working (operating) directory is not defined, or it is defined incorrectly. This might be because the ASETDIR variable or the -d command line option refers to a nonexistent directory.

Solution: Ensure that the correct directory—that is, the directory containing the ASET directory hierarchy—is referred to correctly.

Cannot expand \$ASETDIR to full pathname.

Cause: ASET cannot expand the directory name given by the ASETDIR variable or the -d command line option to a full path name.

Solution: Ensure that the directory name is given correctly, and that it refers to an existing directory to which the user has access.

aset: invalid/undefined security level.

To specify, set ASETSECLEVEL environment variable or use command line option -l, with argument= low/med/high.

Cause: The security level is not defined or it is defined incorrectly. Only the values low, med, or high are acceptable.

Solution: Use the ASETSECLEVEL variable or the -l command line option to specify one of the three values.

ASET environment file asetenv not found in \$ASETDIR.

ASET startup unsuccessful.

Cause: ASET cannot locate an asetenv file in its working directory.

Solution: Ensure there is an asetenv file in ASET's working directory. See asetenv(4) for the details about this file.

filename doesn't exist or is not readable.

Cause: The file referred to by *filename* doesn't exist or is not readable. This can specifically occur when using the -u option where you can specify a file that contains a list of users whom you want to check.

Solution: Ensure the argument to the -u option exists and is readable.

ASET task list TASKLIST undefined.

Cause: The ASET task list, which should be defined in the asetenv file, is not defined. This can mean that your asetenv file is bad.

Solution: Examine your asetenv file. Ensure the task list is defined in the User Configurable section. Also check other parts of the file to ensure the file is intact. See asetenv(4) for the content of a good asetenv file.

ASET task list \$TASKLIST missing.

ASET startup unsuccessful.

Cause: The ASET task list, which should be defined in the `asetenv` file, is not defined. This can mean that your `asetenv` file is bad.

Solution: Examine your `asetenv` file. Ensure the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure the file is intact. See `asetenv(4)` for the content of a good `asetenv` file.

Schedule undefined for periodic invocation.

No tasks executed or scheduled. Check `asetenv` file.

Cause: ASET scheduling is requested using the `-p` option, but the variable `PERIODIC_SCHEDULE` is undefined in the `asetenv` file.

Solution: Check the `User Configurable` section of the `asetenv` file to ensure the variable is defined and is in proper format.

Warning! Duplicate ASET execution scheduled.

Check `crontab` file.

Cause: ASET is scheduled more than once. In other words, scheduling is requested while a schedule is already in effect. This is not necessarily an error if more than one schedule is indeed desired, just a warning that normally this is unnecessary since you should use the `crontab(1)` scheduling format if you want more than one schedule.

Solution: Verify, through the `command`, that the correct schedule is in effect. Ensure that no unnecessary `crontab` entries for ASET are in place.

Auditing Topics

Chapter 22	Provides overview information about auditing.
Chapter 23	Provides information to help with auditing planning issues.
Chapter 24	Provides step-by-step instructions to configure and manage auditing.
Chapter 25	Provides information about the files associated with auditing and the structure of the audit tokens.

Auditing Overview

Starting with the Solaris 2.3 release, the Basic Security Module (BSM) has been included in the full release and is part of the release media. You do not need to install BSM separately because it is included in the installation process. You can disable and enable BSM separately, however. All of the BSM software is included in the initial system installation.

This chapter introduces the concepts behind auditing and explains generally how auditing works, and contains the following information:

- “What Is Auditing?” on page 333
- “How Does Auditing Work?” on page 337
- “How is Auditing Related to Security?” on page 338
- “How Can I Configure Auditing?” on page 338
- “Why is `/etc/security` Important?” on page 340
- “Audit Utilities” on page 341

What Is Auditing?

Successful auditing depends on two other security features: identification and authentication. At login, after a user supplies a user name and password, a unique audit ID is associated with the user’s process. The audit ID is inherited by every process started during the login session. Even if a user changes identity (see the `su(1M)` man page), all actions performed are tracked with the same audit ID.

Auditing makes it possible to:

- Monitor security-relevant events that take place on the host
- Record the events in a network-wide audit trail
- Detect misuse or unauthorized activity (by analyzing the audit trail)

- Review patterns of access, and see the access histories of individuals and objects
- Discover attempts to bypass the protection mechanisms
- Discover extended use of privilege that occurs when a user changes identity
- Supply additional assurance that attempts to bypass protection mechanisms are recorded and discovered

During system configuration, the system administrator selects which activities to monitor. The administrator can also fine-tune the degree of auditing that is done for individual users.

After audit data is collected, audit-reduction and interpretation tools allow the examination of interesting parts of the audit trail. For example, you can choose to look at audit records for individual users or groups, look at all records for a certain type of event on a specific day, or select records that were generated at a certain time of day.

Auditing Terminology

The following terms are used to describe the auditing service. Some of the definitions of these terms include pointers to more complete descriptions of the term

TABLE 22-1 Terms Used in Auditing

Term	Meaning
Audit class	A grouping of audit events. Audit classes provide a way to manage a group of events. See "Audit Classes" on page 336 for more information.
Audit client	A host that is configured to run auditing.
Audit directory	A repository of audit files. See "Audit Directory" on page 337 for a description of the types of audit directories.
Audit event	A security related action that is audited. See "Audit Events" on page 335 for discussion of the types of audit events.
Audit flag	A variable used to determine which classes of events to audit and when to audit them.

TABLE 22-1 Terms Used in Auditing (Continued)

Term	Meaning
Audit policy	A set of auditing options the system administrator can enable or disable for a particular configuration. These options include whether to record or not record certain kinds of audit data or to suspend auditable actions when the queue is full.
Audit record	The binary data that describes a single audit event. An audit record is composed of audit tokens.
Audit server	A host that is set up to store audit records for selected clients.
Audit token	A set of binary data that describes a single set of event information about a process, a path or other objects. See “Audit Token Structure” on page 393 for a description of all of the audit tokens.
Audit trail	A collection of one or more audit files which may reside in separate audit file partitions.
Device allocation	The process of deciding where audit files will be stored and in what order.

Audit Events

Security-relevant actions can be audited. The system actions that are auditable are defined as *audit events*. Audit events are defined in the `/etc/security/audit_event` file. Each auditable event is defined in the file by a symbolic name, an event number, a set of preselection classes, and a short description (see the `audit_event(4)` man page).

There are several categories of audit events. The primary distinction is between events that are generated by the kernel, kernel-level events, and events that are generated by applications, called user-level events. Whether the event is generated by the kernel or by a user-level application determines the number range of the event number that the event is identified by.

TABLE 22-2 Audit Event Categories

Number Range	Type of Event
1–2047	Kernel-level audit events
2048–65535	User-level audit events

TABLE 22-2 Audit Event Categories (Continued)

Number Range	Type of Event
2048–32767	Reserved for SunOS user-level programs
32768–65536	Available for third-party applications

Most events are attributable to an individual user, but not all. These events are known as nonattributable events, Events are nonattributable if they occur at the kernel-interrupt level or before a user is identified and authenticated. Nonattributable events are auditable as well.

Kernel-Level Audit Events

Events generated by the kernel (system calls) have event numbers between 1 and 2047. The event names for kernel events begin with `AUE_`, followed by an uppercase mnemonic for the event. For example, the event number for the `creat ()` system call is 4 and the event name is `AUE_CREAT`.

User-Level Audit Events

Events generated by application software outside the kernel range from 2048 to 65535. The event names begin with `AUE_`, followed by a lowercase mnemonic for the event. Check the file, `/etc/security/audit_event`, for exact numbers of individual events. Table 22-2 shows general categories of user-related events.

Nonattributable Audit Events

Events that are not attributable to a user, such as `AUE_ENTERPROM`.

Audit Classes

Each audit event is also defined as belonging to an *audit class* or classes. By assigning events to classes, an administrator can more easily deal with large numbers of events. When naming a class, you simultaneously addresses all of the events in that class. The mapping of audit events to classes is configurable and the classes themselves are configurable. These configuration changes can be made in the `audit_event` file.

Whether or not an auditable event is recorded in the audit trail depends on whether the administrator preselects a class for auditing that includes the specific event. Out of 32 possible audit classes, 18 are defined. The 18 classes include the two global classes: `all` and `no`.

Audit Directory

An audit directory holds a collection of audit files. Many audit directories are used in a typical installation. There are several types of audit directories.

- A primary audit directory is the directory where the audit files for a system are placed under normal conditions.
- A secondary audit directory is where the audit files for a system are placed if the primary directory is full or not available.
- A directory of last resort is a local audit directory which is used if both the primary and all secondary audit directories are not available.

How Does Auditing Work?

Auditing is the generation of audit records due to the occurrence of specified events. Most commonly, events that generate audit records include the following:

- System startup and shutdown
- Login and logout
- Process or thread creation or destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of privilege, identification, and authentication actions
- Discretionary Access Control (DAC) changes by process or user
- Installation-specific administrative actions

Events that are relevant to security and which cause audit records to be generated are called *auditable events*. These events are grouped into *audit classes*, which are administrator-defined. Machine and user audit *preselection mechanisms* determine whether an actual audit record will be generated when an auditable event occurs.

Audit records come from three sources:

- They can be generated by an application.
- They can be the result of an asynchronous event.
- They can be generated as the result of a process system call.

Once the relevant information has been selected, it is formatted into an *audit record*, which is then placed in a kernel buffer known as the *audit queue*. From this temporary location, audit records are concatenated in *audit files*. Just where the audit files are located is determined by pointers in the `audit_control` file, and can include multiple partitions on the same machine, partitions on different machines, or even machines on different but linked networks. The collection of audit files that are linked together by pointers is considered an *audit trail*.

Audit records accumulate in audit files chronologically. Contained in each audit record is information identifying the event, what caused it, time of the event, and other relevant information.

How is Auditing Related to Security?

Securing a computer system, especially one deployed on a network, involves mechanisms that control activities before system or user processes begin, that monitor activities as they occur, and that report activities after they have happened. While setting up auditing requires that parameters be set before users log in or machine processes begin, most auditing activities involve monitoring current events and reporting those that meet the specified parameters. Just how auditing does this monitoring and reporting is discussed in detail in Chapter 23 and Chapter 24.

Auditing cannot prevent hackers from unauthorized entry. However auditing can report, for example, that a specific user performed specific actions at a specific time and date, and can identify the user by entry path and user name. Information such as this can be reported immediately to a system administrator's terminal and to a stored file for later analysis.

Thus auditing provides data that helps system administrators determine both how system security was compromised and what loopholes need to be changed to ensure the desired level of security.

How Can I Configure Auditing?

There are a large number of parameters that can be enabled or disabled, or set to specific limits. Actions outside these limits trigger audit records if you have set up auditing to do so. This section presents a general discussion of how you set these parameters and which sections of other chapters contain more detailed information.

Audit Events, Classes, and Policies

Some actions, such as logging in, using FTP, executing a program remotely, setting or changing a password, or becoming root automatically become auditable events and thus trigger audit records. Many other actions, though, will only trigger audit records if the relevant parameters have been set.

The set of default auditable events that is loaded when a system boots is defined in a table in `/etc/security/audit_event`. Each of these events can also be assigned to one or more classes by the system administrator, to make it easier to maintain different configurations for different users or groups of users. For more information about planning how to set up classes, see Chapter 23. For information about how to set up audit events for different users, see Chapter 24.

Audit policies are another way to specify what is audited. Generally they control how much audit information is collected and what to do if an audit file is full. Audit policies are discussed in Chapter 23 and specific procedures for changing audit policies are discussed in Chapter 24.

Audit Flags

Audit flags are options that determine whether auditing is turned on or off for specific classes if a system or application action failed or succeeded. Generally audit flags make it easier to refine auditing so that only useful information is collected. Using audit flags is discussed in more detail in Chapter 24.

Audit File Storage Issues

Beginning system administrators tend to start out wanting to collect as much information as possible about user and system actions. However when it becomes obvious that they also must have file space to store this information, and that audit files can quickly grow to fill any available space, it makes sense to be more selective about what kinds of activities are audited.

One audit directory, called the *directory of last resort*, is the only one that must be stored on the machine being audited. The primary audit directory and any secondary directories may be anywhere on the network of which the machine being audited is a part. The primary audit directory is the file location where audit files are first written, until that directory reaches a percentage of capacity that the system administrator defines. Then audit files are written to any secondary directories, in the order specified. When those directories fill to the designated capacity, audit files are written to the directory of last resort and warnings are sent to the administrator's terminal.

For more information about how to plan for use of disk space for audit files, see Chapter 23. For specific procedures to manage audit directories, see Chapter 24.

Stored Audit Files vs. Printed Audit Files

Audit files are stored in binary format to minimize use of disk space. To make it easier for people to analyze the audit files, BSM includes several utility programs that let you sort the audit records by machine, pathname, completion status, mode, date and time, audit class, user, group, and other parameters. See Chapter 24 and the `auditreduce(1M)` man page. Once sorted the way you want, you can then use `praudit` to translate the binary data into a format more easily read. This data can then be displayed on a terminal or printed.

Why is `/etc/security` Important?

The directory `/etc/security` is the default location where the system first looks for security configuration data when it is booted. Because the system looks here first, a number of the audit configuration files are stored here. These include the following files:

- `audit_class`, which defines the audit classes for this system,
- `audit_control`, which defines the location of the audit directories, the audit flags that apply to events for all users of this system, the minimum free-space percentage level for all audit file systems, and the audit flags that specify which audit classes to use when an action can not be attributed to a specific user.
- `audit_event`, which defines the default event-to-class mappings,
- `audit_startup`, which is a shell script used to start the audit daemon,
- `audit_user`, which defines those flags that differ from the default flags for specific users,
- `audit_warn`, a script which contains commands that tell the system what to do whenever the audit daemon encounters an unusual situation when writing audit records, and
- `policy_conf`, which contains the default audit policies.

The `/etc/security` directory also contains the `audit` directory, which in turn contains the per-machine `audit_data` file, which must be available for successful startup of the audit daemon at boot time.

Audit Utilities

The audit utility `praudit` and its options lets you look at the contents of any audit file. You can also use the utility `auditreduce`, either alone or in combination with `praudit`, to sort and extract audit records based on characteristics such as user name, date and time, and machine.

The `auditconfig` utility lets you configure the auditing subsystem and modify audit parameters to meet individual site requirements. Audit statistics, useful in tuning the audit subsystem, may be displayed with the `auditstat` command.

Audit Planning

This chapter describes how to set up auditing for your Solaris 8 installation. In particular, it covers things you need to consider before you install your Solaris 8 software, which automatically implements the auditing features.

This chapter covers the following:

- “How Does Auditing Work?” on page 343
- “Deciding What to Audit” on page 345
- “Determining Where to Store Audit Files” on page 356
- “Device Allocation” on page 360
- “Setting Up an Archive Policy” on page 363
- “Warnings” on page 364
- “When Should I Change Audit Parameters?” on page 366
- “Using the Audit Data” on page 367
- “Using the Results from Audit Trail Analysis” on page 370

How Does Auditing Work?

Auditing is controlled by the following configuration files:

- The `/etc/system` file
- The `audit_class`, `audit_control`, `audit_event`, and `audit_user` files and the `audit_warn` shell script in the `/etc/security` directory
- The `audit_startup` shell script

Auditing generates audit log files. While the audit configuration files are located in `/etc/security` on each machine that is audited, location of the audit log files is controlled by the contents of the `audit_control` file. The `audit_control` file lists, in order, the path names to the various audit directories, the types of audit flags in

effect, the amount of space in each directory that can be filled before a warning is issued, and which nonattributable events are selected.

With the Basic Security Module (BSM), there are three types of directories containing audit logs: the primary audit directory, optionally one or more secondary audit directories, and the audit directory of last resort. The primary audit directory is where audit log files are written when the system comes up after booting up. Secondary directories are where audit log files are written when the primary audit directory reaches the limits you specify in the `audit_control` file. The audit directory of last resort is where audit log files are stored if the network goes down. (You must have an audit partition defined on the local machine or one of two things will happen: either all jobs will stop running or all audit records will be discarded, depending on how you configured auditing.)

The directory of last resort must be located on the machine being audited, but the primary and any secondary directories can be located anywhere on the network, including other partitions on the same machine. However pointers to them must be included in the `audit_control` file.

What is an Audit Trail?

The audit trail is a generic term describing where the audit records are stored, and consisting of a number of audit files. Each audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the machine that generated them. Each audit record contains a number of audit tokens, in a specific order, that indicates what happened and which user was involved.

Audit File Names

The format of the name of a completed audit file is shown below::

start-time , finish-time . machine

Here is an example of a closed audit filename:

`20000515000001 , 20000515225243 . grumpy`

In this example, the file was started at one second after midnight GMT on May 15, 2000, was terminated the same day at 10:52:43 PM, and reflects data collected from a machine named `grumpy`.

Audit files can be open or closed. Open audit files are the result of either of these situations:

1. `auditd` is still adding data to the file, or

2. a system failure resulted in the file being left open.

An audit file is considered closed if there is a `finish-time()` in the name. They are considered open if there is no `finish-time()` or if the value `not_terminated` is found in the file name.

When a machine is writing to a remotely mounted audit file and the file server crashes or becomes inaccessible, the `not_terminated` ending time stamp remains in the current file's name and the audit daemon opens a new audit file.

Deciding What to Audit

Most system administrators find it efficient to set up system defaults for auditing by setting flags in the `audit_control` file. These flags are discussed in “Deciding What to Audit” on page 345 in this chapter and in the `audit_control(4)` man page. These defaults apply to all users and are modified by values set in the `audit_user` file for individual users. For additional information, see “The `audit_user` File” on page 386 and the `audit_user(4)` man page.

Some audit characteristics are set up automatically. The following applications begin auditable sessions and automatically initialize the user's audit characteristics:

- `in.ftpd`
- `login`
- `dtlogin`
- `in.rshd`
- `cron`
- `rpc.rexd`
- `in.rexecd`

The initial value for the audit ID is the user ID at the time of login. This is carried across all actions by the user between login and logout, including changing to another username (such as `su`). Thus the system administrator can trace all actions by a specific user regardless of the user's attempts to hide or change identities.

Actions that can be audited are defined as *audit events*. These are listed in the `/etc/security/audit_event` file and are grouped into classes called *audit classes* to aid in administration. Both audit events and audit classes are configurable. In addition, the `audit_control` file for each host contains the default audit values which apply to anyone logging in to that host. Further, some features are set automatically whenever a new user logs in. You can change these features for each user by editing the appropriate `audit_user` file.

Auditing features are set by audit flags, which are discussed in “Audit Flags” on page 355 in this chapter. Specifying what to audit is a process of choosing which flags to add and which flags to disable.

To prevent security breaches, set up the system defaults for auditing before enabling any users. This sets at least a base level of auditing. Once you have set up an account for a user and unlocked it, you should immediately configure the security attributes and individualize the auditing values for this user.

What Files Can I Change?

While some values in files used for audit configuration are set to defaults when the system first boots, all of the audit files in `/etc/security` (`audit_class`, `audit_control`, `audit_event`, `audit_user`, and `audit_warn`) can be adjusted by the administrator. Further, the default values can also be changed. In addition, there are values that can be set that govern machine-level and user-level characteristics.

Default Values

You can change default values in the following files (located in `/etc/security`):

File	Contents
<code>audit_class</code>	Specifies class definitions in the form <code>mask:name:description</code> , where each class is represented as a bit in the mask. See <code>audit_class(4)</code> .
<code>audit_control</code>	Contains audit control information used to determine where to store audit log files, how much free space to allow, and what audit flags are in effect. See <code>audit_control(4)</code> .
<code>audit_event</code>	Contains audit event definitions and specifies the event-to-class mapping. See <code>audit_event(4)</code> .
<code>audit_startup</code>	Script used to initialize the audit subsystem before the audit daemon is started. Initially it consists of a series of <code>auditconfig</code> commands to set the system default policy and download the initial event-to-class mapping. See the <code>audit_startup(1M)</code> man page for more information.

File	Contents
audit_user	Database that stores per-user auditing preselection data. For more information see <code>audit_user(4)</code> .
audit_warn	A script that processes warning or error messages from the audit daemon. The system administrator can specify a list of mail recipients to be notified when an <code>audit_warn</code> situation arises. Users that make up the <code>audit_warn</code> alias are typically the audit and root users. See <code>audit_startup(1M)</code> .
policy.conf	Specifies the default set of authorizations granted to all users. See <code>policy.conf(4)</code> .

Machine-Level Audit Values

Each machine has an associated set of audit characteristics which is used to control the generation of audit records due to nonattributable events. This lets the system handle preselection of asynchronous/nonattributable events and event ownership in the same way that it handles user events. However, unlike a normal user, a machine does not log in; its audit characteristics (preselection mask) are established during system boot by the system initialization script which is pointed to by the `audit` script in `/etc/init.d`. The machine audit characteristics can be adjusted by the system administrator, using the `auditconfig` utility. Note that the asynchronous events identify themselves as being generated by the machine and thus provide a form of identification.

Presently there are only three nonattributable events defined. They are:

1. system boot (always generated)
2. entering the PROM/kernel debugger (normally not generated), and
3. exiting the PROM/kernel debugger (normally not generated).

An evaluated system (one that has been certified as having met the configuration specified in "Common Criteria") requires that the PROM be put into "*security-mode=full*" state. This prevents the user from entering the PROM/kernel debugger. It also limits the disk that may be booted to one specified by the PROM parameters. These are described in the man page for `eeeprom(1M)`.

User-Level Audit Features

User-level audit characteristics are used to identify a user and control generation of audit records. These audit characteristics are set at initial login and consist of the following:

- Process preselection mask
- Audit ID
- Audit Session ID
- Terminal ID (port ID, machine ID)

A user's audit characteristics are defined on a per process basis. This means that each process can have a distinct and different set of values assigned to it. On the other hand, all threads within a process share the same audit characteristics. Whenever a thread operation results in an auditable event, the process audit characteristics are used to determine if an audit record is to be generated and to identify the originator of the record. The values for these audit characteristics are set when the process is created, and are taken from the parent process. They remain unchanged during the life of the process, except when changed by an audit-related system call. These system calls require super user privileges, which prevents a normal user from modifying or accessing the values of individual audit characteristics. Audit characteristic propagation is discussed in subsequent sections of this chapter.

Process Preselection Mask

When a user logs in, the `login` command combines the machine-wide audit flags from the `audit_control` file with the user-specific audit flags (if any) from the `audit_user` file. The result is the *process preselection mask* for the user's processes. This process preselection mask consists of two 32-bit integers specifying whether events in each audit event class are to generate audit records.

Audit flags from the `flags:` line in the `audit_control` file are first combined with the flags from the *always-audit* field in the user's entry in the `audit_user` file. Then the flags from the *never-audit* field in the user's entry in the `audit_user` file are subtracted from the set of flags. If both masks are empty, no audit records are generated.

The process preselection mask does not change regardless of changes of user ID, programming forks, or execution of other programs. Even if the user changes ID through the use of `su` the original mask is ORed with the new mask, and the new user ID is then audited for both events related to the original user and events related to the new user.

Audit ID

Each time a user logs in the resulting process acquires a unique user ID, also called the *audit ID*. (This is in addition to the standard UNIX *real user ID*, which is assigned at login, and *effective user ID*, which is what you would be using if you changed your ID, such as changing to Superuser.) This audit ID is inherited by all child processes started by that user's initial process. Unlike the real user ID and the effective user ID, from

login to logout the audit ID remains the same for that user. This lets a system administrator trace actions back through a series of processes to identify the user who originally logged in.

Audit Session ID

The audit session ID is assigned at login and is inherited by all descendant processes. Unlike the audit ID, the audit session ID tracks the actions during a particular session. The audit session ID uses the terminal ID (that indicates which port was used at login time), combined with other data to help a system administrator distinguish multiple logins from the same user on the same port.

Terminal ID

The terminal ID consists of the host IP address and a unique number that identifies the terminal I/O port through which the user originally logged in. Most of the time the login is through the console, which is usually identified as device 0. For instance, the terminal ID for a local host with the IP address of 179.179.179.178 could be 16777. If the initial login is a remote login from another system, the TCP/UDP pair is used as the terminal I/O port and the machine ID is the IP address of the remote machine.

Determining Which Systems to Audit

Deciding what to audit is usually governed by the kind of risk model your site is using.

If, for instance, you are running a certified installation (such as Common Criteria), you must audit each machine within your “trusted computing base.” The documentation for Common Criteria state what must be audited. You will also be interested in failure to authenticate at login time and failure to access data, as well as who was logged in for what time. You will also be interested in auditing any administrative actions.

On the other hand, if you are a bank you will have a risk model that determines on which systems you are willing to pay the performance and data storage costs costs to collect audit data. Many banks have already determined what is worth auditing, a set of criteria that is usually found in their risk assessment procedures.

If you are connected to the Internet, you should be interested in intrusion detection. There are two main types of intrusion detection you will want to check:

- Through some means, you have determined that an unauthorized user has connected to your machine. In this case, you will want to audit logins and logouts, file access, and any `exec` actions to see what this user is doing.

- Using the sort capabilities of `auditreduce`, you process the audit logs looking for patterns associated with a breakin. Some special intrusion detection programs can also be used to determine which events to generate, which may also help in detecting an intruder.

Rarely will you want to have auditing running constantly for all users. More likely you will want to focus just on `root`.

Determining Which Audit Policies to Use

Audit policies refer to the auditing options that are enabled or disabled for a particular configuration. You can inspect or change the current audit policy with the `auditon` system call at the program level, or by typing `auditconfig` from the command line.

Generally, the default audit policies are set up to minimize the amount of audit information and thus minimize the amount of audit storage resources required. You can determine if the extra information is worth the extra overhead in audit trail size and system resources to generate the audit records.

The following policies may be enabled or disabled:

Policy name	Default setting	What it does	Why should I change it?
<code>Audit_cnt</code>	Block when an audit record can not be put onto the audit trail.	Blocks a user or application when audit records can not be added to the audit trail. When this policy is active, the event is allowed to complete without an audit record being generated. A count of audit records that have been dropped is maintained.	In the default (no) case, when you run out of room to record audit records, all processes stop and wait for space to be freed. This option makes sense in an environment where security is paramount. In the "yes" case, processes keep running resulting in audit records being lost. This makes sense when system availability is more important than security.

Policy name	Default setting	What it does	Why should I change it?
Audit_halt	Halt the machine if a nonattributable audit record can not be added to the audit trail.	Drop the audit record and keep a count of the ones dropped.	Since there is no process context for these events, there is no way to suspend processing until space becomes available for more audit records. When this policy is active, the machine halts if a nonattributable audit record can not be added to the audit trail.
Audit_argv	Add the arguments of an executed program script to the "exec" audit record. When this policy is active, the arguments supplied to the exec system call are included in the audit record.	Do not include program arguments.	If turned on, when you use the <code>execv</code> command, the resulting audit records contain much more data than if this is off. If you are auditing only a few users, it may be reasonable to turn this on.
Audit_arge	Add the environment variables for the executed program to the "exec" audit record. When this policy is active the environment variables are included with the audit record.	Do not include the environment variables in the audit record.	If turned on, when you use the <code>execv</code> command, the resulting audit records contain much more data than if this is off. If you are auditing only a few users, it may be reasonable to turn this on.

Policy name	Default setting	What it does	Why should I change it?
Audit_seq	Add a sequence token to the audit record. This is a monotonically increasing number.	Do not add a sequence number to the audit record.	There is a trade off between more data, but with a sequence number for each record, and a smaller data set. If this is "yes" then each audit record gets a sequence number in the audit log. In the case of file corruption (for example, a partially written audit record) you may be able to spot bad records faster if the sequence numbers are out of order or missing..
Audit_group	Add the groups list from the process credential to the audit record. When this policy is active, the groups list is included with the audit record as a special token.	Do not include this information.	If turned on, when you use the <code>execv</code> command, the resulting audit records contain much more data than if this is off. If you are auditing only a few users, it may be reasonable to turn this on.
Audit_trail	Add a trailer token at the end of an audit record to delimit it. When this policy is active, a trailer token is added to the end of each audit record.	Do not include a trailer token.	There is a trade off between more data and a more robust data set. If this is "yes" then each audit record's end is clearly marked in the audit log. In the case of file corruption (for example, a partially written audit record) <code>auditreduce</code> will cause a resync faster on good records.

Policy name	Default setting	What it does	Why should I change it?
Audit_path	Allow multiple paths per audit record. When this policy is active, each file name or path used during a system call causes a path token to be added to the audit record.	Allow one path per audit record.	If turned on, when you use the <code>execv</code> command, the resulting audit records contain much more data than if this is off. If you are auditing only a few users, it may be reasonable to turn this on.

Determining Audit Events and Classes

Certain actions that occur are sufficiently important that they are considered auditable actions. These actions are called *auditable events*. When an audit event occurs, an audit record will be generated if the event is selected by the selection mechanism.

The audit events recognized are defined in an administrative database located in `/etc/security/audit_event`. This file lists both the audit events generated within the kernel and the audit events generated by the applications. Each entry in the file represents a particular audit event and contains four pieces of information:

- the unique audit event number of the event,
- the event name,
- a description of the event which is used when displaying the audit event, and
- the list of classes to which the event has been assigned.

The assignment of events to classes is done by the system administrator (see Chapter 24) and may be modified by the system administrator to fit the particular requirements of the site.

```
event no.   event name   event description   class flags
...
7:AUE_EXEC:exec(2):pc,ex
8:AUE_CHDIR:chdir(2):fc
9:AUE_MKNOD:mknod(2):ad
...
6259:AUE_su:su:lo
6260:AUE_halt_solaris:halt:ad
6162:AUE_reboot_solaris:reboot:ad
...
```

In this example, the first entry listed is the entry for the `exec` system call. The event ID which identifies the audit record is 7. The event name of the audit event is `AUE_EXEC`. When the audit record is displayed by `praudit`, the string `exec2` is

displayed instead of a 7, meaning that an audit event has been generated by this event if a user is being audited for either the *pc* (process) or *ex* (execution) audit classes.

Audit events are grouped into *classes* in order to simplify the administration of preselection (see “Process Preselection Mask” on page 348). Each audit event is assigned to one or more audit classes, which form the basis for selective auditing. To support site specific needs, these groupings may be configured by the system administrator and new classes defined. The system call `auditon` and the `auditconfig` utility lets the system administrator inspect and override the default settings. The resulting `audit_event` table is downloaded into the kernel at boot time.

Modification of class information takes effect immediately for all events generated in the kernel. For auditing performed at the user level, a change in the configuration of the event-to-class mapping affects all processes created after the modification occurred.

An audit class is implemented as a bit in a vector. The current implementation of Solaris supports 32 distinct classes (the size of an integer) plus the designations *all* and *none*. This lets the user of logical instructions perform masking operations and quickly determine if an audit class is present.

Class descriptors are defined in the file `/etc/security/audit_class`. This file maps the relationship between a mask name and the associated mask value used for preselection. Each entry in this file represents an event class and contains three pieces of information:

- the event mask specifying a bit in a mask,
- the name of the event class which is used in the `audit_event` file, and
- a description of the class.

The mapping between class and bit is determined by the system administrator.

```
class mask      class name      class description
...
0x00000020:fd:file delete
0x00000040:cl:file close
0x00000080:pd:process
0x00000100:nt:network
```

In this example, the first entry specifies the `fd` audit class. The mask used for preselection operations is defined as `0x00000020`. The identifier for the audit class is `fd`, which is used to specify the class in the `audit_event` file. (This class is for specifying audit events involving *file delete* operations.)

Note – The interpretation of the meaning of the audit classes is purely arbitrary and depends on the events that are assigned to a particular class. In the example above, the events involving a “deletion” of a file system object have been assigned to the `fd` class, but the same action could be called by another name in another class. This is an important concept to keep in mind when assigning and defining a site-specific audit class.

Site-Specific Event-to-Class Mapping

The `/etc/security/audit_event` file contains a list of all defined audit event types. For each event, it defines its class, an arbitrary grouping of similar events. The class of events can be changed to suit your particular needs and new classes may be defined. Any class referenced by an entry in `audit_event` must be defined in `audit_class`.

In the example below, the final value (*fc*) is the class for the audit event `AUE_CREAT`. The *fc* value may be changed to any other class defined in `audit_class` or may be modified to be a list of classes separated by commas (*fc,fa*):

```
4:AUE_CREAT:creat(2):fc
```

You can change the event-to-class mapping by editing the `/etc/security/audit_event` file to reflect how each event is related to a class. You will need to reboot the system or run `audit_config` with the `-conf` option to change the runtime kernel event-to-class mappings.

Audit Flags

Audit flags indicate classes of events to audit. Machine-wide defaults for auditing a specified for all users on each machine by flags in the `audit_control` file, which is described in “The `audit_control` File” on page 383 in Chapter 25.

The system administrator can modify what is audited for individual users by putting audit flags in a user’s entry in the `audit_user` file. The audit flags are also used as arguments to `auditconfig` (see the `auditconfig(1M)` man page).

Table 25–1 in Chapter 25 shows the short and long names for each audit class and gives a description of what each does.

Depending on the prefixes, a class of events can be audited whether it succeeds or fails, or only if it succeeds or only if it fails. The format of the audit flag is shown below.

prefixflag

Table 23-1 shows prefixes that specify whether the audit class is audited for success or failure or both.

TABLE 23-1 Prefixes Used with Audit Flags

Prefix	Definition
none	Audit for both success and failure.
+	Audit for success only.
-	Audit for failure only.

For example, the audit flag `lo` means “all successful attempts to log in and log out and all failed attempts to log in.” (You cannot fail an attempt to logout.) For another example, the `-all` flag refers to all failed attempts of any kind, and the `-+all` flag refers to all successful attempts of any kind.

Note – The `-all` flag can generate large amounts of data and fill up audit file systems quickly, so use it only if you have extraordinary reasons to audit everything.

Determining Where to Store Audit Files

There are two types of audit files: audit configuration files and audit log files.

The `/etc/security` directory contains subdirectories with all of the audit configuration files. It also contains several other files related to audit control.

The `/etc/security` directory must be part of the root file system, because it contains the per-machine `audit_data` file, which must be available for successful startup of the audit daemon at boot time.

By default, the audit post-selection tools look in directories under `/etc/security/audit`. Hence the path name of the mount point for the first audit file system on an audit server is of the form: `/etc/security/audit/server-name`, where `server-name` is the name of the audit server. If more than one audit partition is on an audit server, the name of the second mount point is: `/etc/security/audit/server-name.1`, the third is `/etc/security/audit/server-name.2`, and so forth.

Audit log files, however, are usually located in different directories or partitions. While each machine being audited must have at least one audit log file located locally, other audit log files may be located anywhere that the machine can read and write, such as

dedicated audit partitions on other machines or disks. These directories must be close to the root directory on their respective machines, such as `/var/audit` or `/var/audit.1`.

Ensure that each audit log directory contains nothing except audit files.

The `/etc/security` directory on each auditable machine must contain at least one of the audit directories, referred to as the *directory of last resort*. This is because, if the network goes down, you need to have an audit directory defined on the local machine. Otherwise either all jobs will stop running or all audit records will be discarded, depending on how you configured auditing.

Other audit directories may be located on this machine or elsewhere on the network, depending on your site's configuration and preferences. These directories of audit log files are generally known as the *primary directory* and any *secondary directories*. These are explained in more detail later in this chapter.

The audit log directories can be located anywhere on the network that is visible to the audit configuration directory. Pointers to these directories are contained in the `audit_control` file, and you can have as many audit log directories as you have available space. However the last pointer in the `audit_control` file must point to the directory of last resort.

Many system administrators find it useful to create partitions that are exclusively used for keeping audit logs, thus simplifying tracking a user's behavior from audit file to audit file. Further, using a dedicated partition for audit logs means that there is no danger of application data accidentally overwriting an audit log, or with auditing operations interfering with the operation of other applications.

If you have a distributed network, you may also want to use several disks or partitions to store audit logs. This lets you have some assurance of backup if one of the disks or partitions gets corrupted.

Primary and Secondary Audit Directories

The primary audit directory is the first directory where a machine places its audit log files. Each host must have a primary audit directory and may have a number of secondary directories. However, the primary and secondary directories may be NFS-mounted, that is, not local to the host.

A secondary audit directory is where a machine places audit files if the primary directory is full or inaccessible because of network failure, an NFS server crash, or for some other reason. There can be one or many secondary directories for a particular host.

A host's primary and secondary directories are not necessarily located on the host. Indeed, many installations place their audit directories on machines other than those

hosts that handle user applications, on the theory that if the host running applications goes down, at least the audit records will be safe.

A directory of last resort is the directory used if the other directories are full or inaccessible. It must be located on the machine being audited, and its use triggers a warning to the console.

Setting Up or Modifying Audit Directories

In order for a directory to be used to store audit log files, it must be accessible to the audit daemon. This means that the following conditions must be met:

- It must have a pointer in the appropriate `audit_control` file.
- The directory must be mounted. (For NFS-mounted directories, set the option `noac` to obtain the correct behavior when an audit partition fills; otherwise audit records may be lost when moving to a new partition.)
- The network connection (if remote) must permit successful access.
- The permissions on the directory must allow access.
- It must have sufficient free space remaining.

The first condition is covered by the `dir:` entries in the `audit_control` file. The second, third, and fourth conditions are outside the scope of auditing, but are concerns of the system administrator.

The fifth is handled by the `minfree` value in the `audit_control` file (see “The `audit_control` File” on page 383. The default value of 20 means that whenever a file system becomes more than 80 percent full, an email notice is sent to the `audit_warn` alias. When the current directory has insufficient space left for more audit files, the next directory listed is used. When no directories on the list have sufficient free space left, the audit daemon starts over from the beginning of the list and picks the first accessible directory that has any space available, until the hard limit is reached. In the default configuration, if no directories are available, the daemon stops processing audit records and they accumulate within the kernel until all processes generating audit records are suspended.

See “Device Allocation” on page 360 later in this chapter.

Determining Audit Space Usage

Storage costs are the most significant auditing cost, and are directly related to the amount of data you collect. The amount of audit data depends on the following:

- Number of users
- Number of machines

- Amount of use
- Degree of security required

Because the factors vary from one situation to the next, no formula can determine in advance the amount of disk space to set aside for audit data storage.

Full auditing (with the *all* flag) can fill up a disk quickly. Even a simple task like compiling a program of modest size (for example, 5 files, 5000 lines total) in less than a minute could generate thousands of audit records, occupying many megabytes of disk space. Therefore it is very important to use the preselection features to reduce the volume of records generated. For example, just omitting the *fr* class can reduce the audit volume by more than two-thirds. Efficient auditing file management is also important after the audit records are created, to reduce the amount of storage required.

“Using the Audit Data” on page 367 gives some tips about reducing the costs of storage by selectively auditing in order to reduce the amount of audit data collected, while still meeting your site’s security needs. Also discussed in those sections are how to set up audit file storage and archiving procedures to reduce storage requirements.

Before configuring auditing, be sure you understand the audit flags and the types of events they flag. As you set up your auditing configuration, remember that each new type of audit file will take up storage space and analysis time.

Unless the process audit preselection mask is modified dynamically, the audit characteristics in place when a user logs are inherited by all processes during the login session. Unless the databases are modified, the process preselection mask applies it in all subsequent login sessions.

Each process has two sets of one-bit flags for audit classes. One set controls whether the process is audited when an event in the class is successfully requested; the other set controls whether an event is requested but fails for any reason. Processes are commonly more heavily audited for failures than for successes, since the failures tend to indicate attempts at browsing and other attempts to violate system security. The process preselection mask thus should be carefully watched to see whether the data it is generating is truly useful.

Dynamic controls refer to controls put in place by the administrator while processes are running. These persist only while the affected processes and any of their children exist, but will not continue in effect at the next login. Dynamic controls apply to one machine at a time, since the `audit` command only applies to the current machine where you are logged in. However, if you make dynamic changes on one machine, you should make them on all machines at the same time.

Device Allocation

How you implement auditing depends to a large extent on how you allocate devices available to you on the network. This section takes you through the elements of device allocation as regards auditing.

Managing Device Allocation

The components of the allocation mechanism that you must understand in order to manage device allocation are:

- How the allocation mechanism works.
- The `allocate`, `deallocate`, `dminfo`, and `list_devices` commands.
- The `/etc/security/device_allocate` file (see the `device_allocate(4)` man page).
- The `/etc/security/device_maps` file (see the `device_allocate(4)` man page).
- The lock files that must exist for each allocatable device in `/etc/security/dev`.
- The changed attributes of the files that are associated with each allocatable device.
- Device-clean scripts for each allocatable device.

The `device_allocate` file, the `device_maps` file, and the lock files are specific to each machine. The configuration files are not administered as NIS databases because tape drives, diskette drives, and the printers are all connected to specific machines.

How the Allocate Mechanism Works

The `allocate` command first checks for the presence of a lock file under the device name for the specified device in the `/etc/security/dev` directory. If the file is owned by `allocate`, then the system changes ownership of the lock file to the name of the user entering the `allocate` command.

The system then checks for an entry for the device in the `device_allocate` file, and checks whether the entry shows the device as allocatable.

The first listing in the screen example below shows that a lock file exists with owner `bin`, group `bin`, and mode `600` for the `st0` device in `/etc/security/dev`. The second part of the listing shows that the associated device-special files are set up properly, with owner `bin`, group `bin`, and mode `000`:


```

peaches % ls -lg /etc/security/dev/st0
-rw----- 1 bin bin          0 Dec 6 14:21 /etc/security/dev/st0
peaches % ls -lg /devices/sbus@1,f800000/esp@0,80000
c----- 1 bin bin          18,  4 May 12 13:11 st@4,0:
c----- 1 bin bin          18, 20 May 12 13:11 st@4,0:b
c----- 1 bin bin          18, 28 May 12 13:11 st@4,0:bn
c----- 1 bin bin          18, 12 May 12 13:11 st@4,0:c
.
.
.
c----- 1 bin bin          18,  0 May 12 13:11 st@4,0:u
c----- 1 bin bin          18, 16 May 12 13:11 st@4,0:ub
c----- 1 bin bin          18, 24 May 12 13:11 st@4,0:ubn
c----- 1 bin bin          18,  8 May 12 13:11 st@4,0:un

```

In the screen below, user *vanessa* allocates device *st0*:

```

peaches % whoami
vanessa
peaches % allocate st0

```

When user *vanessa* enters the `allocate` command to allocate the tape *st0*, the system first checks for the existence of an `/etc/security/dev/st0` file. If no lock file exists or if the lock file is owned by a user other than `allocate`, then *vanessa* cannot allocate the device.

In the example below, the default `device_allocate` entry for the *st0* device specifies that the device is allocatable. Because the `allocate` command finds that all the above conditions are met, the device is allocated to *vanessa*.

The system then changes the ownership and permissions of the device-special files associated with the device in the `/dev` directory. To allocate the *st0* device to *vanessa*, the mode on its associated device-special files is changed to `600`, and the owner is changed to *vanessa*.

The system then also changes the ownership of the lock file associated with the device in the `/etc/security/dev` directory. To allocate the *st0* device to *vanessa*, the owner of `/etc/security/dev/st0` is changed to *vanessa*.

After the user *vanessa* executes the `allocate` command using the device name *st0*, the following example shows that the owner of `/etc/security/dev` is now changed to *vanessa*, the owner of the associated device-special file is now *vanessa* as well, and *vanessa* now has permission to read and write the files.

```

peaches % whoami
vanessa
peaches % allocate st0
peaches % ls -lg /etc/security/dev/st0
-rw----- 1 vanessa staff    0 Dec 6 15:21 /etc/security/dev/st0
peaches % ls -la /devices/sbus@1,f8000000/esp@0,800000
.
.
.

```

```

crw----- l vanessa 18,      4 May 12 13:11 st@4,0:
crw----- l vanessa 18,      12 May 12 13:11 st@4,0:b
crw----- l vanessa 18,      12 May 12 13:11 st@4,0:bn
crw----- l vanessa 18,      12 May 12 13:11 st@4,0:c
.
.
.
crw----- l vanessa 18,      4 May 12 13:11 st@4,0:u
crw----- l vanessa 18,      12 May 12 13:11 st@4,0:ub
crw----- l vanessa 18,      12 May 12 13:11 st@4,0:ubn
crw----- l vanessa 18,      12 May 12 13:11 st@4,0:un

```

Risks Associated with Device Use

Solaris' device allocation mechanism is designed to correct security lapses when multiple users have the right to access the same storage device. For instance, consider how cartridge devices are typically used.

Often several users share a single tape drive, which can be located in an office or lab away from where an individual user's own machine is located. Once a user loads a tape into the tape drive, some length of time can elapse before the user can return to the machine to invoke the command that reads or writes data to or from the tape. Then another time lapse can occur before the user is able to take the tape out of the drive. Because tape drives are typically accessible by all users, during the time when the tape is unattended some unauthorized user can access or overwrite data on the tape.

Solaris' device allocation mechanism makes it possible to assign certain devices to one user at a time, so that the device can only be accessed by that user while it is assigned to that user's name. The device allocation mechanism ensures the following for tape devices and provides related security services for other allocatable devices:

- Prevents simultaneous access to a device
- Prevents a user from reading a tape just written to by another user, before the first user has removed the tape from the tape drive.
- Prevents a user from gleaning any information from the device's or the driver's internal storage after another user is finished with the device.

Using Device Allocation

The commands explained in this section generally describe how to manage devices and how to add devices. For specific procedures, see Chapter 24. You must enter the device allocation and device deallocation commands from the command line in a Command Tool or Shell Tool window:

- `allocate` assigns a device to a user. You can specify the device in either of two ways:
 - `device-name` allocates the device that matches the device name.
 - `-g device-type` allocates the device that matches the device group type.
 - `deallocate` releases a previously allocated device.
 - `list_devices` lets you see a list of all allocatable devices, devices currently allocated, and allocatable devices not currently allocated. This command requires one of these three options:
 - `-l` lists all allocatable devices or information about the device.
 - `-n` lists all devices not currently allocation or information about the device.
 - `-u` lists devices currently allocated or information about the device.
-

Setting Up an Archive Policy

In the Solaris environment, an archive is a file or set of files that is stored for historical purposes. Most often, auditing archives are copies of audit log files that are no longer current. The primary and secondary audit log file directories you specify in the `audit_config` file are good for storing current audit log files, but can fill up quickly. To reuse that directory space, you should create and use an archive policy that periodically copies the audit log files to tape or some other medium, and then move that medium to offline storage once it is full. The archive medium should be labeled and kept for six months to allow for research, then discarded or reused.

What you include in your archive policy depends on the degree of security needed. Trusted Computer Solaris installations (usually government-related sites) are required to track use of identification and authorization mechanisms, a user's use of objects such as file open and program initiation, file deletion, and actions taken by computer operators, system administrators, and system security officers, among other data and have specific requirements regarding archiving. Banking installations may need different auditing and archive policies. Small enterprises without many security risks may need much less audit data.

However a good archive policy gives you historical data that can help you identify the source of problems. A good archive policy includes the following:

- A plan that is shared by all system administrators using the same network.
- Regular moving of audit logs from current directories to archive directories. This is usually implemented by a `cron` job.
- Periodic checking of the archive files for any suspicious entries.

- Storage of the offline archives in an off-site location, such as another office or storage area, that most users can't access, but which system administrators and security personnel can use if necessary.

Warnings

One of the space management elements you can change is the level of warning messages you get and where they go. This section briefly explains the warning levels, the script that is used to implement them, and how you determine who gets the warning messages.

Determining the Warning Levels

There are several kinds of warnings issued when space becomes a problem. The `audit_control` file contains the parameters that determine when warnings about low available space are sent and to whom they are sent.

A typical `audit_control` file might look like this:

```
dir:/var/audit
flags: lo,ad,-all
minfree:25
naflags:10
```

The *audit threshold* line (`minfree:`) defines the minimum free-space percentage level for all audit file systems. The `minfree` percentage must be greater than or equal to 0. While the default is 20 percent, in the example above, the `minfree` value is 25 percent.

The `minfree` value (here 20 percent) specifies that the warning script is to be run when the file systems are 80 percent filled, and the audit data for the current machine will be stored in the next available audit directory, if any. The warning script is located in `audit_warn` (see the `audit_warn(1M)` man page).

The `minfree` value is only one type of space warning you may see. It is considered a “soft” warning, since the message associated with it is only advising that you’re getting close to partition limits, not that you’ve exceeded any of them. You may also see other types of warnings.

The audit_warn Script

Whenever the audit daemon encounters an unusual condition while writing audit records, it invokes the `audit_warn` script (see the `audit_warn(1M)` man page). Located in the `/etc/security` directory, this script can be customized to handle situations automatically or to warn of conditions that may require manual intervention. When invoked, `audit_warn` writes a message to the console and sends a message to the `audit_warn` alias.

When any of the following conditions are detected by the audit daemon, by default it invokes `audit_warn`:

- An audit directory has reached or exceeded the free space allowed by *minfree*. This is considered the “soft” limit.

The `audit_warn` script is invoked with the string *soft* and the name of the directory whose available space has reached or exceeded the minimum level. The audit daemon automatically switches to the next suitable directory and writes the audit files there until this new directory reaches its *minfree* limit. The audit daemon then writes audit records in the next directory in the order listed in `audit_control` until it reaches the *minfree* limit.

- All the audit directories are more full than the *minfree* threshold.

In this case, the `audit_warn` script is invoked with the string *allsoft* as an argument. The audit daemon writes a message to the console and sends email to the `audit_warn` alias.

When all audit directories listed in `audit_control` are at their *minfree* limits, the audit daemon switches back to the first one and writes audit records until the directory fills completely.

- All the audit directories are completely full. This is known as the “hard” limit. The audit daemon invokes the `audit_warn` script with the string *allhard* as an argument.

In the default configuration, the audit daemon writes a message to the console and sends mail to the `audit_warn` alias. Processes generating audit records are suspended. The audit daemon goes into a loop, waiting for space to become available, and resumes processing audit records when that happens. While audit records are not being processed, no auditable activities take place — every process that attempts to generate an audit record is suspended.

This is one reason why you will want to set up a separate audit administrator account that can operate without any auditing enabled. As an administrator, you can then operate the system without your activities being suspended.

- An internal error occurs because another audit daemon process is already running (string *ebusy*), a temporary file cannot be used (string *tmpfile*) or a signal was received during auditing shutdown (string *postsigterm*). Mail containing the string is sent to the `audit_warn` alias and a message is sent to the console.

- A problem is discovered with the `audit_control` file's contents. By default, mail is sent to the `audit_warn` alias and a message is sent to the console.

During a system panic or crash, the in-core image of the operating system is written to disk. `audit_warn` does not cover these situations. However you can use `savecore(1M)` and `savecore(1M)` to recover the audit data which may be buffered at the time of system failure but not yet written to the audit file.

When Should I Change Audit Parameters?

Most organizations use the following techniques to help them manage auditing efficiently:

- Perform random auditing only on a certain percentage of users at any one time.
- Monitor audit data in real time for unusual behaviors. You can set up procedures to monitor the audit trail as it is generated for certain activities and to trigger higher levels of auditing of particular users or machines when suspicious events occur.
- Reduce the storage requirements for audit files by combining, reducing, and compressing them and by developing procedures to store them offline. See "Controlling the Cost of Auditing" on page 367 later in this section.

In addition to supplying the per-user audit control information in the static databases, you can dynamically adjust the state of auditing while a user's processes are active on a single machine.

The `auditconfig` command provides a command line interface to get and set audit configuration parameters. To change the audit flags for a specific user to a supplied value, use the `auditconfig` command with the `-setpmask`, `-setsmask`, or `-setumask` options. The command changes the process audit flags for one process, one audit session ID, or one audit user ID respectively. See the `auditconfig(1M)` man page.

Using the Audit Data

It is unfortunately very easy to inadvertently create auditing parameters in such a way that your time as administrator is largely spent setting up and inspecting audit logs. To avoid this, it pays to set up auditing policies such that you are most likely to capture unusual behavior while keeping storage requirements for audit logs to reasonable levels. This section discusses techniques for efficiently performing auditing.

Controlling the Cost of Auditing

Auditing consumes system resources. The more audit files you request, the more impact auditing has on overall system performance. Auditing affects your system in three basic ways:

- Increased processing time
- Increased item spent on analysis of data
- Increased storage space for audit data.

Processing Time

The cost of increased processing time is the least significant cost of auditing, primarily because auditing generally occurs in periods when computational-intensive tasks, such as image processing and complex calculations, are not being run. Servers will tend to spend a greater percentage of their time processing auditing tasks than single-user workstations.

Data Analysis

The cost of data analysis is roughly proportional to the amount of audit data collected. This includes the time it takes to merge and review audit records and the time it takes to archive them and keep them in a safe place.

The fewer records you generate, the less time it takes to analyze them. However, the fewer records you generate, the more likely you are to miss something potentially dangerous.

Audit Data Storage Space

Audit log files can quickly use up allocated space unless carefully monitored. Once you have decided that the audit files you are collecting contain information that you really need, you need to consider how to most efficiently reuse the directory space allocated for new audit logs.

To keep audit files to a manageable size, you can set up a `cron` job that periodically switches audit files (see the `cron(1M)` man page.) You can specify intervals from once per hour to twice per day, depending on the amount of audit data being collected. The data can be filtered to remove unnecessary information and then compressed.

In addition to the space necessary for the primary and secondary audit directories and the audit directory of last resort, you should have a place set aside to archive the filtered and compressed audit files. Most installations store these archives on tape or some other removable medium so that the archive files can be physically moved to a separate location, but retrieved if necessary. The script in the `cron` job can also write the archive files to the appropriate device, delete the existing closed files from the audit directory space, and adjust the pointer so that it reflects the newly available space.

Determining When to Merge Audit Records

The `auditreduce` command merges audit records from one or more input audit files. Normally you enter this command from the directory where you keep the audit log files.

Without options, `auditreduce` merges the entire audit trail (all of the audit files in all of the subdirectories pointed to by the `dir:` entries in the `audit_control` file) and sends the merged file to standard output.

Note that this output is not human readable. To get human-readable output, you must use `praudit` to process the file.

Among other things, you can use the `auditreduce` options to:

- Show output containing audit records generated only by certain audit flags.
- Show audit records generated by one particular user.
- Collect audit records generated on specific dates.

For a full description of the `auditreduce` command and its options, see the `auditreduce(1M)` man page.

Changing Audit Trail File Locations

An audit trail consists of a group of audit files. Where these audit files are located is dictated by the directory definition lines (starting `dir:` in the `audit_control` file on each machine.

For example, the contents of a standalone server's `audit_control` file might look like this:

```
flags:lo,ad,-all,^fc
naflags:lo,nt
minfree:20
#
# Primary and secondary audit directories
#
dir:/var/audit/file1
dir:/var/audit/file2
#
# Audit directory used when others are full
#
dir:/var/audit/filelast
```

Since this server has no connections to a network, all audit files must be stored on itself — a somewhat risky situation since if the disk crashes, there is no way to tell what happened.

The contents of server `grape`'s `audit_control` file might point to dedicated audit partitions and directories on other servers, such as `strawberry` and `banana`, as shown in this example:

```
flags:lo,ad,-all,^fc
naflags:lo,nt
minfree:20
#
# Primary and secondary audit directories
#
dir:/etc/security/audit/strawberry1
dir:/etc/security/audit/strawberry2
dir:/etc/security/audit/banana
#
# Audit directory used when others are full
#
dir:/var/audit/
```

Note that the audit file of last resort for `grape` is located on a separate directory on itself.

You can change the directories where you want to store audit files any time you want. You will need to change only the `dir:` entries in the `audit_control` files in the servers affected. However, you must reboot the server where auditing is taking place once you have changed its `audit_control` file.

Preventing Audit Trail Overflow

If the audit file systems fill up, the `audit_warn` script sends a message to the console that the hard limit has been exceeded on all audit file systems, and also sends mail to the alias. By default, the audit daemon remains in a loop sleeping and checking for space until some space is freed. All auditable actions are suspended.

A site's security requirements can be such that the loss of some audit data is preferable to having system activities suspended due to audit trail overflow. In that case, you can create the `audit_warn` script to automatically delete or move audit files or to set the `auditconfig` policy to drop records.

Using the Results from Audit Trail Analysis

This section briefly explains how the system creates an audit trail and how to use the data thus created.

How the Audit Trail is Created

The *audit trail* is created by the audit daemon `auditd` (see the `auditd(1M)` man page) on each machine when the machine is brought up. Once `auditd` starts at boot time, it collects the audit trail data and writes the audit records into *audit files* (also called *audit log files*). See "Audit Record Structure" on page 393 or the `audit(1M)` man page for a description of the file format.

The audit daemon runs as `root`. All files it creates are owned by `root`. It runs continuously, looking for a place to put audit records, even when it has no classes to audit or when the rest of the machine's activities are suspended because the kernel's audit buffers are full.

Only one audit daemon can run at a time. If there is an attempt to start a second one, an error message is generated and the new one exits. If there is a problem with the audit daemon, use `audit -t` to terminate `auditd` gracefully, then restart it by entering `audit -s`.

`auditd` runs the `audit_warn` script whenever the daemon switches audit directories or encounters difficulty (such as a lack of storage). As released, the `audit_warn` script sends mail to an `audit_warn` alias and sends a message to the console. Your site should customize the `audit_warn` script to suit your needs.

When `auditd` starts on each machine, it creates the file `/etc/security/audit_data`. This file consists of a single entry with two fields separated by a colon (see the `audit_data(4)` man page). The first field is the audit daemon's process ID and the second field is the path name of the audit file to which the audit daemon is currently writing audit records. If you want to see what this file looks like, you can use the `cat` command as follows:

```
# cat /etc/security/audit_data
116:/etc/security/audit/grumpy.1/files/
1910320100002.not_terminated.dopey
```

The `not_terminated` appendage means that the file is not finished, and that the system is continuing to write audit records there.

The Audit Daemon's Role

The audit daemon performs several functions:

- It opens and closes the audit log files in the directories specified in the `audit_control` file, in the order in which the directories are specified.
- It reads audit data from the kernel and writes it to an audit file.
- It executes the `audit_warn` script when the audit directories reach limits specified in the `audit_control` file. By default, the script sends warnings to the `audit_warn` alias and to the console.
- It switches to an alternate location for audit log files when it encounters an I/O error or file system full condition. When it creates a file, it inserts an initial audit token that names the previous active audit file of the audit trail (NULL if this is the beginning of an audit trail). When it switches to a new file to continue writing, it terminates the previous file with an audit token that contains the new file name. Upon termination, it closes the active audit file and indicates in the final record that this is the last record of the last file of a sequence of files. These pointers let you follow a chain of events as you move backward and forward in the audit trail.
- Under the default system configuration, when all audit directories are full, processes that generate audit records are suspended. In addition, `auditd` writes a message to the console and to the `audit_warn` alias. (The auditing policy can be reconfigured with `auditconfig`.) At this point, only the system administrator can log in to write audit files to tape, delete audit files from the system, or do other cleanup.

The audit daemon starts whenever the machine is brought up in multiuser mode or when the `audit -s` command instructs the audit daemon to reread the `audit_control` file after it has been edited. The audit daemon then determines the amount of free space necessary and uses the list of directories in the `audit_control` file to determine where to put the audit files.

The audit daemon maintains a pointer into this list of directories. Every time the audit daemon needs to create an audit file, it writes the file into the first available directory in the list, starting at the audit daemon's current pointer. The pointer can be reset to the beginning of the list if you enter the `audit -s` command. If you use the `audit -n` command to instruct the daemon to switch to a new audit file, the new file is created in the same directory as the current file.

Managing Auditing (Tasks)

This chapter presents step-by-step procedures designed to help the system administrator set up and manage a Solaris environment that includes auditing.

If you are trying to solve a specific problem, you may want to skip directly to one or more of these sections.

Configuring Audit Files

Before enabling auditing on your network, you may want to edit the audit configuration files.

▼ How to Change Audit Flags

Audit flags are defined in `/etc/security/audit_control`. The flags control which classes of audit classes are audited and when the classes are audited.

1. **Become superuser or assume an equivalent role.**
2. **Optional: Save a backup copy of the `audit_control` file.**

```
# cp /etc/security/audit_control /etc/security/audit_control.save
```

3. Add new entries to the `audit_control` file.

Each entry has the format:

title : *string*

title

Defines the type of flag

string

Lists specific data associated with the flag

Example — Changing Audit Trail File Locations

Follow the previous procedure. In Step 3 change the lines that start with the `dir` title to change the locations. In this example two entries are added.

```
# grep dir /etc/security/audit_control
dir:/etc/security/audit/host.1/files
dir:/etc/security/audit/host.2/files
dir:/var/audit
```

Example — Changing the Soft Limit for Warnings

Follow the previous procedure. In Step 3 add a line like the following to set the soft limit so that a warning is set when only 10% of the file system is free.

```
minfree:10
```

▼ How to Change User's Audit Characteristics

Definitions for each user may be stored in `/etc/security/audit_user`.

1. Become superuser or assume an equivalent role.
2. Optional: Save a backup copy of the `audit_control` file.

```
# cp /etc/security/audit_user /etc/security/audit_user.save
```

3. Add new entries to the `audit_user` file.

Each entry has the format:

username : *always* : *never*

username

Name of the user to be audited.

<i>always</i>	List of audit classes which should always be audited
<i>never</i>	List of audit classes which should never be audited

Multiple flags may be entered by separating the audit classes with commas.

▼ How to Create Audit Classes

Audit classes are defined in `/etc/security/audit_class`.

1. **Become superuser or assume an equivalent role.**
2. **Optional: Save a backup copy of the `audit_control` file.**

```
# cp /etc/security/audit_class /etc/security/audit_class.save
```

3. **Add new entries to the `audit_class` file.**

Each entry has the format:

```
0xnumber:name:description
```

<i>number</i>	Unique audit class mask
<i>name</i>	Two letter name of the audit class
<i>description</i>	Descriptive name of the audit class

Example — Setting a New Audit Class for Device Allocation

In step 3, add an entry like the following:

```
0x00010000:de:device allocation
```

▼ How to Create Audit Events

Audit event definitions are stored in `/etc/security/audit_event`.

1. **Become superuser or assume an equivalent role.**

2. Optional: Save a backup copy of the audit_event file.

```
# cp /etc/security/audit_event /etc/security/audit_event.save
```

3. Add new entries to the audit_event file.

Each entry has the format:

```
number : name : description : classes
```

<i>number</i>	Unique audit event number — must start after 32768
<i>name</i>	Unique audit event name
<i>description</i>	audit event description — often includes the name of the man page
<i>classes</i>	event classes that this event should be included in

Configuring an Audit Server

▼ How to Create Partitions for Auditing

1. Determine the amount of space that is required.

Assign at least 200 MB of space per host. However, the disk space requirements are based on how much auditing you perform and may be far greater than this figure. Remember to include space for a directory of last resort if the server is going to be audited.

2. Create dedicated audit partitions as needed.

This step is most easily done during server installation. It is also possible to create the partitions on disks that have not been mounted on the server yet. See “Creating a UFS File System” in *System Administration Guide: Basic Administration* for full instructions on how to create the partitions.

```
newfs /dev/rdisk/cwtxdysz
```

```
/dev/rdisk/cwtxdysz    Raw device name for the partition
```

If the server is to be audited, create an audit directory of last resort.

3. Create mount points for each new partition.

```
mkdir /var/audit/servername.n
```

servername.n Use the servername and a number to identify each partition

4. Add entries to automatically mount the new partitions.

Add a line to `/etc/vfstab` like:

```
/dev/dsk/cwtxdysz /dev/rdisk/cwtxdysz /var/audit/servername.n ufs 2 yes
```

5. (Optional) Reduce the minimum free space threshold on each partition.

```
tunefs -m 0 /var/audit/servername.n
```

servername.n Use the servername and a number to identify each partition

6. Mount the new audit partitions

```
mount /var/audit/servername.n
```

7. Create audit directories on the new partitions.

```
mkdir /var/audit/servername.n/files
```

8. Correct the permissions on the mount points and new directories.

```
chmod -R 750 /var/audit/servername.n/files
```

Example — Creating New Audit Partitions

In this example, a new file system is created on two new disks.

```
# newfs /dev/rdsk/c0t2d0
# newfs /dev/rdsk/c0t2d1
# mkdir /var/audit/egret
# mkdir /var/audit/egret.1
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1 /dev/rdsk/c0t2d0s1 /var/audit/egret ufs 2 yes -
/dev/dsk/c0t2d1s1 /dev/rdsk/c0t2d1s1 /var/audit/egret.1 ufs 2 yes -
# tunefs -m 0 /var/audit/egret
# tunefs -m 0 /var/audit/egret.1
# mount /var/audit/egret
# mount /var/audit/egret.1
# mkdir /var/audit/egret/files
# mkdir /var/audit/egret.1/files
# chmod -R 750 /var/audit/egret/files /var/audit/egret/files.1
```

Setting Up Auditing

The procedures shown in this section help you implement auditing.

▼ How to Enable Auditing

1. **Become superuser.**

2. **Bring the system into single-user mode.**

See the `telinit(1M)` man page for more information.

```
# /etc/telinit 1
```

3. **Run the script to configure the system to run auditing.**

Go to the `/etc/security` directory and execute the `bsmconv` script there. The script sets up a standard Solaris machine to run BSM after a reboot. See the `bsmconv(1M)` man page.

```
# cd /etc/security
# ./bsmconv
```

4. **Bring the system into multi-user mode.**

The startup file in `/etc/security/audit_startup` causes the audit daemon to run automatically when the system enters multiuser mode.

```
# /etc/telinit 6
```

Note – The `bsmconv` script adds a line to the `/etc/system` file which disallows aborting the system with the Stop-A keyboard sequence. If you want to retain the ability to abort the system with the Stop-A keyboard sequence, you must comment out the line in the `/etc/system` file that reads `set abort_enable =0`.

▼ How to Disable Auditing

If at some point BSM is no longer required, you can disable it by running `bsmunconv` (see the `bsmconv(1M)` man page).

1. **Become superuser.**

2. **Bring the system into single-user mode.**

See the `telinit(1M)` man page for more information.

```
# /etc/telinit 1
```

3. Run the script to disable auditing.

Change to the `/etc/security` directory and execute the `bsmunconv` script there.

```
# cd /etc/security
# ./bsmunconv
```

4. Bring the system into multi-user mode.

```
# /etc/telinit 6
```

Note – The `bsmunconv` script removes the line in the `/etc/system` file which allows aborting the system with the Stop-A keyboard sequence. If you want to continue to disable the ability to abort the system with the Stop-A keyboard sequence after running the `bsmunconv` script, you must reenter into the `/etc/system` file the line that reads

```
# set abort_enable = 0
```

Audit Reference

This chapter describes the important components of auditing: the audit files, audit records and tokens, and utilities for auditing. The auditing mechanism lets an administrator detect potential security breaches.

Auditing can reveal suspicious or abnormal patterns of system usage and provides a means to trace suspect actions back to a particular user. Auditing can also serve as a deterrent: if users know that their activities are likely to be audited, they might be less likely to attempt malicious activities.

This chapter covers the following topics:

- “Audit Files” on page 381
- “Audit Programs” on page 389
- “Using `crontab` and `at job`” on page 391
- “Audit Record Structure” on page 393
- “Audit Token Structure” on page 393
- “Utilities Summary” on page 412

Audit Files

Auditing is controlled by the following configuration files:

- The `/etc/system` file
- The `audit_class`, `audit_control`, `audit_event`, and `audit_user` files and the `audit_warn` shell script in the `/etc/security` directory
- The `audit_startup` shell script

The /etc/system File

The `/etc/system` file contains commands which are read by the kernel during initialization and are used to customize the operation of the system. The file is modified by the `auditconv` and `auditunconv` shell scripts, which are used to activate and deactivate auditing. The `auditconv` adds the following lines to the `/etc/system` file:

```
set c2audit:audit_load=1
set abort_enable=0
```

The first command causes the `c2audit` loadable kernel module to be loaded when the system is booted. The second command disables the use of `Stop-A` which gives access to the debugger. The `auditunconv` command removes these lines, resulting in auditing being disabled when the system is rebooted.

The audit_class File

The `audit_class` file contains definitions of the existing audit classes. (Audit classes are groups of audit events.) Each class has an associated audit flag, the short name that stands for the class. The root user can define new audit classes, rename existing classes, or otherwise edit existing classes using `vi`, `ed`, or some other editor. See the `audit_class(4)` man page for details about modifying the `audit_class` file. The predefined classes in the default `audit_class` file are shown below:

TABLE 25-1 Audit Classes

Flag (short name)	Long name	Description
ad	administrative	Administrative actions.
all	all	Not specifically a class; sets all flags.
ap	application	Application-defined event
cl	file_close	Close system call.
ex	exec	Program execution.
fa	file_attr_acc	Access of object attributes: stat, pathconf, and so forth
fc	file_creation	Creation of object
fd	file_deletion	Deletion of object.
fm	file_attr_mod	Change of object attributes: chown, flock, and so forth.
fr	file_read	Read of data, open for reading, and so forth.

TABLE 25-1 Audit Classes (Continued)

Flag (short name)	Long name	Description
fw	file_write	Write of data, open for writing, and so forth.
io	ioctl	ioctl system call.
ip	ipc	System V IPC operations.
lo	login_logout	Login and logout events
na	non-attrib	Non-attributable events
no	no_class	Not specifically a class; is a null value for turning off event preselection.
nt	network	Network events: bind, connect, accept, and so forth.
ot	other	None of the above.
pc	process	Process operations: fork, exec, exit, and so forth.

To define a new class, use the editor of your choice to create a new class, such as the one in this example:

```
0x00008000:ts:test
```

Changes will take effect only for new user level audit record generation. The kernel events, which are loaded at boot time, will have to be reinitialized. Since there are now daemons that have been modified to generate audit records, they may or may not use the new values, depending on how they have been implemented. Most likely, the daemons will need to be restarted.

Note – The system should be rebooted if the audit class mappings are altered, to ensure that all daemons and the kernel are initialized with current class settings.

The audit_control File

The audit daemon reads the `audit_control` file on each machine (see the `audit_control(4)` man page).

The `audit_control` file is located in the `/etc/security` directory for each machine. A separate `audit_control` file is maintained for each machine in a distributed system because machines can mount their audit file systems from different locations or in a different order. For example, the primary audit file system for Machine A might be the secondary audit file system for Machine B.

A typical `audit_control` file might look like this:

```
dir:/var/audit
flags: lo,ad,-all
minfree:20
naflags:lo
```

The `audit_control` file contains four kinds of information:

- The *directory definition* lines (`dir:`) define which audit files and directories the machine will use to store its audit trail files. There can be one or more directory definition lines, and their order is significant because `auditd` opens audit files in these directories in the order specified (see the `audit(1M)` man page). The first audit directory specified is the primary audit directory for the machine, the second is the secondary audit directory (where the audit daemon puts audit trail files when the first one fills), and so forth. In the example above, the machine is being directed to store its audit trail files on the same machine, in the directory `/var/audit`.
- The *audit flags* line (`flags:`) contains the audit flags that define what classes of events are audited for all users on the machine. The audit flags specified here are referred to as the machine-wide audit flags or the machine-wide audit preselection mask. Audit flags are separated by commas, with no spaces. In the example above, auditing is done for logins and logouts, for administrative actions, and for all failed actions by any user.
- The *audit threshold* line (`minfree:`) defines the minimum free-space percentage level for all audit file systems. The `minfree` percentage must be greater than or equal to 0. The default is 20 percent. In the example above, the `minfree` value is 20 percent.
- The *nonattributable flags* line (`naflags:`) contains the audit flags that define what classes of events are audited when an action cannot be attributed to a specific user. The flags are separated by commas, with no spaces. In the example above, the only nonattributable flag is `lo`.

The administrator creates the `audit_control` file during the configuration process for each machine. Once created, the administrator can edit it. After a change, the administrator runs `audit -s` to instruct the audit daemon to reread the `audit_control` file.

Note – Note: The `audit -s` command does not change the preselection mask for existing processes. Use `auditconfig`, `setaudit` (see the `getuid(2)` man page) or `auditon` for existing processes.

Sample `audit_control` file

Following is a sample `audit_control` file for the machine `dopey`. `dopey` uses two audit file directories on the audit server `grumpy`, and a third audit directory mounted

from the second audit server `sleepy`, which is used only when the audit directories on `grumpy` fill up or become unavailable. The `minfree` value of 20 percent specifies that the warning script is run when the file systems are 80 percent filled; then the audit data for the current machine is stored in the next available audit directory, if any (see the `audit_warn(1M)` man page). The `flags` line specifies that all logins and administrative operations are to be audited (regardless of whether they succeed) and that failures of all types, except failures to create a file system object, are to be audited.

```
flags:lo,ad,-all,^-fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/grumpy/files
dir:/etc/security/audit/grumpy.1/files
#
# Audit filesystem used when grumpy fills up
#
dir:/var/audit/sleepy
```

The `audit_data` File

When `auditd` starts on each machine, it creates the file `/etc/security/audit_data`. This file consists of a single entry with two fields separated by a colon (see the `audit_data(4)` man page). The first field is the audit daemon's process ID, and the second field is the path name of the audit file to which the audit daemon is currently writing audit records. This file is not to be edited by users.

A typical `audit_data` file entry might look like this:

```
# cat /etc/security/audit_data
116:/etc/security/audit/grump.1/files/
20000522060002.not_terminated.sleepy
```

The `audit_event` File

The `audit_event` file, located in the `/etc/security` directory, contains the default event-to-class mappings (see the `audit_event(4)` man page). You can edit this file to change the class mappings, but if you do you must reboot the system or run `auditconfig -conf` to change the runtime kernel event-to-class mappings.

The `audit_startup` File

Auditing is enabled by starting the audit daemon (see the `auditd(1M)` man page). Normally this is done automatically, but you can manually start the audit daemon by

executing `/usr/sbin/audit` as root. In either case, the startup processes call the `audit_startup` shell script.

The `audit_startup` shell script resides in `/etc/security` and contains the commands to download the correct preselection masks for audit events into the kernel. It also sets up the initial audit policies. You can modify the file to perform some site-specific audit processing and set audit policies at system boot.

The `audit_user` File

You can set flags in the `audit_user` file to modify the defaults from the `audit_control` file for a particular user. For more information, see the `audit_user(4)` man page. The `audit_user` file has an entry for each user. This entry has three fields:

- username,
- the *always-audit* field, and
- the *never-audit* field.

Flags in the *always_audit* field specify the set of event classes that are to always be audited, and flags in the *never-audit* field specify the set of event classes that are to never be audited.

Auditing is processed in sequence, so auditing is enabled by flags in the *always -audit* field and disabled by flags in the *never-audit* field.

Using the *never-audit* field is not the same as removing classes from the *always-audit* set. Suppose you have a user named *fred* for whom you want to audit everything except successful reads of file system objects. This is a good way to audit almost everything for a user while generating only about 75% of the audit data that would be produced if all data reads were also audited.) You also want to apply the system defaults to this user. Here are two possible `audit_user` entries:

```
fred:all,^+fr:
```

This example says “always audit everything except successful file-reads.” In effect, this example applies any earlier default, as well as what is specified in this user’s `audit_user` entry.

```
fred:all:+fr
```

This example says “always audit everything, but never audit successful file reads.” However the *all* also overrides the system defaults.

The audit_warn File

The `audit_warn` script contains commands that tell the system what to do whenever the audit daemon encounters an unusual condition while writing audit records. See the `auditd(1M)` man page for more information. The administrator sets up the `audit_warn` script after enabling auditing.

By default, when invoked `audit_warn` writes a message to the console and sends a message to the `audit_warn` alias. However you can customize the script to warn of conditions requiring manual intervention or to handle certain situations automatically, for example.

The following exception conditions are supported by `audit_warn`:

Exception	What it does
<code>allhard [-count]</code>	Without <code>-count</code> , all of the audit partitions are completely filled. The audit daemon will remain in a loop, sleeping and checking for space, until some space becomes free. With <code>-count</code> , the hard limit for all audit partitions has been exceeded <code>-count</code> times. Do not set <code>count=1</code> or you may saturate the file system containing the mail spool directory.
<code>allsoft</code>	All of the audit partitions have filled to the soft limit. The audit daemon will now completely fill each of the audit partitions.
<code>auditoff</code>	Some entity other than the audit daemon has changed the system audit state. The audit daemon will exit.
<code>ebusy</code>	The audit daemon is already running and someone has attempted to start another daemon executing. The Solaris 8 auditing system only allows one audit daemon to be started.
<code>getacdir</code>	Could not obtain the location of the audit partitions. Audit daemon will exit.
<code>hard</code>	One of the audit partitions has completely filled.
<code>nostart</code>	Auditing could not be started.
<code>postsigterm</code>	The audit daemon received a signal during shutdown. Some audit records may be lost.
<code>postsigterm</code>	The audit daemon received a signal during shutdown. Some audit records may be lost.
<code>soft</code>	One audit file system has filled to the soft limit (<code>minfree()</code> percentage specified in <code>audit_control</code>) and the audit daemon is switching to the next audit partition.
<code>tmpfile</code>	The audit daemon could not open temporary files and will exit.

The device_maps File

The `device_maps` file defines device-special file mappings for each device, which in many cases is not intuitive. For more information see the `device_maps(4)` man page. It allows various programs to discover which device-special files go with which devices.

A rudimentary `device_maps` file is created by the `auditconv` command when auditing is enabled. However the system administrator updates the file when setting up device allocation, and is expected to augment and customize the file for the site.

In this file each device is represented by a one-line entry of the form:

device-name:device-type:device-list

Lines in the file can end with a `\` to continue an entry on the next line. You can also include comments by preceding text with a `#` and ending it with a newline that is not preceded by a `\`. Leading and trailing blanks are also allowed.

The terms are described in the table below:

Term	What it means
device-name	The device name, such as <code>st0</code> , <code>fd0</code> , or <code>audio</code> . This name must correspond to the name of the lockfile used in the <code>/etc/security/dev</code> directory.
device-type	The generic device type, such as <code>ast</code> , <code>fd</code> , or <code>audio</code> . The device-type logically groups related devices.
device-list	The list of device-special files associated with the physical device. This list must contain all of the special files that allow access to a particular device. If the list is incomplete, a malevolent user could still obtain or modify private information.

You can use the `dminfo` command to read the `device_maps` file and determine the device name, device type, and device-special files when setting up an allocatable device.

In the example below, either the real device files located under `/devices` or the symbolic links in `/dev` (provided for binary compatibility) are valid entries in the device-list field. The screen below shows an example of entries for SCSI tape `st0` and diskette `fd0` in a `device_maps` file.

```
fd0:\  
fd:\
```

```
/dev/fd0 /dev/fd0a /dev/fd0b /dev/rfd0 /dev/rfd0a /dev/rfd0b:\  
.  
st0:\  
st:\  
/dev/rst0 /dev/rst8 /dev/rst16 /dev/nrst0 /dev/nrst8 /dev/nrst16:\
```

Audit Programs

The following section describes the programs that are associated with auditing.

The audit Program

The `audit` program and associated `audit` command control the actions of the audit daemon. The `audit` program lets you switch audit files, reset the daemon, activate or deactivate auditing, and adjust the preselection mask of auditing on a single machine. See the `audit(1M)` man page for a discussion of the available options.

The `auditd` Daemon

`auditd` is the audit daemon. It performs the following functions:

- It invokes a special system call, `auditsvc(2)`, by which audit records are written directly to the audit trail instead of being passed through `auditd` in a series of read/write operations. Except when handling errors, all the data it manipulates is in kernel memory and not subject to swapping or paging.

It opens and closes the audit log files in the directories specified in the `audit_control` file, in the order specified.

It reads audit data from the kernel and writes it to an audit file.

It executes the `audit_warn` script when the audit directories fill to limits specified in the `audit_control` file. The script sends warnings as directed, by default to the console and to the `audit_warn` alias.

The `auditconfig` Program

The `auditconfig` program provides a command line interface to get and set audit configuration parameters. See the `auditconfig(1M)` man pages for detailed explanations about the options available with `auditconfig`. You can also use `auditconfig` with the `-setpolicy` flag to change the default audit policies. The `-lspolicy` argument shows the audit policies you can change.

The `auditreduce` Program

Audit analysis is performed with the aid of the `auditreduce` program. `auditreduce` does not actually display audit data, but assembles audit files together into one file, sorted by time stamp, and performs some post-selection tasks. Options available with the `auditreduce` command let you select audit records by time, type of record, selection class, originating user, etc. See the `auditreduce(1M)` man page for a list of selection criteria that can be used to choose audit records from the audit trail.

When multiple machines are running BSM and are administered as part of a distributed system, each machine performs auditable events and each machine writes the resulting audit records to its own machine-specific audit file. However if users are allowed to work on multiple machines, you would have to inspect each machine's records to find out about the activities of a particular user.

The `auditreduce` program makes the job of maintaining the system-wide audit trail practical. It (or a shell script you write to provide a higher-level interface) lets you read the logical combination of all audit files in the system as a single audit trail, without regard to where the records were generated or where they are stored. It operates on the audit records produced by the audit daemon. Records from one or more audit files are selected and merged into a single, chronologically ordered output file. It then selects messages from the input files as requested, as the records are read, before the files are merged and written to disk.

Without options, `auditreduce` merges the entire audit trail (consisting of all the audit files in all of the subdirectories in the audit root directory `/etc/security/audit`) and sends these records to standard output. You can make the records human-readable with the `praudit` command.

You can direct `auditreduce` to treat only certain files by specifying them as command arguments:

```
# auditreduce /var/auditbongos/files/1993*.1993*.bongos
```

The praudit Program

The praudit program displays audit records in a readable format. It has several options to control the output format and the degree of binary information displayed in audit records (for example, user IDs can be translated into names).

auditreduce and praudit are designed to be used together. The output of auditreduce either is sent to a file, which is then viewed with praudit, or is piped directly to praudit.

```
# auditreduce | praudit | more
```

The praudit program inserts control characters, such as ^J for newline, when printing text tokens which can make the outlook look strange. For instance you may see a line like this one:

```
# text,Enter your name on the next line^JName:
```

See the praudit(1M) man page for more details.

Using crontab and atjob

Audit characteristics for a user are passed down to all descendants of the login process. However in the case of at(1) and crontab(1) jobs, the process performing the action on behalf of the requestor is not in any way related to the user. With Solaris 8 release, the cron utilities pass the user's audit characteristics in an out-of-band manner. Associated with each at job entry and with the individual's crontab entry is another file containing the audit characteristics that are used by cron when servicing the request.

The cron daemon recognizes the ancillary file with each atjob entry and each crontab entry. This file is read when the cron daemon processes the request, and the audit information is saved for future use. The internal queue within cron includes the audit characteristics from the ancillary file. For example:

```
root    is the crontab for root in /usr/spool/cron/crontabs/
```

```
root.au is the ancillary audit data file (also in /usr/spool/cron/crontabs/)
```

When the cron entry is executed, the audit information is used to set the user's audit characteristics of the agent process. Thus when the actual service is performed, the audit characteristics will be those of the requestor. Since the preselection mask may have been modified for the user between the time the request is made and the time it

is executed, the mask is adjusted by combining the user's preselection mask at the time the service is performed with the one saved when the request was made.

The `atjob` occurs only once and the `cron` daemon removes the request file when it is serviced. It likewise removes the ancillary audit file at that time. In the case of a `crontab` service, the ancillary file is left unchanged since the services are performed periodically.

The ancillary audit file contains the user audit characteristics from the requestor. The data is in ASCII format and contains the following:

audit ID	recorded as an unsigned integer.
preselection mask	recorded as two unsigned integers: audit on success and audit on failure.
terminal ID	recorded as two unsigned integers: one for port and one for the host name.
session ID	recorded as an unsigned integer.

The `at` command lets the user perform a command at a later time. This command is contained within a control file located in `/usr/lib/spool/cron/atjobs`. The information within the file contains both the command to be executed and the environment at the time of the request. An ancillary file containing the audit characteristics of the user making the request is created with the same prefix as the `atjob` file but with an `.au` postfix. For example:

<code>836327447.a</code>	is the <code>atjob</code> request.
<code>836827447.a.au</code>	is the ancillary audit data file

The `atrm` command is used to remove a pending `at` request. It removes the ancillary audit file when it removes the `atjob` file.

When the user edits a `crontab` file, the ancillary audit file is updated. The old audit values are replaced by the audit values of the person editing the `crontab` file. When the `crontab` entry is read by `cron`, the new audit characteristics take effect. Note that all periodic services contained with the `crontab` are affected.

Audit Record Structure

Each auditable event in the system generates a particular type of audit record, containing tokens which describe the event. “Audit Token Structure” on page 393 gives a detailed description of each audit token.

Each audit record begins with a header token, contains a subject token, and optionally ends with a trailer token. Within the record may be other tokens describing the event. For user-level and kernel events, these tokens describe the process that performed the event, the objects on which it was performed, and the objects’ tokens, such as the owner or mode.

The audit record’s event type is contained in the header token, and it gives specific meaning to the content and position of the other tokens in the record.

An audit record does not describe the audit event class to which the event belongs; that mapping is determined by a separate table, stored in `/etc/security/audit_event`.

Audit records are stored and manipulated in binary form. However, the byte order and size of data is predetermined to simplify compatibility between different machines.

In order to read the audit records, they must be processed by the `praudit` command. By default, `praudit` displays each token’s type, followed by a comma, followed by the data from the token. The token type is displayed by default as a name, such as `header`, or in `-r` format as a decimal number.

Audit Token Structure

Logically, each token has a token type identifier followed by data specific to the token. Each token type has its own format and structure. The current tokens are shown in Table 25–2. The token scheme can be extended.

TABLE 25–2 Basic Security Module Audit Tokens

Token Name	Description
“acl token” on page 395	Access Control List information
“arbitrary Token” on page 395	Data with format and type information

TABLE 25-2 Basic Security Module Audit Tokens (Continued)

Token Name	Description
"arg Token" on page 396	System call argument value
"attr Token" on page 397	Vnode tokens
"exec_args Token" on page 397	Exec system call arguments
"exec_env Token" on page 398	Exec system call environment variables
"exit Token" on page 399	Program exit information
"file Token" on page 399	Audit file information
"groups Token (Obsolete)" on page 400	Process groups information (obsolete)
"header Token" on page 401	Indicates start of record
"in_addr Token" on page 401	Internet address
"ip Token" on page 402	IP header information
"ipc Token" on page 402	System V IPC information
"ipc_perm Token" on page 403	System V IPC object tokens
"ipport Token" on page 404	Internet port address
"newgroups Token" on page 404	Process groups information
"opaque Token" on page 405	Unstructured data (unspecified format)
"path Token" on page 405	Path information (path)
"process Token" on page 406	Process token information
"return Token" on page 407	Status of system call
"seq Token" on page 408	Sequence number token
"socket Token" on page 408	Socket type and addresses
"subject Token" on page 409	Subject token information (same structure as process token)
"text Token" on page 410	ASCII string
"trailer Token" on page 411	Indicates end of record

An audit record always contains a `header` token. The `header` token indicates where the audit record begins in the audit trail. Every audit record contains a `subject` token, except for audit records from some nonattributable events. In the case of attributable events, these two tokens refer to the values of the process that caused the event. In the case of asynchronous events, the `process` tokens refer to the system.

acl token

The `acl` token records information about ACLs. It consists of four fixed fields. The fixed fields are:

- a token ID that identifies this token as an `acl` token,
- a field that specifies the ACL type,
- an ACL ID field,
- a field that lists the permissions associated with this ACL.

The `praudit` command displays the `acl` token as follows:

```
acl,tpanero,staff,0755
```

The `acl` token appears as follows:

Token ID	ACL type	ACL ID	ACL permissions
----------	----------	--------	-----------------

FIGURE 25-1 `acl` Token Format

arbitrary Token

The `arbitrary` token encapsulates data for the audit trail. It consists of four fixed fields and an array of data. The fixed fields are:

- a token ID that identifies this token as an `arbitrary` token,
- a suggested format field (for example, hexadecimal),
- a size field that specifies the size of data encapsulated (for example, short), and
- a count field that gives the number of following items.

The remainder of the token is composed of one or more items of the specified type. The `praudit` command displays the `arbitrary` token as follows:

```
arbitrary,decimal,int,1  
42
```

The `arbitrary` token appears as follows:

Token ID	Print format	Item size	Number items	Item 1	0 0 0	Item <i>n</i>
----------	--------------	-----------	--------------	--------	-------	---------------

FIGURE 25-2 `arbitrary` Token Format

The print format field can take the values shown in Table 25-3.

TABLE 25-3 arbitrary Token Print Format Field Values

Value	Action
AUP_BINARY	Print date in binary
AUP_OCTAL	Print date in octal
AUP_DECIMAL	Print date in decimal
AUP_HEX	Print date in hex
AUP_STRING	Print date as a string

The item size field can take the values shown in Table 25-4.

TABLE 25-4 arbitrary Token Item Size Field Values

Value	Action
AUR_BYTE	Data is in units of bytes (1 byte)
AUR_SHORT	Data is in units of shorts (2 bytes)
AUR_LONG	Data is in units of longs (4 bytes)

arg Token

The `arg` token contains system call argument information: the argument number of the system call, the argument value, and an optional descriptive text string. This token allows a 32-bit integer system-call argument in an audit record. The `arg` token has 5 fields:

- a token ID that identifies this token as an `arg` token,
- an argument ID that tells which system call argument the token refers to,
- the argument value,
- the length of a descriptive text string, and
- the text string.

The `praudit` command displays the `arg` token as follows:

```
argument,1,0x00000000,addr
```

Figure 25-3 shows the token form.

Token ID	Argument #	Argument value	Text length	Text
----------	------------	----------------	-------------	------

FIGURE 25-3 `arg` Token Format

`attr` Token

The `attr` token contains information from the file `vnode`. This token has 7 fields:

- a token ID that identifies this as an `attr` token,
- the file access mode and type,
- the owner user ID,
- the owner group ID,
- the file system ID,
- the inode ID, and
- the device ID the file might represent.

See the `statvfs(2)` man page for further information about the file system ID and the device ID.

This token usually accompanies a `path` token and is produced during path searches. In the event of a path-search error, this token is not included as part of the audit record since there is no `vnode` available to obtain the necessary file information. An `attr` token is displayed by `praudit` as follows:

```
attribute,100555,root,staff,1805,13871,-4288
```

Figure 25-4 shows the `attr` token format.

Token ID	File mode	Owner UID	Owner GID	File system ID	File inode ID	Device ID
----------	-----------	-----------	-----------	----------------	---------------	-----------

FIGURE 25-4 `attr` Token Format

`exec_args` Token

The `exec_args` token records the arguments to an `exec` system call. The `exec_args` record has two fixed fields:

- a token ID field that identifies this as an `exec_args` token
- a count that represents the number of arguments passed to the `exec` call

The remainder of the token is composed of zero or more null-terminated strings. An `exec_args` token is displayed by `praudit` as follows:

```
vi,/etc/security/audit_user
```

Figure 25-5 shows an `exec_args` token.

Token ID	Count	env_args
----------	-------	----------

FIGURE 25-5 `exec_args` Token Format

Note – The `exec_args` token is output only when the audit policy `argv` is active.

`exec_env` Token

The `exec_env` token records the current environment variables to an `exec` system call. The `exec_env` record has two fixed fields:

- a token ID field that identifies this as an `exec_env` token
- a count that represents the number of arguments passed to the `exec` call.

The remainder of the token is composed of zero or more null-terminated strings. An `exec_env` token is displayed by `praudit` as follows:

```
exec_env,25,  
GROUP=staff,HOME=/export/home/matrix,HOST=mestrix,HOSTTYPE=sun4,HZ=100,  
LC_COLLATE=en_US.ISO8859-1,LC_CTYPE=en_US.ISO8859-1,LC_MESSAGES=C,  
LC_MONETARY=en_US.ISO8859-1,LC_NUMERIC=en_US.ISO8859-1,  
LC_TIME=en_US.ISO8859-1,LOGNAME=matrix,MACHTYPE=sparc,  
MAIL=/var/mail/matrix,OSTYPE=solaris,PATH=/usr/sbin:/usr/bin,PS1=#,  
PWD=/var/audit,REMOTEHOST=209.198.087.208,SHELL=/usr/bin/csh,SHLV=1,  
TERM=dtterm,TZ=US/Pacific,USER=matrix,VENDOR=sun
```

Figure 25-6 shows an `exec_env` token.

Token ID	Count	env_args
----------	-------	----------

FIGURE 25-6 `exec_env` Token Format

Note – The `exec_env` token is output only when the audit policy `arg0` is active.

exit Token

The `exit` token records the exit status of a program. The `exit` token contains:

- a token ID,
- a program exit status as passed to the `exit()` system call, and
- a return value that describes the exit status or indicates a system error number.

The `praudit` command displays the `exit` token as follows:

```
exit,Error 0,0
```

Figure 25-7 shows an `exit` token.

Token ID	Status	Return value
----------	--------	--------------

FIGURE 25-7 `exit` Token Format

file Token

The `file` token is a special token generated by the audit daemon to mark the beginning of a new audit trail file and the end of an old file as it is deactivated. The audit daemon builds a special audit record containing this token to “link” together successive audit files into one audit trail. The `file` token has four fields:

- a token ID that identifies this token as a `file` token,
- a time and date stamp that identifies the time the file was created or closed,
- a byte count of the file name including a null terminator, and
- a field holding the file null-terminated name.

The `praudit` command displays the `file` token as follows:

```
file,Tue Sep  1 13:32:42 1992, + 79249 msec,  
  /baudit/localhost/files/19920901202558.19920901203241.quisp
```

Figure 25-8 shows a file token.

Token ID	Date & time	Name length	Previous/next file name
----------	-------------	-------------	-------------------------

FIGURE 25-8 file Token Format

groups Token (Obsolete)

This token has been replaced by the `newgroups` token, which provides the same type of information but requires less space. A description of the `groups` token is provided here for completeness, but the application designer should use the `newgroups` token. Notice that `praudit` does not distinguish between the two tokens, as both token IDs are labelled `groups` when ASCII style output is displayed.

The `groups` token records the `groups` entries from the process's credential. The `groups` token has two fixed fields:

- Token ID
- Array of groups entries of size `NGROUPS_MAX` (16)

The remainder of the token consists of zero or more group entries. The `praudit` command displays the `file` token as follows:

```
group, staff, admin, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
```

Figure 25-9 shows a `groups` token.

Token ID	Groups
----------	--------

FIGURE 25-9 groups Token Format

Note – The `groups` token is output only when the audit policy group is active.

The `praudit` command displays the `group` token as follows:

```
group,staff,wheel,daemon,kmem,bin,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
```


header Token

The header token is special in that it marks the beginning of an audit record and combines with the trailer token to bracket all the other tokens in the record. The header token has six fields:

- a token ID field that identifies this as a header token,
- a byte count of the total length of the audit record, including both header and trailer,
- a version number that identifies the version of the audit record structure,
- the audit event ID that identifies the type of audit event the record represents,
- and the time and date the record was created.

On 64-bit systems, the header token will be displayed with a 64-bit time stamp, in place of the 32-bit time stamp.

When displayed by `praudit` in default format, a header token looks like the following example from `ioctl`:

```
header,240,1,ioctl(2),es,Tue Sept 1 16:11:44 1992, + 270000 msec
```

Figure 25–10 shows a header token.

Token ID	Byte count	Version #	Event ID	ID modifier	Date and time
----------	------------	-----------	----------	-------------	---------------

FIGURE 25–10 header Token Format

The event modifier field has the following flags defined:

0x4000	PAD_NOTATTR	nonattributable event
0x8000	PAD_FAILURE	fail audit event

in_addr Token

The `in_addr` token contains an Internet address. This 4-byte value is an Internet Protocol address. The token has two fields:

- a token ID that identifies this token as an ip address token, and
- an Internet address.

The `praudit` command displays the ip address token as follows:

```
ip address,129.150.113.7
```

For the Solaris 8 release, the Internet Address can be displayed as a IPv4 address using 4 bytes, or as an IPv6 address using 16 bytes to describe the type, and 16 bytes to describe the address. Figure 25–11 shows an `in_addr` token.

Token ID	Internet address
----------	------------------

FIGURE 25–11 `in_addr` Token Format

`ip` Token

The `ip` token contains a copy of an Internet Protocol header but does not include any IP options. The IP options can be added by including more of the IP header in the token. The token has two fields:

- a token ID that identifies this as an `ip` token, and
- a copy of the IP header (all 20 bytes).

The `praudit` command displays the `ip` token as follows:

```
ip address,0.0.0.0
```

The IP header structure is defined in `/usr/include/netinet/ip.h`. Figure 25–12 shows an `ip` token.

Token ID	IP header
----------	-----------

FIGURE 25–12 `ip` Token Format

`ipc` Token

The `ipc` token contains the System V IPC message/semaphore/shared-memory handle used by the caller to identify a particular IPC object. This token has three fields:

- a token ID that identifies this as an `IPC` token,
- a type field that specifies the type of the IPC object, and
- the handle that identifies the IPC object.

The `praudit` command displays the `IPC` token as follows:

```
IPC,msg,3
```

Note – The IPC object identifiers violate the context-free nature of the Solaris CMW audit tokens. No global “name” uniquely identifies IPC objects; instead, they are identified by their handles, which are valid only during the time the IPC objects are active. The identification should not be a problem since the System V IPC mechanisms are seldom used and they all share the same audit class.

The IPC object type field can have the values shown in Table 25–5. The values are defined in `/usr/include/bsm/audit.h`.

TABLE 25–5 IPC Object Type Field

Name	Value	Description
AU_IPC_MSG	1	IPC message object
AU_IPC_SEM	2	IPC semaphore object
AU_IPC_SHM	3	IPC shared memory object

Figure 25–13 shows an `ipc` token.

Token ID	IPC object type	IPC object ID
----------	-----------------	---------------

FIGURE 25–13 `ipc` Token Format

`ipc_perm` Token

The `ipc_perm` token contains a copy of the System V IPC access information. This token is added to audit records generated by shared memory, semaphore, and message IPC events. The token has eight fields:

- a token ID that identifies this token as an `IPC_perm` token,
- the user ID of the IPC owner,
- the group ID of the IPC owner,
- the user ID of the IPC creator,
- the group ID of the IPC creator,
- the access modes of the IPC,
- the sequence number of the IPC, and
- the IPC key value.

The `praudit` command displays the `IPC_perm` token as follows:

```
IPC perm,root,wheel,root,wheel,0,0,0x00000000
```

The values are taken from the `ipc_perm` structure associated with the IPC object. Figure 25-14 shows an `ipc_perm` token format.

Token ID	Owner uid	Owner gid	Creator uid	Creator gid	IPC mode	Sequence ID	IPC key
----------	-----------	-----------	-------------	-------------	----------	-------------	---------

FIGURE 25-14 `ipc_perm` Token Format

`ipport` Token

The `ipport` token contains the TCP (or UDP) port address. The token has two fields:

- a token ID that identifies this as an `ip port` token, and
- the TCP/UDP port address.

The `praudit` command displays the `ip port` token as follows:

```
ip port,0xf6d6
```

Figure 25-15 shows an `ipport` token.

Token ID	Port ID
----------	---------

FIGURE 25-15 `ipport` Token Format

`newgroups` Token

This token is the replacement for the `groups` token. Notice that `praudit` does not distinguish between the two tokens, as both token IDs are labelled `groups` when ASCII output is displayed.

The `newgroups` token records the `groups` entries from the process's credential. The `newgroups` token has two fixed fields:

- a token ID field that identifies this as a `newgroups` token, and
- a count that represents the number of groups contained in this audit record.

The remainder of the token is composed of zero or more group entries. The `praudit` command displays the `ip port` token as follows:

```
group, staff, admin
```

Figure 25-16 shows a `newgroups` token.

Token ID	Count	Groups
----------	-------	--------

FIGURE 25-16 newgroups Token Format

Note – The `newgroups` token is output only when the audit policy group is active.

opaque Token

The `opaque` token contains unformatted data as a sequence of bytes. The token has three fields:

- a token ID that identifies this as an `opaque` token,
- a byte count of the amount of data, and
- an array of byte data.

The `praudit` command displays the `opaque` token as follows:

```
opaque, 12, 0x4f5041515545204441544100
```

Figure 25-17 shows an `opaque` token.

Token ID	Data length	Data bytes
----------	-------------	------------

FIGURE 25-17 opaque Token Format

path Token

The `path` token contains access path information for an object. The token contains the following fields:

- a token ID,
- a byte count of the path length (does not show), and
- the absolute path to the object based on the real root of the system.

The `praudit` command displays the `path` token as follows:

```
path, /etc/security/audit_user
```

Figure 25-18 shows a `path` token.

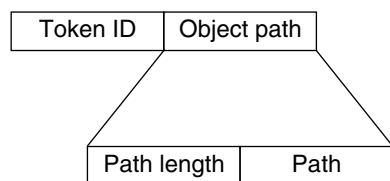


FIGURE 25-18 path Token Format

process Token

The `process` token contains information describing a user associated with a process, such as the recipient of a signal. The token has 9 fields:

- a token ID that identifies this token as a `process` token,
- the invariant audit ID,
- the effective user ID,
- the effective group ID,
- the real user ID,
- the real group ID,
- the process ID,
- the audit session ID, and
- a terminal ID consisting of a device ID and a machine ID.

The `praudit` command displays the `process` token as follows:

```
process,root,root,wheel,root,wheel,0,0,0,0.0.0.0
```

Figure 25-19 shows a `process` token.

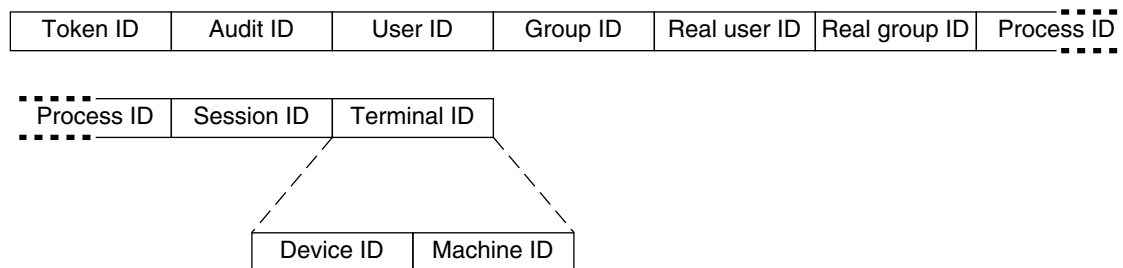


FIGURE 25-19 process Token Format

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note – The `process` token fields for the session ID, the real user ID, or the real group ID might be unavailable. The entry is then set to -1.

Any token containing a terminal ID has several variations. `praudit` hides these variations on output of the terminal ID so that they all appear to be the same. This field is handled the same way for any token that contains it. The terminal ID is either an IP address and port number or a device ID, such as the serial port connected to a modem, in which case it is zero. The terminal ID is in one of several formats:

For device numbers:

- 32 bit applications: 4 byte device number, 4 bytes unused
- 64 bit applications: 8 byte device number, 4 bytes unused

For port numbers in the Solaris 7 or earlier releases:

- 32 bit applications: 4 byte port number, 4 byte IP address
- 64 bit applications: 8 byte port number, 4 byte IP address

For port numbers in the Solaris 8 or 9 releases:

- 32 bit with IPV4: 4 byte port number, 4 byte IP type, 4 byte IP address
- 32 bit with IPV6: 4 byte port number, 4 byte IP type, 16 byte IP address
- 64 bit with IPV4: 8 byte port number, 4 byte IP type, 4 byte IP address
- 64 bit with IPV6: 8 byte port number, 4 byte IP type, 16 byte IP address

return Token

The `return` token contains the return status of the system call (`u_error`) and the process return value (`u_rval1`). The token has three fields:

- a token ID that identifies this token as a `return` token,
- the error status of the system call, and
- the system call return value.

This token is always returned as part of kernel-generated audit records for system calls. The token indicates exit status and other return values in application auditing.

The `praudit` command displays the `return` token as follows:

```
return, success, 0
```

Figure 25–20 shows a `return` token.

Token ID	Process error	Process value
----------	---------------	---------------

FIGURE 25-20 return Token Format

seq Token

The `seq` token (sequence token) is an optional token that contains an increasing sequence number. This token is for debugging. The token is added to each audit record when the `AUDIT_SEQ` policy is active. The `seq` token has 2 fields:

- a token ID that identifies this token as a `seq` token, and
- a 32-bit unsigned long field that contains the sequence number.

The sequence number is incremented every time an audit record is generated and put onto the audit trail. The `praudit` command displays the `seq` token as follows:

```
sequence,1292
```

Figure 25-21 shows a `seq` token.

Token ID	Sequence number
----------	-----------------

FIGURE 25-21 seq Token Format

socket Token

The `socket` token contains information describing an Internet socket. The `socket` token has 6 fields:

- a token ID that identifies this token as a `socket` token,
- a socket type field that indicates the type of socket referenced (TCP/UDP/UNIX),
- the local port address,
- the local Internet address,
- the remote port address, and
- the remote Internet address.

The `praudit` command displays the `socket` token as follows:

```
socket,0x0000,0x0000,0.0.0.0,0x0000,0.0.0.0
```


For the Solaris 8 release, the Internet Address can be displayed as a IPv4 address using 4 bytes, or as an IPv6 address using 16 bytes to describe the type, and 16 bytes to describe the addresses. Figure 25–22 shows a socket token.

Token ID	Type	Remote port	Remote Internet address
----------	------	-------------	-------------------------

FIGURE 25–22 socket Token Format

subject Token

The subject token describes a user that performs or attempt to perform an operation. The structure is the same as the process token. The token has 9 fields:

- an ID that identifies this as a subject token,
- the invariant audit ID,
- the effective user ID,
- the effective group ID,
- the real user ID,
- the real group ID,
- the process ID,
- the audit session ID, and
- a terminal ID consisting of a device ID and a machine ID.

This token is always returned as part of kernel-generated audit records for system calls. The `praudit` command displays the subject token as follows:

```
subject,cjc,cjc,staff,cjc,staff,424,223,0 0 quisp
```

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note – The subject token fields for the session ID, the real user ID, or the real group ID might be unavailable. The entry is then set to -1.

Any token containing a terminal ID has several variations. `praudit` hides these variations on output of the terminal ID so that they all appears to be the same. This field is handled the same way for any token that contains it. The terminal ID is either an IP address and port number or a device ID, such as the serial port connected to a modem, in which case it is zero. The terminal ID is in one of several formats:

For device numbers:

- 32 bit applications: 4 byte device number, 4 bytes unused
- 64 bit applications: 8 byte device number, 4 bytes unused

For port numbers in the Solaris 7 or earlier releases::

- 32 bit applications: 4 byte port number, 4 byte IP address
- 64 bit applications: 8 byte port number, 4 byte IP address

For port numbers in the Solaris 8 or 9 releases:

- 32 bit with IPV4: 4 byte port number, 4 byte IP type, 4 byte IP address
- 32 bit with IPV6: 4 byte port number, 4 byte IP type, 16 byte IP address
- 64 bit with IPV4: 8 byte port number, 4 byte IP type, 4 byte IP address
- 64 bit with IPV6: 8 byte port number, 4 byte IP type, 16 byte IP address

Figure 25–23 shows the token.

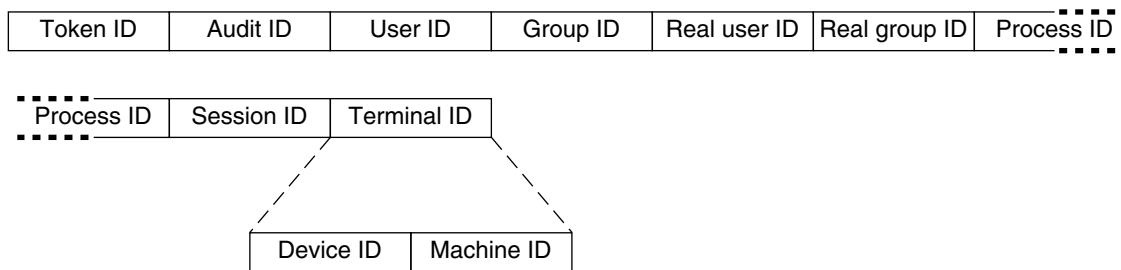


FIGURE 25–23 subject Token Format

text Token

The `text` token contains a text string. The token has three fields:

- a token ID that identifies this token as a `text` token,
- the length of the text string, and
- the text string itself.

The `prauditcommand` displays the `text` token as follows:

```
text,aw_test_token
```

Figure 25–24 shows a text token.

Token ID	Text length	Text string
----------	-------------	-------------

FIGURE 25–24 text Token Format

trailer Token

The two tokens, `header` and `trailer`, are special in that they distinguish the endpoints of an audit record and bracket all the other tokens. A `header` token begins an audit record. A `trailer` token ends an audit record. It is an optional token that is added as the last token of each record only when the `AUDIT_TRAIL` audit policy has been set.

The `trailer` token is special in that it marks the termination of an audit record. Together with the `header` token, the `trailer` token delimits an audit record. The `trailer` token supports backward seeks of the audit trail. The `trailer` token has three fields: a token ID that identifies this token as a `trailer` token, a pad number to aid in marking the end of the record, and the total number of characters in the audit record, including both the `header` and `trailer` tokens.

The `praudit` command displays the `trailer` token as follows:

```
trailer,136
```

Figure 25–25 shows a `trailer` token.

Token ID	Pad number	Byte count
----------	------------	------------

FIGURE 25–25 trailer Token Format

The audit trail analysis software ensures that each record contains both `header` and `trailer`. In the case of a write error, as when a file system becomes full, an audit record can be incomplete and truncated. `auditsvc`, the system call responsible for writing data to the audit trail, attempts to put out complete audit records. See the `auditsvc(2)` man page. When file system space runs out, the call terminates without releasing the current audit record. When the call resumes, it can then repeat the truncated record.

Utilities Summary

BSM brings a number of additional utilities to the Solaris Operating Environment. The utilities are listed here in four sections, each of which has a table below. Each table gives utility names and a short description of the task performed by each utility. The sections are identified by the man page suffix.

TABLE 25-6 Maintenance Commands

Command	Task
allocate(1M)	Allocate a device
audit(1M)	Control the audit daemon
audit_startup(1M)	Initialize the audit subsystem
audit_warn(1M)	Run the audit daemon warning script
auditconfig(1M)	Configure auditing
auditd(1M)	Control audit trail files
auditreduce(1M)	Merge and select audit records from audit trail files
auditstat(1M)	Display kernel audit statistics
bsmconv(1M)	Enable a Solaris system to use the Basic Security Module
bsmunconv(1M)	Disable the Basic Security Module and return to the Solaris operating environment (see the <code>bsmconv(1M)</code> man page)
deallocate(1M)	Deallocate a device
dminfo(1M)	Report information about a device entry in a device maps file
list_devices(1M)	List allocatable devices
praudit(1M)	Print contents of an audit trail file

TABLE 25-7 System Calls

System Call	Task
audit(2)	Write a record to the audit log
auditon(2)	Manipulate auditing
auditsvc(2)	Write audit log to specified file descriptor
getaudit(2)	Get process audit information

TABLE 25-7 System Calls *(Continued)*

System Call	Task
getaudit(2)	Get user audit identity
setaudit(2)	Get process audit information (see getaudit(2))
setaudit(2)	Get user audit identity (see getaudit(2))

TABLE 25-8 Library Functions

Library Call	Task
au_open(3BSM), au_close(3), au_write(3)	Construct and write audit records
au_preselect(3BSM)	Preselect an audit event
au_to_arg(3), au_to_attr(3), au_to_data(3), au_to_groups(3), au_to_in_addr(3), au_to_ipc(3), au_to_ipc_perm(3), au_to_ipport(3), au_to_me(3), au_to_opaque(3), au_to_path(3), au_to_process(3), au_to_return(3), au_to_socket(3), au_to_text(3) (see au_to(3BSM) for all of these functions)	Create audit record tokens
au_user_mask(3BSM)	Get user's binary preselection mask
getacinfo(3BSM), getacdir(3BSM), getacflg(3BSM), getacmin(3BSM), , getacinfo(3BSM), setac(3BSM), endac(3BSM)	Get audit control file information
getauclassent(3BSM), getauclassnam(3BSM), endauclass(3BSM), getauclassnam_r(3BSM), getauclassent_r(3BSM)	Get audit_class entry
getauditflags(3BSM), getauditflagsbin(3BSM), getauditflagschart(3BSM)	Convert audit flag specifications
getauevent(3BSM), getauevnam(3BSM), getauevnum(3BSM), getauevnonam(3BSM), setauevent(3BSM), endauevent(3BSM), getauevent_r(3BSM), getauevnam_r(3BSM), getauevnum_r(3BSM)	Get audit_user entry
getauusername(3BSM), getauuserent(3BSM), setauuser(3BSM), endauuser(3BSM)	Get audit_user entry
getfauditflags(3BSM)	Generate the process audit state

TABLE 25-9 Headers, Tables, and Macros

Files	Task
audit.log(4)	Gives format for an audit trail file
audit_class(4)	Gives audit class definitions
audit_control(4)	Controls information for system audit daemon
audit_data(4)	Holds current information on the audit daemon
audit_event(4)	Holds audit event definition and class mapping
audit_user(4)	Holds per-user auditing data file
device_allocate(4)	Contains physical device information
device_maps(4)	Contains physical device information

Glossary

admin principal	A user principal with a name of the form <i>username/admin</i> (as in <i>joe/admin</i>). An admin principal can have more privileges (for example, to change policies) than a regular user principal. See also principal name, user principal.
application server	See network application server.
authentication	The process of verifying the claimed identity of a principal.
authenticator	Authenticators are passed by clients when requesting tickets (from a KDC) and services (from a server). They contain information, generated using a session key known only by the client and server, that can be shown to be of recent origin, thus indicating the transaction is secure. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user's host, and a timestamp. Unlike a ticket, an authenticator can be used only once, usually when access to a service is requested. An authenticator is encrypted using the session key for that client and that server.
authorization	<p>The process of determining whether a principal can use a service, which objects the principal is allowed to access, and the type of access allowed for each.</p> <ol style="list-style-type: none">1. The process of determining whether a principal can use a service, which objects the principal is allowed to access, and the type of access allowed for each.2. In RBAC, a permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions otherwise prohibited by security policy.
client	<ul style="list-style-type: none">■ Narrowly, a process that makes use of a network service on behalf of a user; for example, an application that uses <code>rlogin</code>. In some

cases, a server can itself be a client of some other server or service.

- More broadly, a host that a) receives a Kerberos credential and b) makes use of a service provided by a server.

Informally, a principal that makes use of a service.

client principal	(RPCSEC_GSS API) A client (a user or an application) that uses RPCSEC_GSS-secured network services. Client principal names are stored in the form of <code>rpc_gss_principal_t</code> structures.
clock skew	The maximum amount of time that the internal system clocks on all hosts participating in the Kerberos authentication system can differ. If the clock skew is exceeded between any of the participating hosts, requests will be rejected. Clock skew can be specified in the <code>krb5.conf</code> file.
confidentiality	See privacy.
credential	An information package that includes a ticket and a matching session key. Used to authenticate the identity of a principal. See also ticket, session key.
credential cache	A storage space (usually a file) containing credentials received from the KDC.
flavor	Historically, <i>security flavor</i> and <i>authentication flavor</i> meant the same thing, as a flavor indicated a type of authentication (AUTH_UNIX, AUTH_DES, AUTH_KERB). RPCSEC_GSS is also a security flavor, even though it provides integrity and privacy services in addition to authentication.
forwardable ticket	A ticket that can be used by a client to request a ticket on a remote host without the client having to go through the full authentication process on that host. For example, if the user <code>david</code> obtains a forwardable ticket while on <code>jennifer</code> 's machine, he can log in to his own machine without having to get a new ticket (and thus authenticate himself again). See also proxiable ticket.
FQDN	Fully Qualified Domain Name. For example, <code>denver.mtn.acme.com</code> (as opposed to simply <code>denver</code>).
GSS-API	The Generic Security Service Application Programming Interface. A network layer providing support for various modular security services (including SEAM). GSS-API provides for security authentication, integrity, and privacy services. See also authentication, integrity, privacy.
host	A machine accessible over a network.
host principal	A particular instance of a service principal in which the principal (signified by the primary name <code>host</code>) is set up to provide a range of network services, such as <code>ftp</code> , <code>rcp</code> , or <code>rlogin</code> .

host/boston.eng.acme.com@ENG.ACME.COM is an example of a host principal. See also server principal.

initial ticket

A ticket that is issued directly (that is, not based on an existing ticket-granting ticket). Some services, such as applications that change passwords, might require tickets to be marked *initial* so as to assure themselves that the client can demonstrate a knowledge of its secret key — because an initial ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might have been around for a long time).

instance

The second part of a principal name, an instance qualifies the principal's primary. In the case of a service principal, the instance is required and is the host's fully qualified domain name, as in host/boston.eng.acme.com. For user principals, an instance is optional; note, however, that joe and joe/admin are unique principals. See also principal name, service principal, user principal.

integrity

A security service that, in addition to user authentication, provides for the validity of transmitted data through cryptographic checksumming. See also authentication, privacy.

invalid ticket

A postdated ticket that has not yet become usable. It will be rejected by an application server until it becomes validated. To be validated, it must be presented to the KDC by the client in a TGS request, with the VALIDATE flag set, after its start time has passed. See also postdated ticket.

KDC

(Key Distribution Center) A machine that has three Kerberos V5 components:

- Principal and key database
- Authentication service
- Ticket-granting service

Each realm has a master KDC and should have one or more slave KDCs.

Kerberos

An authentication service, the protocol used by that service, or the code used to implement that service.

SEAM is an authentication implementation closely based on Kerberos V5.

While technically different, "SEAM" and "Kerberos" are often used interchangeably in SEAM documentation; the same is true for "Kerberos" and "Kerberos V5."

Kerberos (also spelled Cerberus) was a fierce, three-headed mastiff who guarded the gates of Hades in Greek mythology.

key

1. An entry (principal name) in a keytab. (See keytab.)
2. An encryption key, of which there are three types:
 - a. A *private key*. An encryption key shared by a principal and the KDC, distributed outside the bounds of the system. See also private key.
 - b. A *service key*. This key serves the same purpose as the private key, but is used by servers and services. See also service key.
 - c. A *session key*. A temporary encryption key used between two principals, with a lifetime limited to the duration of a single login session. See also session key.

keytab

A key table file containing one or more keys (principals). A host or service uses a keytab file in the much the same way that a user uses a password.

kvno

Key Version Number. A sequence number tracking a particular key in order of generation. The highest kvno is the latest and current key.

management scope

The scope in which a role is permitted to operate, that is, an individual host or all hosts served by a specified name service such as NIS, NIS+, or LDAP. Scopes are applied to Solaris Management Console toolboxes.

master KDC

The main KDC in each realm, including a Kerberos administration server, `kadmind`, and an authentication and ticket-granting daemon, `krb5kdc`. Each realm must have at least one master KDC, and can have many duplicate, or slave, KDCs that provide authentication services to clients.

mechanism

A software package that specifies cryptographic techniques to achieve data authentication or confidentiality. Examples: Kerberos V5, Diffie-Hellman public key.

network application server

A server providing an network application, such as `ftp`. A realm can contain several network application servers.

NTP

(Network Time Protocol) Software from the University of Delaware that enables you to manage precise time and/or network clock synchronization in a network environment. You can use NTP to maintain clock skew in a Kerberos environment.

PAM

(Pluggable Authentication Module) A framework that allows for multiple authentication mechanisms to be used without having to recompile the services using them. PAM enables SEAM session initialization at login.

policy	A set of rules, initiated when SEAM is installed or administered, governing ticket usage. Policies can regulate principals' accesses, or ticket parameters, such as lifespan.
postdated ticket	A postdated ticket is one that does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs intended to be run late at night, since the ticket, if stolen, cannot be used until the batch job is to be run. When a postdated ticket is issued, it is issued as <i>invalid</i> and remains that way until a) its start time has passed, and b) the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket; however, if it is marked <i>renewable</i> , its lifetime is normally set to be equal to the duration of the full life of the ticket-granting ticket. See also invalid ticket, renewable ticket.
primary	The first part of a principal name. See also instance, principal name, realm.
principal	<ol style="list-style-type: none"> 1. A uniquely named client/user or server/service instance that participates in a network communication; Kerberos transactions involve interactions between principals (service principals and user principals) or between principals and KDCs. Put another way, a principal is a unique entity to which Kerberos can assign tickets. See also principal name, service principal, user principal. 2. (RPCSEC_GSS API) See client principal, server principal.
principal name	<ol style="list-style-type: none"> 1. The name of a principal, having the format of <i>primary/instance@REALM</i>. See also instance, primary, realm. 2. (RPCSEC_GSS API) See client principal, server principal.
privacy	A security service, in which transmitted data is encrypted before being sent. Privacy also includes data integrity and user authentication. See also authentication, integrity, service.
private key	A key is given to each user principal and known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. See also key.
private-key encryption	In private-key encryption, the sender and receiver use the same key for encryption. See also public-key encryption.
privileged application	An application that can override system controls and checks for specific UIDs, GIDs, or authorizations.
profile shell	A shell used in RBAC that enables a role (or user) to run any privileged applications assigned to the role's rights profiles from the

command line. The profile shells are `pfsh`, `pfssh`, and `pfksh`, and they correspond to Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`) respectively.

proxiabable ticket	A ticket that can be used by a service on behalf of a client to perform an operation for the client. (Thus the service is said to act as the client's proxy.) With the ticket, the service can take on the identity of the client. The service can use this to obtain a service ticket to another service, but it cannot obtain a ticket-granting ticket. The difference between a proxiabable ticket and a forwardable ticket is that a proxiabable ticket is only valid for a single operation. See also forwardable ticket.
public-key encryption	An encryption scheme in which each user has two keys, one public and one private. In public-key encryption, the sender uses the receiver's public key to encrypt the message, and the receiver uses a private key to decrypt it. SEAM is a private-key system. See also private-key encryption.
QOP	(Quality of Protection) A parameter used to select the cryptographic algorithms to be used in conjunction with the integrity or privacy service.
RBAC	(Role-Based Access Control) An alternative to the all-or-nothing superuser model. RBAC lets an organization separate superuser's capabilities and assign them to special user accounts called roles. Roles can be assigned to specific individuals according to their job needs.
realm	<ol style="list-style-type: none">1. The logical network served by a single SEAM database and a set of Key Distribution Centers (KDCs).2. The third part of a principal name. For the principal name <code>joe/admin@ENG.ACME.COM</code>, the realm is <code>ENG.ACME.COM</code>. See also principal name.
relation	A configuration variable or relationship defined in the <code>kdc.conf</code> or <code>krb5.conf</code> files.
renewable ticket	Because it is a security risk to have tickets with very long lives, tickets can be designated as <i>renewable</i> . A renewable ticket has two expiration times: the time at which the current instance of the ticket expires, and maximum lifetime for any ticket. If a client wants to continue to use a ticket, it renews it before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of ten hours. If the client holding the ticket wants to keep it for more than an hour, it must renew it. When a ticket reaches the maximum ticket lifetime, it automatically expires and cannot be renewed.
rights profile	(Also referred to as right or profile) A collection of overrides used in RBAC that can be assigned to a role or user. A rights profile can consist

	of authorizations, commands with set UIDs or GIDs (referred to as security attributes) and other rights profiles.
role	A special identity for running privileged applications that can be assumed by assigned users only.
SEAM	(Sun Enterprise Authentication Mechanism) A system for authenticating users over a network, based on the Kerberos V5 technology developed at the Massachusetts Institute of Technology. "SEAM" and "Kerberos" are often used interchangeably in the SEAM documentation.
secret key	See private key.
secure shell	A special protocol for secure remote login and other secure network services over an insecure network.
security flavor	See flavor.
security mechanism	See mechanism.
security service	See service.
server	A particular principal that provides a resource to network clients. For example, if you <code>rlogin</code> to the machine <code>boston.eng.acme.com</code> , then that machine is the server providing the <code>rlogin</code> service. See also service principal.
server principal	(RPCSEC_GSS API) A principal providing a service. It is stored as an ASCII string of the form <code>service@host</code> . See also client principal.
service	<ol style="list-style-type: none"> 1. A resource provided to network clients; often provided by more than one server. For example, if you <code>rlogin</code> to the machine <code>boston.eng.acme.com</code>, then that machine is the server providing the <code>rlogin</code> service. 2. A security service — either integrity or privacy, providing a level of protection beyond authentication. See also integrity and privacy.
service key	An encryption key shared by a service principal and the KDC, distributed outside the bounds of the system. See also key.
service principal	A principal that provides a Kerberos authentication for a service or services. For service principals, the primary name is a name of a service, such as <code>ftp</code> , and its instance is the fully qualified hostname of the system that provides the service. See also host principal, user principal.

session key	A key generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service. Its lifetime is limited to a single login session. See also key.
slave KDC	A copy of a master KDC, capable of performing most of the functions of the master. Each realm usually has several slave KDCs (and only one master KDC). See also KDC, master KDC.
stash file	A stash file contains an encrypted copy of the master key for the KDC. This key is used when a server is rebooted to automatically authenticate the KDC before starting <code>kadmind</code> and <code>krb5kdc</code> processes. Because this file includes the master key, the file and any backups of the file should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.
ticket	An information packet used to securely pass the identity of a user to a server or service. A ticket is good for only a single client and a particular service on a specific server. It contains the principal name of the service, the principal name of the user, the IP address of the user's host, a timestamp, and a value to define the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. Once a ticket has been created, it can be reused until the ticket expires. A ticket only serves to authenticate a client when presented along with a fresh authenticator. See also authenticator, credential, service, session key.
ticket file	See credential cache.
TGS	(Ticket-Granting Service) That portion of the KDC that is responsible for issuing tickets.
TGT	(Ticket-Granting Ticket) A ticket issued by the KDC that enables a client to request tickets for other services.
user principal	A principal attributed to a particular user, whose primary name is a user name and its optional instance is a name used to described the intended use of the corresponding credentials (for example, <code>joe</code> or <code>joe/admin</code>). Also known as a user instance. See also service principal.
VPN	(Virtual Private Network) A network that provides secure communication by using encryption and tunneling to connect users over a public network.