Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux États-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certaines composants de ce produit peuvent être dérivés du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux États-Unis et dans d'autres pays; elle est licensiée exclusivement par X/Open Company Ltd.

Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, SunOS, Java, JumpStart, Trusted Solaris, Java, et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux États-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 États-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture déposée par Sun Microsystems, Inc. Le produit de Xylogics est protégé par le copyright et autorisé au Sun par Xylogics. Xylogics et Annex sont des marques déposées de Xylogics, Inc., Copyright 1996 des portions du logiciel par Massachusetts Institute of Technology. Tous droits réservés.

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.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des États-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIÈRE OU A L'ABSENCE DE CONTREFACON.
# Contents

Preface ........................................................................................................................................................... 23

Part I  Security Overview ....................................................................................................................................... 27

1  Security Services (Overview) ........................................................................................................................... 29  
   System Security .............................................................................................................................................. 29  
   Solaris Cryptographic Services ....................................................................................................................... 30  
   Authentication Services ................................................................................................................................. 31  
   Authentication With Encryption ...................................................................................................................... 32  
   Solaris Auditing .............................................................................................................................................. 32  
   Security Policy ............................................................................................................................................... 32

Part II  System, File, and Device Security ............................................................................................................. 35

2  Managing Machine Security (Overview) ........................................................................................................ 37  
   Enhancements to Machine Security in the Solaris 10 Release ....................................................................... 37  
   Controlling Access to a Computer System .................................................................................................. 38  
      Maintaining Physical Security .................................................................................................................. 38  
      Maintaining Login Control ...................................................................................................................... 39  
   Controlling Access to Devices .................................................................................................................... 44  
      Device Policy (Overview) ......................................................................................................................... 45  
      Device Allocation (Overview) ................................................................................................................. 45  
   Controlling Access to Machine Resources ................................................................................................. 46  
      Limiting and Monitoring Superuser .......................................................................................................... 46  
      Configuring Role-Based Access Control to Replace Superuser ............................................................. 46  
      Preventing Unintentional Misuse of Machine Resources ........................................................................ 46  
      Restricting setuid Executable Files ......................................................................................................... 48
Using the Automated Security Enhancement Tool ................................................................. 48
Using the Solaris Security Toolkit ............................................................................................ 48
Using Solaris Resource Management Features ................................................................. 49
Using Solaris Zones .............................................................................................................. 49
Monitoring Use of Machine Resources ............................................................................ 49
Monitoring File Integrity .................................................................................................... 49
Controlling Access to Files .................................................................................................. 50
Protecting Files With Encryption ........................................................................................ 50
Using Access Control Lists ................................................................................................. 50
Sharing Files Across Machines .......................................................................................... 51
Restricting root Access to Shared Files .............................................................................. 51
Controlling Network Access ............................................................................................... 51
Network Security Mechanisms ............................................................................................ 51
Authentication and Authorization for Remote Access ....................................................... 52
Firewall Systems .................................................................................................................. 54
Encryption and Firewall Systems ......................................................................................... 55
Reporting Security Problems ............................................................................................... 55

3 Controlling Access to Systems (Tasks) ............................................................................. 57
Controlling System Access (Task Map) ............................................................................. 57
Securing Logins and Passwords (Task Map) ...................................................................... 58
Securing Logins and Passwords .......................................................................................... 58
▼ How to Display a User’s Login Status ........................................................................... 58
▼ How to Display Users Without Passwords ................................................................. 59
▼ How to Temporarily Disable User Logins ..................................................................... 60
▼ How to Monitor Failed Login Attempts ....................................................................... 61
▼ How to Monitor All Failed Login Attempts ............................................................... 62
▼ How to Create a Dial-Up Password ............................................................................. 63
▼ How to Temporarily Disable Dial-Up Logins ............................................................. 65
Changing the Password Algorithm (Task Map) .................................................................. 65
Changing the Default Algorithm for Password Encryption .............................................. 66
▼ How to Specify an Algorithm for Password Encryption .............................................. 66
▼ How to Specify a New Password Algorithm for an NIS Domain ............................... 67
▼ How to Specify a New Password Algorithm for an NIS+ Domain ............................. 67
▼ How to Specify a New Password Algorithm for an LDAP Domain ......................... 67
▼ How to Install a Password Encryption Module From a Third Party ......................... 68
Contents

7 Using the Automated Security Enhancement Tool (Tasks) ................................................................. 143

Automated Security Enhancement Tool (ASET) .................................................................................... 143
  ASET Security Levels .......................................................................................................................... 144
  ASET Task List ..................................................................................................................................... 145
  ASET Execution Log ............................................................................................................................ 147
  ASET Reports ...................................................................................................................................... 148
  ASET Master Files ............................................................................................................................... 151
  ASET Environment File (asetenv) ........................................................................................................ 152
  Configuring ASET .................................................................................................................................. 152
  Restoring System Files Modified by ASET ......................................................................................... 154
  Network Operation With the NFS System ............................................................................................ 155
  ASET Environment Variables .............................................................................................................. 155
  ASET File Examples ............................................................................................................................ 158

Running ASET (Task Map) .................................................................................................................. 160
  ▼ How to Run ASET Interactively ...................................................................................................... 160
  ▼ How to Run ASET Periodically ...................................................................................................... 161
  ▼ How to Stop Running ASET Periodically ...................................................................................... 162
  ▼ How to Collect ASET Reports on a Server ..................................................................................... 162

Troubleshooting ASET Problems ........................................................................................................ 164
  ASET Error Messages .......................................................................................................................... 164
10 Role-Based Access Control (Reference) ................................................................. 213

Contents of Rights Profiles .................................................................................. 213
  Primary Administrator Rights Profile ................................................................. 214
  System Administrator Rights Profile ................................................................. 214
  Operator Rights Profile ...................................................................................... 215
  Printer Management Rights Profile ................................................................. 215
  Basic Solaris User Rights Profile ....................................................................... 216
  All Rights Profile ............................................................................................. 217
  Order of Rights Profiles .................................................................................... 217
  Viewing the Contents of Rights Profiles ........................................................... 217

Authorization Naming and Delegation ............................................................... 218
  Authorization Naming Conventions ................................................................. 218
  Example of Authorization Granularity ............................................................ 218
  Delegation Authority in Authorizations ............................................................ 218

Databases That Support RBAC ........................................................................ 219
  RBAC Database Relationships ......................................................................... 219
  RBAC Databases and the Name Service ............................................................ 220
  user_attr Database ........................................................................................ 220
  auth_attr Database ........................................................................................ 221
  prof_attr Database ........................................................................................ 222
  exec_attr Database ........................................................................................ 223
  policy.conf File ................................................................................................ 224

RBAC Commands .............................................................................................. 225
  Commands That Manage RBAC ....................................................................... 225
  Commands That Require Authorizations ......................................................... 226
14 Solaris Cryptographic Framework (Tasks) .......................................................................................................................... 255
   Using the Cryptographic Framework (Task Map) ...................................................................................................................... 255
   Protecting Files With the Solaris Cryptographic Framework (Task Map) .................................................................................. 256
   Protecting Files With the Solaris Cryptographic Framework .................................................................................................... 256
   ▼ How to Generate a Symmetric Key .................................................................................................................................. 256
   ▼ How to Compute a Digest of a File .................................................................................................................................... 258
   ▼ How to Compute a MAC of a File ....................................................................................................................................... 259
   ▼ How to Encrypt and Decrypt a File .................................................................................................................................... 261
   Administering the Cryptographic Framework (Task Map) ..................................................................................................... 263
   Administering the Cryptographic Framework .......................................................................................................................... 264
   ▼ How to List Available Providers ...................................................................................................................................... 264
   ▼ How to Add a Software Provider ...................................................................................................................................... 266
   ▼ How to Prevent the Use of a User-Level Mechanism ......................................................................................................... 268
   ▼ How to Prevent the Use of a Kernel Software Provider .................................................................................................... 269
   ▼ How to List Hardware Providers ...................................................................................................................................... 272
   ▼ How to Disable Hardware Provider Mechanisms and Features ............................................................................................ 272
   ▼ How to Refresh or Restart All Cryptographic Services .................................................................................................. 274

Part V Authentication Services and Secure Communication ................................................................................................................. 275

15 Using Authentication Services (Tasks) ........................................................................................................................................ 277
   Overview of Secure RPC ............................................................................................................................................................. 277
   NFS Services and Secure RPC ................................................................................................................................................... 277
   DES Encryption With Secure NFS ........................................................................................................................................... 278
   Kerberos Authentication .............................................................................................................................................................. 278
   Diffie-Hellman Authentication .................................................................................................................................................. 278
   Administering Secure RPC (Task Map) .................................................................................................................................. 281
   Administering Authentication With Secure RPC ...................................................................................................................... 282
   ▼ How to Restart the Secure RPC Keyserver ............................................................................................................................ 282
   ▼ How to Set Up a Diffie-Hellman Key for an NIS+ Host .......................................................................................................... 282
   ▼ How to Set Up a Diffie-Hellman Key for an NIS+ User ........................................................................................................ 284
   ▼ How to Set Up a Diffie-Hellman Key for an NIS Host .......................................................................................................... 284
   ▼ How to Set Up a Diffie-Hellman Key for an NIS User .......................................................................................................... 285
   ▼ How to Share NFS Files With Diffie-Hellman Authentication ............................................................................................. 286
## Using PAM

- **PAM (Overview)** .......................................................... 289
- **Benefits of Using PAM** .................................................. 289
- **PAM Components** .......................................................... 290
- **Changes to PAM for the Solaris 10 Release** ...................... 291
- **PAM (Tasks)** ................................................................. 292
- **PAM (Task Map)** ............................................................. 292
- **Planning for Your PAM Implementation** ......................... 292
- **How to Add a PAM Module** .......................................... 293
- **How to Prevent Rhost-Style Access From Remote Systems With PAM** .................................................. 294
- **How to Log PAM Error Reports** .................................... 294
- **PAM Configuration File (Reference)** ............................... 294
- **PAM Configuration File Syntax** ...................................... 294
- **Service Names for PAM** .................................................. 295
- **PAM Module Types** .......................................................... 295
- **PAM Control Flags** .......................................................... 295
- **PAM Modules** ............................................................... 297
- **Examples From the Generic `pam.conf` File** .................... 297

## Using SASL

- **SASL (Overview)** .......................................................... 299
- **SASL (Reference)** .......................................................... 300
- **SASL Plug-ins** ............................................................... 300
- **SASL Environment Variable** .......................................... 300
- **SASL Options** .............................................................. 301

## Using Solaris Secure Shell (Tasks)

- **Solaris Secure Shell (Overview)** .................................... 303
- **Solaris Secure Shell Authentication** .................................. 304
- **Solaris Secure Shell in the Enterprise** .............................. 306
- **Solaris Secure Shell Enhancements in the Solaris 10 Release** ............................ 306
- **Solaris Secure Shell (Task Map)** ..................................... 307
- **Configuring Solaris Secure Shell (Task Map)** ..................... 307
- **Configuring Solaris Secure Shell** ..................................... 307
- **How to Set Up Host-Based Authentication for Solaris Secure Shell** ......................... 307
- **How to Enable Solaris Secure Shell v1** ............................. 310
Contents

19 Solaris Secure Shell (Reference) ..........................................................323
   A Typical Solaris Secure Shell Session ..............................................323
       Session Characteristics in Solaris Secure Shell .........................323
       Authentication and Key Exchange in Solaris Secure Shell ..........324
       Command Execution and Data Forwarding in Solaris Secure Shell ...325
   Client and Server Configuration in Solaris Secure Shell ..................325
       Client Configuration in Solaris Secure Shell ............................325
       Server Configuration in Solaris Secure Shell ............................326
   Keywords in Solaris Secure Shell ...................................................326
       Host-Specific Parameters in Solaris Secure Shell .......................330
       Solaris Secure Shell and Login Environment Variables ............330
   Maintaining Known Hosts in Solaris Secure Shell .........................331
   Solaris Secure Shell Packages and Initialization .........................331
   Solaris Secure Shell Files ............................................................332
   Solaris Secure Shell Commands ...................................................334

Part VI Kerberos Service .................................................................337

20 Introduction to the Kerberos Service ...........................................339
   What Is the Kerberos Service? .......................................................339
   How the Kerberos Service Works ..................................................340
       Initial Authentication: the Ticket-Granting Ticket ....................341
       Subsequent Kerberos Authentications .................................342
       The Kerberos Remote Applications .................................344
21 Planning for the Kerberos Service ................................................................................................. 353

Why Plan for Kerberos Deployments? ............................................................................................. 353
Kerberos Realms .............................................................................................................................. 354
    Realm Names .............................................................................................................................. 354
    Number of Realms ...................................................................................................................... 354
    Realm Hierarchy .......................................................................................................................... 354
Mapping Host Names Onto Realms ..................................................................................................... 355
Client and Service Principal Names .................................................................................................. 355
Ports for the KDC and Admin Services ............................................................................................... 356
The Number of Slave KDCs .............................................................................................................. 356
Mapping GSS Credentials to UNIX Credentials ............................................................................. 357
Automatic User Migration to a Kerberos Realm ............................................................................... 357
Which Database Propagation System to Use ................................................................................... 357
Clock Synchronization Within a Realm ............................................................................................... 358
Client Installation Options ................................................................................................................ 358
Kerberos Encryption Types ................................................................................................................ 358
Online Help URL in the SEAM Administration Tool ......................................................................... 359

22 Configuring the Kerberos Service (Tasks) .................................................................................... 361
Configuring the Kerberos Service (Task Map) .................................................................................. 361
Configuring Additional Kerberos Services (Task Map) ..................................................................... 362
Configuring KDC Servers ................................................................................................................ 363
    ▼ How to Configure a Master KDC ........................................................................................... 363
# Contents

- How to Remove a Stash File ................................................................. 414
- Increasing Security on Kerberos Servers ........................................ 414
- How to Enable Only Kerberized Applications ............................... 414
- How to Restrict Access to KDC Servers ......................................... 415

## 23 Kerberos Error Messages and Troubleshooting ............................. 417
- Kerberos Error Messages ................................................................. 417
- SEAM Administration Tool Error Messages ................................. 417
- Common Kerberos Error Messages (A–M) .................................... 418
- Common Kerberos Error Messages (N–Z) .................................... 425
- Kerberos Troubleshooting ............................................................... 429
  - Problems With the Format of the `krb5.conf` File ...................... 429
  - Problems Propagating the Kerberos Database ......................... 429
  - Problems Mounting a Kerberized NFS File System .................. 429
  - Problems Authenticating as `root` ............................................ 430
  - Observing Mapping from GSS Credentials to UNIX Credentials ... 430

## 24 Administering Kerberos Principals and Policies (Tasks) ............... 431
- Ways to Administer Kerberos Principals and Policies .................... 431
- SEAM Administration Tool .............................................................. 432
  - Command-Line Equivalents of the SEAM Tool ......................... 432
  - The Only File Modified by the SEAM Tool ............................... 433
  - Print and Online Help Features of the SEAM Tool ................. 433
  - Working With Large Lists in the SEAM Tool ......................... 434
  - How to Start the SEAM Tool .................................................... 434
- Administering Kerberos Principals ................................................. 436
  - Administering Kerberos Principals (Task Map) ...................... 437
  - Automating the Creation of New Kerberos Principals ............ 437
  - How to View the List of Kerberos Principals ......................... 438
  - How to View a Kerberos Principal’s Attributes ...................... 440
  - How to Create a New Kerberos Principal ............................... 442
  - How to Duplicate a Kerberos Principal .................................. 444
  - How to Modify a Kerberos Principal ...................................... 444
  - How to Delete a Kerberos Principal ........................................ 445
  - How to Set Up Defaults for Creating New Kerberos Principals ... 446
  - How to Modify the Kerberos Administration Privileges .......... 447
Contents

Administering Kerberos Policies ..............................................................................................................449
   Administering Kerberos Policies (Task Map) ..........................................................................................449
   ▼ How to View the List of Kerberos Policies .........................................................................................449
   ▼ How to View a Kerberos Policy’s Attributes ......................................................................................451
   ▼ How to Create a New Kerberos Policy .................................................................................................453
   ▼ How to Duplicate a Kerberos Policy ..................................................................................................455
   ▼ How to Modify a Kerberos Policy ......................................................................................................455
   ▼ How to Delete a Kerberos Policy .......................................................................................................456
SEAM Tool Reference ................................................................................................................................457
   SEAM Tool Panel Descriptions ............................................................................................................457
   Using the SEAM Tool With Limited Kerberos Administration Privileges ........................................460
Administering Keytab Files ..........................................................................................................................461
   Administering Keytab Files (Task Map) .................................................................................................462
   ▼ How to Add a Kerberos Service Principal to a Keytab File ..........................................................463
   ▼ How to Remove a Service Principal From a Keytab File ............................................................464
   ▼ How to Display the Keylist (Principals) in a Keytab File ...............................................................465
   ▼ How to Temporarily Disable Authentication for a Service on a Host ........................................466

25 Using Kerberos Applications (Tasks) ......................................................................................................469
   Kerberos Ticket Management ..................................................................................................................469
      Do You Need to Worry About Tickets? ..............................................................................................469
      Creating a Kerberos Ticket ................................................................................................................470
      Viewing Kerberos Tickets ................................................................................................................470
      Destroying Kerberos Tickets ..............................................................................................................472
   Kerberos Password Management .............................................................................................................473
      Advice on Choosing a Password .......................................................................................................473
      Changing Your Password ................................................................................................................473
      Granting Access to Your Account ....................................................................................................476
   Kerberos User Commands .......................................................................................................................477
      Overview of Kerberized Commands ................................................................................................477
      Forwarding Kerberos Tickets ..............................................................................................................480
      Examples — Using Kerberized Commands ....................................................................................481

26 The Kerberos Service (Reference) ...........................................................................................................485
   Kerberos Files ......................................................................................................................................485
   Kerberos Commands .................................................................................................................................487
### Contents

- AuditClassSyntax .......................................................................................................................... 566
- Definitions of Audit Classes ........................................................................................................ 565
- Audit Policy .................................................................................................................................... 567
- Process Audit Characteristics ....................................................................................................... 567
- Audit Trail ....................................................................................................................................... 568
- Conventions for Binary Audit File Names ................................................................................... 568
- Binary Audit File Names ................................................................................................................ 568
- Binary Audit File Timestamps ....................................................................................................... 569
- Audit Record Structure .................................................................................................................. 569
- Audit Record Analysis .................................................................................................................... 570
- Audit Token Formats ..................................................................................................................... 571
- acl Token ......................................................................................................................................... 572
- arbitrary Token (Obsolete) ............................................................................................................... 572
- arg Token ......................................................................................................................................... 573
- attribute Token ............................................................................................................................... 574
- cmd Token ......................................................................................................................................... 574
- exec_args Token .............................................................................................................................. 575
- exec_env Token ............................................................................................................................... 575
- exit Token (Obsolete) ...................................................................................................................... 575
- file Token ......................................................................................................................................... 576

---

### File Used in the Auditing Service

- bsmrecord Command ....................................................................................................................... 555
- auditreduce Command ....................................................................................................................... 555
- praudit Command ............................................................................................................................. 556
- auditconfig Command ...................................................................................................................... 558

### Rights Profiles for Administering Auditing

- system File ......................................................................................................................................... 559
- syslog.conf File .................................................................................................................................. 559
- audit_class File .................................................................................................................................. 559
- audit_control File .............................................................................................................................. 559
- audit_event File .................................................................................................................................. 561
- audit_startup Script .......................................................................................................................... 561
- audit_user Database ......................................................................................................................... 561
- audit_warn Script ............................................................................................................................. 562
- bsmconv Script .................................................................................................................................. 563

### Auditing and Solaris Zones

- Audit Classes ..................................................................................................................................... 564
- Audit Classes
  - Definitions of Audit Classes ........................................................................................................ 565
  - Audit Class Syntax .......................................................................................................................... 566
  - Audit Policy .................................................................................................................................... 567
  - Process Audit Characteristics ....................................................................................................... 567
  - Audit Trail ....................................................................................................................................... 568

### Conventions for Binary Audit File Names

- Binary Audit File Names .................................................................................................................. 568
- Binary Audit File Timestamps .......................................................................................................... 569

### Audit Record Structure

- Audit Record Structure ..................................................................................................................... 569

### Audit Record Analysis

- Audit Record Analysis ...................................................................................................................... 570

### Audit Token Formats

- Audit Token Formats
  - acl Token ......................................................................................................................................... 572
  - arbitrary Token (Obsolete) ............................................................................................................. 572
  - arg Token ........................................................................................................................................ 573
  - attribute Token ............................................................................................................................. 574
  - cmd Token ........................................................................................................................................ 574
  - exec_args Token ............................................................................................................................. 575
  - exec_env Token .............................................................................................................................. 575
  - exit Token (Obsolete) ..................................................................................................................... 575
  - file Token ......................................................................................................................................... 576
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>group Token (Obsolete)</td>
<td>576</td>
</tr>
<tr>
<td>groups Token</td>
<td>576</td>
</tr>
<tr>
<td>header Token</td>
<td>577</td>
</tr>
<tr>
<td>in_addr Token</td>
<td>577</td>
</tr>
<tr>
<td>ip Token (Obsolete)</td>
<td>578</td>
</tr>
<tr>
<td>ipc Token</td>
<td>578</td>
</tr>
<tr>
<td>ipc_perm Token</td>
<td>579</td>
</tr>
<tr>
<td>iport Token</td>
<td>579</td>
</tr>
<tr>
<td>opaque Token (Obsolete)</td>
<td>579</td>
</tr>
<tr>
<td>path Token</td>
<td>580</td>
</tr>
<tr>
<td>path_attr Token</td>
<td>580</td>
</tr>
<tr>
<td>privilege Token</td>
<td>581</td>
</tr>
<tr>
<td>process Token</td>
<td>581</td>
</tr>
<tr>
<td>return Token</td>
<td>582</td>
</tr>
<tr>
<td>sequence Token</td>
<td>583</td>
</tr>
<tr>
<td>socket Token</td>
<td>583</td>
</tr>
<tr>
<td>subject Token</td>
<td>584</td>
</tr>
<tr>
<td>text Token</td>
<td>585</td>
</tr>
<tr>
<td>trailer Token</td>
<td>586</td>
</tr>
<tr>
<td>uauth Token</td>
<td>586</td>
</tr>
<tr>
<td>zonename Token</td>
<td>586</td>
</tr>
</tbody>
</table>

**Glossary** .......................................................................................................................................................589

**Index** ............................................................................................................................................................601
Preface

*System Administration Guide: Security Services* is part of a multivolume set that covers a significant part of the Solaris™ Operating System administration information. This book assumes that you have already installed the SunOS™ 5.10 operating system, and you have set up any networking software that you plan to use. The SunOS 5.10 operating system is part of the Solaris 10 product family, which includes many features, such as the Solaris Common Desktop Environment (CDE).

**Note** – This Solaris release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems appear in the *Solaris 10 Hardware Compatibility List* at http://www.sun.com/bigadmin/hcl. This document cites any implementation differences between the platform types.

In this document these x86 related terms mean the following:

- “x86” refers to the larger family of 64-bit and 32-bit x86 compatible products.
- “x64” points out specific 64-bit information about AMD64 or EM64T systems.
- “32-bit x86” points out specific 32-bit information about x86 based systems.

For supported systems, see the *Solaris 10 Hardware Compatibility List*.

**Who Should Use This Book**

This book is intended for anyone who is responsible for administering one or more systems that run the Solaris 10 release. To use this book, you should have more than two years of UNIX® system administration experience. Attending training courses in UNIX system administration might be helpful.
How the System Administration Volumes Are Organized

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

<table>
<thead>
<tr>
<th>Book Title</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Administration Guide: Basic Administration</td>
<td>User accounts and groups, server and client support, shutting down and booting a system, managing services, and managing software (packages and patches)</td>
</tr>
<tr>
<td>System Administration Guide: Advanced Administration</td>
<td>Printing services, terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems</td>
</tr>
<tr>
<td>System Administration Guide: Devices and File Systems</td>
<td>Removable media, disks and devices, file systems, and backing up and restoring data</td>
</tr>
<tr>
<td>System Administration Guide: IP Services</td>
<td>TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, Solaris IP filter, Mobile IP, IP network multipathing (IPMP), and IPQoS</td>
</tr>
<tr>
<td>System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)</td>
<td>DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP and transitioning from NIS+ to LDAP</td>
</tr>
<tr>
<td>System Administration Guide: Naming and Directory Services (NIS+)</td>
<td>NIS+ naming and directory services</td>
</tr>
<tr>
<td>System Administration Guide: Network Services</td>
<td>Web cache servers, time-related services, network file systems (NFS and Autos), mail, SLP, and PPP</td>
</tr>
<tr>
<td>System Administration Guide: Security Services</td>
<td>Auditing, device management, file security, BART, Kerberos services, PAM, Solaris Cryptographic Framework, privileges, RBAC, SASL, and Solaris Secure Shell</td>
</tr>
<tr>
<td>System Administration Guide: Solaris Containers-Resource Management and Solaris Zones</td>
<td>Resource management topics projects and tasks, extended accounting, resource controls, fair share scheduler (FSS), physical memory control using the resource capping daemon (rcapd), and dynamic resource pools; virtualization using Solaris Zones software partitioning technology</td>
</tr>
</tbody>
</table>

Related Third-Party Web Site References

Third party URLs are referenced in this document and provide additional, related information.

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.
Documentation, Support, and Training

The Sun website provides information about the following additional resources:

- Documentation (http://www.sun.com/documentation/)
- Support (http://www.sun.com/support/)
- Training (http://www.sun.com/training/)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AaBbCc123</td>
<td>The names of commands, files, and directories, and onscreen computer output</td>
<td>Edit your .login file. Use ls -a to list all files. machine_name% you have mail.</td>
</tr>
<tr>
<td>AaBbCc123</td>
<td>What you type, contrasted with onscreen computer output</td>
<td>machine_name% su</td>
</tr>
<tr>
<td>aabbcc123</td>
<td>Placeholder: replace with a real name or value</td>
<td>The command to remove a file is rm filename.</td>
</tr>
<tr>
<td>AaBbCc123</td>
<td>Book titles, new terms, and terms to be emphasized</td>
<td>Read Chapter 6 in the User's Guide. A cache is a copy that is stored locally. Do not save the file. Note: Some emphasized items appear bold online.</td>
</tr>
</tbody>
</table>

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.
<table>
<thead>
<tr>
<th>Shell</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C shell</td>
<td><code>machine_name%</code></td>
</tr>
<tr>
<td>C shell for superuser</td>
<td><code>machine_name#</code></td>
</tr>
<tr>
<td>Bourne shell and Korn shell</td>
<td><code>$</code></td>
</tr>
<tr>
<td>Bourne shell and Korn shell for superuser</td>
<td><code>#</code></td>
</tr>
</tbody>
</table>
Security Overview

This book focuses on the features that enhance security in the Solaris Operating System. This book is intended for system administrators and users of these security features. The overview chapter introduces the topics in the book.
Security Services (Overview)

To maintain the security of the Solaris Operating System (Solaris OS), Solaris software provides the following features:

- "System Security" on page 29 – The ability to prevent intrusion, to protect machine resources and devices from misuse, and to protect files from malicious modification or unintentional modification by users or intruders
  
  For a general discussion of system security, see Chapter 2.

- "Solaris Cryptographic Services" on page 30 – The ability to scramble data so that only the sender and the designated receiver can read the contents, and to manage cryptographic providers

- "Authentication Services" on page 31 – The ability to securely identify a user, which requires the user’s name and some form of proof, typically a password

- "Authentication With Encryption" on page 32 – The ability to ensure that authenticated parties can communicate without interception, modification, or spoofing

- "Solaris Auditing" on page 32 – The ability to identify the source of security changes to the system, including file access, security-related system calls, and authentication failures

- "Security Policy" on page 32 – The design and implementation of security guidelines for a computer or network of computers

System Security

System security ensures that the system’s resources are used properly. Access controls can restrict who is permitted access to resources on the system. The Solaris OS features for system security and access control include the following:

- **Login administration tools** – Commands for monitoring and controlling a user’s ability to log in. See "Securing Logins and Passwords (Task Map)" on page 58.

- **Hardware access** – Commands for limiting access to the PROM, and for restricting who can boot the system. See “SPARC: Controlling Access to System Hardware (Task Map)” on page 72.
Resource access – Tools and strategies for maximizing the appropriate use of machine resources while minimizing the misuse of those resources. See “Controlling Access to Machine Resources” on page 46.

Role-based access control (RBAC) – An architecture for creating special, restricted user accounts that are permitted to perform specific administrative tasks. See “Role-Based Access Control (Overview)” on page 169.

Privileges – Discrete rights on processes to perform operations. These process rights are enforced in the kernel. See “Privileges (Overview)” on page 178.

Device management – Device policy additionally protects devices that are already protected by UNIX permissions. Device allocation controls access to peripheral devices, such as a microphone or CD-ROM drive. Upon deallocation, device-clean scripts can then erase any data from the device. See “Controlling Access to Devices” on page 44.

Basic Audit Reporting Tool (BART) – A snapshot, called a manifest, of the file attributes of files on a system. By comparing the manifests across systems or on one system over time, changes to files can be monitored to reduce security risks. See Chapter 5.

File permissions – Attributes of a file or directory. Permissions restrict the users and groups that are permitted to read, write, or execute a file, or search a directory. See Chapter 6.

Security enhancement scripts – Through the use of scripts, many system files and parameters can be adjusted to reduce security risks. See Chapter 7.

Solaris Cryptographic Services

Cryptography is the science of encrypting and decrypting data. Cryptography is used to insure integrity, privacy, and authenticity. Integrity means that the data has not been altered. Privacy means that the data is not readable by others. Authenticity for data means that what was delivered is what was sent. User authentication means that the user has supplied one or more proofs of identity. Authentication mechanisms mathematically verify the source of the data or the proof of identity. Encryption mechanisms scramble data so that the data is not readable by a casual observer. Cryptographic services provide authentication and encryption mechanisms to applications and users.

Cryptographic algorithms use hashing, chaining, and other mathematical techniques to create ciphers that are difficult to break. Authentication mechanisms require that the sender and the receiver compute an identical number from the data. Encryption mechanisms rely on the sender and the receiver sharing information about the method of encryption. This information enables only the receiver and the sender to decrypt the message. The Solaris OS provides a centralized cryptographic framework, and provides encryption mechanisms that are tied to particular applications.

Solaris™ Cryptographic Framework – A central framework of cryptographic services for kernel-level and user-level consumers. Uses include passwords, IPsec, and third-party applications. The cryptographic framework includes a number of software encryption modules. The framework enables you to specify which software encryption modules or hardware encryption sources an application can use. The framework is built on the PKCS #11 v2 library.
This library is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). The library provides an API for third-party developers to plug in the cryptographic requirements for their applications. See Chapter 13.

- **Encryption mechanisms per application** –
  - For the use of DES in Secure RPC, see “Overview of Secure RPC” on page 277.
  - For the use of DES, 3DES, AES, and ARCFOUR in the Kerberos service, see Chapter 20.
  - For the use of RSA, DSA, and ciphers such as Blowfish in Solaris Secure Shell, see Chapter 18.
  - For the use of cryptographic algorithms in passwords, see “Changing the Password Algorithm (Task Map)” on page 65.

**Authentication Services**

Authentication is a mechanism that identifies a user or service based on predefined criteria. Authentication services 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, and a personal item that can be verified. A user name is an example of information that the person knows. A smart card or a fingerprint, for example, can be verified. The Solaris features for authentication include the following:

- **Secure RPC** – An authentication mechanism that uses the Diffie-Hellman protocol to protect NFS mounts and a name service, such as NIS or NIS+. See “Overview of Secure RPC” on page 277.
- **Pluggable Authentication Module (PAM)** – A framework that enables various authentication technologies to be plugged into a system entry service without recompiling the service. Some of the system entry services include `login` and `ftp`. See Chapter 16.
- **Simple Authentication and Security Layer (SASL)** – A framework that provides authentication and security services to network protocols. See Chapter 17.
- **Solaris Secure Shell** – A secure remote login and transfer protocol that encrypts communications over an insecure network. See Chapter 18.
- **Kerberos service** – A client-server architecture that provides encryption with authentication. See Chapter 20.
- **Solaris 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.
Authentication With Encryption

Authentication with encryption is the basis of secure communication. Authentication helps ensure that the source and the destination are the intended parties. Encryption codes the communication at the source, and decodes the communication at the destination. Encryption prevents intruders from reading any transmissions that the intruders might manage to intercept. The Solaris features for secure communication include the following:

- **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 services and other network services can be tunneled safely over Secure Shell connections for additional protection. See Chapter 18.

- **Kerberos service** – A client-server architecture that provides authentication with encryption. See Chapter 20.


Solaris Auditing

Auditing is a fundamental concept of system security and maintainability. Auditing is the process of examining the history of actions and events on a system to determine what happened. The history is kept in a log of what was done, when it was done, by whom, and what was affected. See Chapter 27.

Security Policy

The phrase security policy, or policy, is used throughout this book to refer to an organization’s security guidelines. Your site’s security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. Security technologies such as Solaris Secure Shell, authentication, RBAC, authorization, privileges, and resource control provide measures to protect information.

Some security technologies also use the word policy when describing specific aspects of their implementation. For example, Solaris auditing uses audit policy options to configure some aspects of auditing policy. The following table points to glossary, man page, and information on features that use the word policy to describe specific aspects of their implementation.
## TABLE 1–1 Use of Policy in the Solaris OS

<table>
<thead>
<tr>
<th>Glossary Definition</th>
<th>Selected Man Pages</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit policy</td>
<td>audit_control(4), audit_user(4), auditconfig(1M)</td>
<td>Chapter 27</td>
</tr>
<tr>
<td>policy in the cryptographic framework</td>
<td>cryptoadm(1M)</td>
<td>Chapter 13</td>
</tr>
<tr>
<td>device policy</td>
<td>getdevpolicy(1M)</td>
<td>“Controlling Access to Devices” on page 44</td>
</tr>
<tr>
<td>Kerberos policy</td>
<td>krb5.conf(4)</td>
<td>Chapter 24</td>
</tr>
<tr>
<td>network policies</td>
<td>ipfilter(5), ifconfig(1M), ike.config(4), ipseccnf(1M), routeadm(1M)</td>
<td>Part IV, “IP Security,” in <em>System Administration Guide: IP Services</em></td>
</tr>
<tr>
<td>password policy</td>
<td>passwd(1), nsswitch.conf(4), crypt.conf(4), policy.conf(4)</td>
<td>“Maintaining Login Control” on page 39</td>
</tr>
<tr>
<td>RBAC policy</td>
<td>rbac(5)</td>
<td>“exec_attr Database” on page 223</td>
</tr>
</tbody>
</table>
PART II

System, File, and Device Security

This section covers security that can be configured on a non-networked system. The chapters discuss planning, monitoring, and controlling access to the disk, to files, and to peripheral devices.
Keeping a machine’s information secure is an important system administration responsibility. This chapter provides overview information about managing machine security.

The following is a list of the overview information in this chapter.

- “Enhancements to Machine Security in the Solaris 10 Release” on page 37
- “Controlling Access to a Computer System” on page 38
- “Controlling Access to Devices” on page 44
- “Controlling Access to Machine Resources” on page 46
- “Controlling Access to Files” on page 50
- “Controlling Network Access” on page 51
- “Reporting Security Problems” on page 55

## Enhancements to Machine Security in the Solaris 10 Release

Since the Solaris 9 release, the following features have been introduced to enhance system security:

- Strong password encryption is available and configurable. For more information, see “Password Encryption” on page 40.
- Device policy is enforced with privileges. For more information, see “Device Policy (Overview)” on page 45.
  
  For device allocation, the `/etc/security/dev` directory might not be supported in future releases of the Solaris OS.
- The Basic Audit Reporting Tool (BART) can monitor the authenticity of the files on your system. For more information, see Chapter 5.
- Files can be protected with strong encryption. For more information, see “Protecting Files With Encryption” on page 50.
- Privileges enforce process rights at the kernel level. For more information, see “Privileges (Overview)” on page 178.
The Solaris Cryptographic Framework centralizes cryptographic services for providers and for consumers. For more information, see Chapter 13.

The PAM framework provides functionality for many programs, such as Solaris Secure Shell. For more information, see “Changes to PAM for the Solaris 10 Release” on page 291.

Solaris zones and resource management control access to machine resources. For more information, see System Administration Guide: Solaris Containers-Resource Management and Solaris Zones.

Controlling Access to a Computer System

In the workplace, all machines that are connected to a server can be thought of as one large multifaceted system. You are responsible for the security of this larger system. You need to defend the network from outsiders who are trying to gain access to the network. You also need to ensure the integrity of the data on the machines within the network.

At the file level, the Solaris OS provides 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. The first line of security defense is to control access to your system. You can control and monitor system access by doing the following:

- “Maintaining Physical Security” on page 38
- “Maintaining Login Control” on page 39
- “Controlling Access to Devices” on page 44
- “Controlling Access to Machine Resources” on page 46
- “Controlling Access to Files” on page 50
- “Controlling Network Access” on page 51
- “Reporting Security Problems” on page 55

Maintaining Physical Security

To control access to your system, you must maintain the physical security of your computing environment. For instance, a system that is logged in and left unattended is vulnerable to unauthorized access. An intruder can gain access to the operating system and to the network. The computer’s surroundings and the computer hardware should be physically protected from unauthorized access.

You can protect a SPARC system from unauthorized access to the hardware settings. Use the `eeprom` command to require a password to access the PROM. For more information, see “How to Require a Password for Hardware Access” on page 72.
Maintaining Login Control

You also must prevent unauthorized logins to a system or the network, which you can do through password assignment and login control. All accounts on a system should have a password. A password is a simple authentication mechanism. An account without a password makes your entire network accessible to an intruder who guesses a user name. A strong password algorithm protects against brute force attacks.

When a user logs in to a system, the `login` command checks the appropriate name service or directory service database according to the information that is listed in the `/etc/nsswitch.conf` file. This file can include the following entries:

- `files` – Designates the `/etc` files on the local system
- `ldap` – Designates the LDAP directory service on the LDAP server
- `nisd` – Designates the NIS database on the NIS master server
- `nisp` – Designates the NIS+ database on the NIS+ root server

For a description of the `nsswitch.conf` file, see the `nsswitch.conf(4)` man page. For information about naming services and directory services, see the System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) or the System Administration Guide: Naming and Directory Services (NIS+).

The `login` command verifies the user name and password that were supplied by the user. If the user name is not in the password file, the `login` command denies access to the system. If the password is not correct for the user name that was specified, the `login` command denies access to the system. When the user supplies a valid user name and its corresponding password, the system grants the user access to the system.

PAM modules can streamline login to applications after a successful system login. For more information, see Chapter 16.

Sophisticated authentication and authorization mechanisms are available on Solaris systems. For a discussion of authentication and authorization mechanisms at the network level, see "Authentication and Authorization for Remote Access" on page 52.

Managing Password Information

When users log in to a system, they must supply both a user name and a password. Although logins are publicly known, passwords must be kept secret. Passwords should be known only to each user. You should ask your users to choose their passwords carefully. Users should change their passwords 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. You can also disable a user account by locking the password. For detailed information about administering passwords, see Chapter 4, "Managing User Accounts and Groups (Overview)," in System Administration Guide: Basic Administration and the `passwd(1)` man page.
Local Passwords

If your network uses local files to authenticate users, 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. The encrypted password itself is kept in a separate shadow file, /etc/shadow. This security measure 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 system, only superuser or an equivalent role can read the /etc/shadow file. You can use the passwd command to change a user’s password on a local system.

NIS and NIS+ Passwords

If your network uses NIS to authenticate users, password information is kept in the NIS password map. NIS does not support password aging. You can use the command passwd -r nis to change a user’s password that is stored in an NIS password map.

If your network uses NIS+ to authenticate users, password information is kept in the NIS+ database. Information in the NIS+ database can be protected by restricting access to authorized users only. You can use the passwd -r nisplus command to change a user’s password that is stored in an NIS+ database.

LDAP Passwords

The Solaris LDAP naming service stores password information and shadow information in the ou=people container of the LDAP directory tree. On the Solaris LDAP naming service client, you can use the passwd -r ldap command to change a user’s password. The LDAP naming service stores the password in the LDAP repository.

In the Solaris 10 release, password policy is enforced on the Sun Java™ System Directory Server. Specifically, the client’s pam_ldap module follows the password policy controls that are enforced on the Sun Java System Directory Server. For more information, see “LDAP Naming Services Security Model” in System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP).

Password Encryption

Strong password encryption provides an early barrier against attack. Solaris software provides four password encryption algorithms. The two MD5 algorithms and the Blowfish algorithm provide more robust password encryption than the UNIX algorithm.

Password Algorithm Identifiers

You specify the algorithms configuration for your site in the /etc/security/policy.conf file. In the policy.conf file, the algorithms are named by their identifier, as shown in the following table.
### TABLE 2–1 Password Encryption Algorithms

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Algorithm Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The MD5 algorithm that is compatible with MD5 algorithms on BSD and Linux systems.</td>
<td>crypt_bsdmd5(5)</td>
</tr>
<tr>
<td>2a</td>
<td>The Blowfish algorithm that is compatible with the Blowfish algorithm on BSD systems.</td>
<td>crypt_bsdbf(5)</td>
</tr>
<tr>
<td>md5</td>
<td>The Sun MD5 algorithm, which is considered stronger than the BSD and Linux version of MD5.</td>
<td>crypt_sunmd5(5)</td>
</tr>
<tr>
<td><strong>unix</strong></td>
<td>The traditional UNIX encryption algorithm. This algorithm is the default module in the policy.conf file.</td>
<td>crypt_unix(5)</td>
</tr>
</tbody>
</table>

### Algorithms Configuration in the policy.conf File

The following shows the default algorithms configuration in the policy.conf file:

```plaintext
# ...
# crypt(3c) Algorithms Configuration
#
# CRYPT_ALGORITHMS_ALLOW specifies the algorithms that are allowed to be used for new passwords. This is enforced only in crypt_gensalt(3c).
# CRYPT_ALGORITHMS_ALLOW=1,2a,md5

# To deprecate use of the traditional unix algorithm, uncomment below and change CRYPT_DEFAULT= to another algorithm. For example,
# CRYPT_DEFAULT=1 for BSD/Linux MD5.
# CRYPT_ALGORITHMS_DEPRECATE= __unix__

# The Solaris default is the traditional UNIX algorithm. This is not listed in crypt.conf(4) since it is internal to libc. The reserved name __unix__ is used to refer to it.
# CRYPT_DEFAULT= __unix__
...
```

When you change the value for CRYPT_DEFAULT, the passwords of new users are encrypted with the algorithm that is associated with the new value. When current users change their passwords, how their old password was encrypted affects which algorithm is used to encrypt the new password.

For example, assume that CRYPT_ALGORITHMS_ALLOW=1, 2a, md5 and CRYPT_DEFAULT=1. The following table shows which algorithm would be used to generate the encrypted password.
Controlling Access to a Computer System

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Initial Password</th>
<th>Changed Password</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>crypt_bsdmd5</td>
<td>Uses same</td>
<td>The 1 identifier is also the value of CRYPT_DEFAULT. The user’s password continues to be encrypted with the crypt_bsdmd5 algorithm.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm</td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>crypt_bsdbf</td>
<td>Uses same</td>
<td>The 2a identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_bsdbf algorithm.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm</td>
<td></td>
</tr>
<tr>
<td>md5</td>
<td>crypt_md5</td>
<td>Uses same</td>
<td>The md5 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_md5 algorithm.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm</td>
<td></td>
</tr>
<tr>
<td><strong>unix</strong></td>
<td>crypt_unix</td>
<td>Uses crypt_bsdmd5</td>
<td>The <strong>unix</strong> identifier is not in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the crypt_unix algorithm cannot be used. The new password is encrypted with the CRYPT_DEFAULT algorithm.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm</td>
<td></td>
</tr>
</tbody>
</table>

For more information on configuring the algorithm choices, see the policy.conf(4) man page. To specify password encryption algorithms, see “Changing the Password Algorithm (Task Map)” on page 65.

Special System Logins

Two common ways to access a system are by using a conventional user login, or by using the root login. In addition, a number of special system logins enable a user to run administrative commands without using the root account. As system administrator, you assign passwords to these login accounts.

The following table lists some system login accounts and their uses. The system logins perform special functions. Each login has its own group identification number (GID). Each login should have its own password, which should be divulged on a need-to-know basis.

TABLE 2–2 System Login Accounts and Their Uses

<table>
<thead>
<tr>
<th>Login Account</th>
<th>GID</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>0</td>
<td>Has almost no restrictions. 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. The root account, superuser, owns most of the Solaris commands.</td>
</tr>
<tr>
<td>daemon</td>
<td>1</td>
<td>Controls background processing.</td>
</tr>
<tr>
<td>bin</td>
<td>2</td>
<td>Owns some Solaris commands.</td>
</tr>
<tr>
<td>sys</td>
<td>3</td>
<td>Owns many system files.</td>
</tr>
</tbody>
</table>
TABLE 2–2 System Login Accounts and Their Uses (Continued)

<table>
<thead>
<tr>
<th>Login Account</th>
<th>GID</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>adm</td>
<td>4</td>
<td>Owns certain administrative files.</td>
</tr>
<tr>
<td>lp</td>
<td>71</td>
<td>Owns the object data files and spooled data files for the printer.</td>
</tr>
<tr>
<td>uucp</td>
<td>5</td>
<td>Owns the object data files and spooled data files for UUCP, the UNIX-to-UNIX copy program.</td>
</tr>
<tr>
<td>nuucp</td>
<td>9</td>
<td>Is used by remote systems to log in to the system and start file transfers.</td>
</tr>
</tbody>
</table>

Remote Logins

Remote logins offer a tempting avenue for intruders. The Solaris OS provides several commands to monitor, limit, and disable remote logins. For procedures, see "Securing Logins and Passwords (Task Map)" on page 58.

By default, remote logins cannot gain control or read certain system devices, such as the system mouse, keyboard, frame buffer, or audio device. For more information, see the logindevperm(4) man page.

Dial-Up Logins

When a computer can be accessed through a modem or a dial-up port, you can add an extra layer of security. You can require a dial-up password for users who access a system through a modem or dial-up port. The dial-up password is an additional password that a user must supply before being granted access to the system.

Only superuser or a role of equivalent capabilities 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 feature is to require a dial-up password to gain access to a gateway system. To set up dial-up passwords, see "How to Create a Dial-Up Password" on page 63.

Two files are involved in creating a dial-up password, /etc/dialups and /etc/d_passwd. The dialups file contains a list of ports that require a dial-up password. The d_passwd file contains a list of shell programs that require an encrypted password as the additional dial-up password. The information in these two files is processed as follows:

- 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 default password is 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 are not prompted for a dial-up password.
Controlling Access to Devices

Peripheral devices that are attached to a computer system pose a security risk. Microphones can pick up conversations and transmit them to remote systems. CD-ROMs can leave their information behind for reading by the next user of the CD-ROM device. Printers can be accessed remotely. Devices that are integral to the system can also present security issues. For example, network interfaces such as hme0 are considered integral devices.

Solaris software provides two methods of controlling access to devices. Device policy restricts or prevents access to devices that are integral to the system. Device policy is enforced in the kernel. Device allocation restricts or prevents access to peripheral devices. Device allocation is enforced at user allocation time.

Device policy uses privileges to protect selected devices in the kernel. For example, the device policy on network interfaces such as hme requires all privileges for reading or writing.

Device allocation uses authorizations to protect peripheral devices, such as printers or microphones. By default, device allocation is not enabled. Once enabled, device allocation can be configured to prevent the use of a device or to require authorization for access to the device. When a device is allocated for use, no other user can access the device until the current user deallocates it.

A Solaris system can be configured in several areas to control access to devices:

- **Set device policy** – In the Solaris 10 release, you can require that the process that is accessing a particular device be running with a set of privileges. Processes without those privileges cannot use the device. At boot time, Solaris software configures device policy. Third-party drivers can be configured with device policy during installation. After installation, you, as the system administrator can add device policy to a device.

- **Make devices allocatable** – When you enable device allocation, you can restrict the use of a device to one user at a time. You can further require that the user fulfill some security requirements. For example, you can require that the user be authorized to use the device.

- **Prevent devices from being used** – You can prevent the use of a device, such as a microphone, by any user on a computer system. A computer kiosk might be a good candidate for making certain devices unavailable for use.

- **Confine a device to a particular zone** – You can assign the use of a device to a non-global zone. For more information, see “Device Use in Non-Global Zones” in System Administration Guide: Solaris Containers-Resource Management and Solaris Zones. For a more general discussion of devices and zones, see “Configured Devices in Zones” in System Administration Guide: Solaris Containers-Resource Management and Solaris Zones.
Device Policy (Overview)

The device policy mechanism enables you to specify that processes that open a device require certain privileges. Devices that are protected by device policy can only be accessed by processes that are running with the privileges that the device policy specifies. The Solaris OS provides default device policy. For example, network interfaces such as hme0 require that the processes that access the interface be running with the net rawaccess privilege. The requirement is enforced in the kernel.

In earlier Solaris OS releases, device nodes were protected by file permissions alone. For example, devices owned by group sys could be opened only by members of group sys. In the Solaris 10 release, file permissions do not predict who can open a device. Instead, devices are protected with file permissions and with device policy. For example, the /dev/1p file has 666 permissions. However, the device can only be opened by a process with the appropriate privileges.

The configuration of device policy can be audited. The AUE_MODDEVPLCY audit event records changes in device policy.

For more information about device policy, see the following:

- “Configuring Device Policy (Task Map)” on page 76
- “Device Policy Commands” on page 89
- “Privileges and Devices” on page 184

Device Allocation (Overview)

The device allocation mechanism enables you to restrict access to a peripheral device, such as a CD-ROM. You manage the mechanism locally. If device allocation is not enabled, peripheral devices are protected only by file permissions. For example, by default, peripheral devices are available for the following uses:

- Any user can read and write to a diskette or CD-ROM.
- Any user can attach a microphone.
- Any user can access an attached printer.

Device allocation can restrict a device to authorized users. Device allocation can also prevent a device from being accessed at all. A user who allocates a device has exclusive use of that device until the user deallocates the device. When a device is deallocated, device-clean scripts erase any leftover data. You can write a device-clean script to purge information from devices that do not have a script. For an example, see “Writing New Device-Clean Scripts” on page 94.

Attempts to allocate a device, deallocate a device, and list allocatable devices can be audited. The audit events are part of the ot audit class.

For more information on device allocation, see the following:

- “Managing Device Allocation (Task Map)” on page 79
- “Device Allocation” on page 89
Controlling Access to Machine Resources

As system administrator, you can control and monitor system activity. You can set limits on who can use what resources. You can log resource use, and you can monitor who is using the resources. You can also set up your machines to minimize improper use of resources.

Limiting and Monitoring Superuser

Your system requires a root password for superuser access. In the default configuration, a user cannot remotely log in to a system as root. When logging in remotely, a user must log in with the user’s username and then use the su command to become root. You can monitor who has been using the su command, especially those users who are trying to gain superuser access. For procedures that monitor superuser and limit access to superuser, see “Monitoring and Restricting Superuser (Task Map)” on page 69.

Configuring Role-Based Access Control to Replace Superuser

Role-based access control, or RBAC, is designed to limit the capabilities of superuser. Superuser, the root user, has access to every resource in the system. With RBAC, you can replace root with a set of roles with discrete powers. For example, you can set up one role to handle user account creation, and another role to handle system file modification. When you have established a role to handle a function or set of functions, you can remove those functions from root’s capabilities.

Each role requires that a known user log in with their user name and password. After logging in, the user then assumes the role with a specific role password. As a consequence, someone who learns the root password has limited ability to damage your system. For more on RBAC, see “Role-Based Access Control (Overview)” on page 169.

Preventing Unintentional Misuse of Machine Resources

You can prevent you and your users from making unintentional errors in the following ways:

- You can keep from running a Trojan horse by correctly setting the PATH variable.
- You can assign a restricted shell to users. A restricted shell prevents user error by steering users to those parts of the system that the users need for their jobs. In fact, through careful setup, you can ensure that users access only those parts of the system that help the users work efficiently.
You can set restrictive permissions on files that users do not need to access.

**Setting the PATH Variable**

You should take care to correctly set the PATH variable. Otherwise, you can accidentally run a program that was introduced by someone else. The intruding program can corrupt your data or harm 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 the substitute program. Such a script would look just like the regular su command. Because the script removes itself after execution, you would have little evidence to show that you have actually run a Trojan horse.

The PATH variable is automatically set at login time. The path is set through the startup files: .login, .profile, and .cshrc. When you set up the user search path so that the current directory (.) comes last, you are protected from running this type of Trojan horse. The PATH variable for superuser should not include the current directory at all.

The Automated Security Enhancement Tool (ASET) examines the startup files to ensure that the PATH variable is set up correctly. ASET also ensures that the PATH variable does not contain a dot (.) entry.

**Assigning a Restricted Shell to Users**

The standard shell allows a user to open files, execute commands, and so on. The restricted shell limits the ability of a user to change directories and to execute commands. The restricted shell is invoked with the /usr/lib/rsh command. Note that the restricted shell is not the remote shell, which is /usr/sbin/rsh. The restricted shell differs from the standard shell in the following ways:

- The user is limited to the user’s home directory, so the user cannot use the cd command to change directories. Therefore, the user cannot browse system files.
- The user cannot change the PATH variable, so the user can use only commands in the path that is set by the system administrator. The user also cannot execute commands or scripts by using a complete path name.
- The user cannot redirect output with > or >>.

The restricted shell enables you to limit a user’s ability to stray into system files. The shell creates a limited environment for a user who needs to perform specific tasks. The restricted shell is not completely secure, however, and is only intended to keep unskilled users from inadvertently doing damage.

For information about the restricted shell, use the man -s1 rsh command to see the rsh(1M) man page.

A more secure alternative to the restricted shell is the ssh command in Solaris Secure Shell. Solaris Secure Shell enables users to securely access a remote host over an unsecured network. For information about using Solaris Secure Shell, see Chapter 19.
Restricting Access to Data in Files

Because the Solaris OS is a multiuser environment, file system security is the most basic security risk on a system. You can use traditional UNIX file protections to protect your files. You can also use the more secure access control lists (ACLs).

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 6 discusses how to set file permissions.

Restricting setuid Executable Files

Executable files can be security risks. Many executable programs have to be run as root, that is, as superuser, to work properly. These setuid programs run with the user ID set to 0. Anyone who is running these programs runs the programs with the root ID. A program that runs with the root ID creates a potential security problem if the program was not written with security in mind.

Except for the executables that Sun ships with the setuid bit set to root, you should disallow the use of setuid programs. If you cannot disallow the use of setuid programs, then you should at least restrict their use. Secure administration requires few setuid programs.

For more information, see “Preventing Executable Files From Compromising Security” on page 128. For procedures, see “Protecting Against Programs With Security Risk (Task Map)” on page 140.

Using the Automated Security Enhancement Tool

The ASET security package provides automated administration tools that enable you to control and monitor your system’s security. ASET provides three security levels: low, medium, and high. You specify an ASET security level. At each higher level, ASET’s file-control functions increase to reduce file access and tighten your system’s security. For more information, see Chapter 7.

Using the Solaris Security Toolkit

While ASET can be used to make a small number of security changes to a system, the Solaris Security Toolkit provides a flexible and extensible mechanism to minimize, harden, and secure a Solaris system. The Solaris Security Toolkit, informally known as the JASS toolkit, is a tool that enables the user to perform security modifications to a system. The tool can provide a report on the security status of a system. The tool also has the ability to undo previous runs of the tool. The JASS toolkit can be downloaded from the Sun web site, http://wwws.sun.com/security/jass. The web site contains pointers to online documentation.

Using Solaris Resource Management Features

Solaris software provides sophisticated resource management features. Using these features, you can allocate, schedule, monitor, and cap resource use by applications in a server consolidation environment. The resource controls framework enables you to set constraints on system resources that are consumed by processes. Such constraints help to prevent denial-of-service attacks by a script that attempts to flood a system’s resources.

With Solaris resource management features, you can designate resources for particular projects. You can also dynamically adjust the resources that are available. For more information, see Part I, “Resource Management,” in System Administration Guide: Solaris Containers-Resource Management and Solaris Zones.

Using Solaris Zones

Solaris zones provide an application execution environment in which processes are isolated from the rest of the system within a single instance of the Solaris OS. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Even a process running with superuser capabilities cannot view or affect activity in other zones.

Solaris zones are ideal for environments that place several applications on a single server. For more information, see Part II, “Zones,” in System Administration Guide: Solaris Containers-Resource Management and Solaris Zones.

Monitoring Use of Machine Resources

As a system administrator, you need to monitor system activity. You need to be aware of all aspects of your machines, including the following:

- What is the normal load?
- Who has access to the system?
- When do individuals access the system?
- What programs normally run on 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 a breach in security is suspected. For more information on the auditing service, see Chapter 27.

Monitoring File Integrity

As a system administrator, you need assurance that the files that were installed on the systems that you administer have not changed in unexpected ways. In large installations, a comparison and reporting tool about the software stack on each of your systems enables you to track your systems. The Basic Audit Reporting Tool (BART) enables you to comprehensively validate systems by
performing file-level checks of one or more systems over time. Changes in a BART manifest across systems, or for one system over time, can validate the integrity of your systems. BART provides manifest creation, manifest comparison, and rules for scripting reports. For more information, see Chapter 5.

Controlling Access to Files

The Solaris OS is a multiuser environment. In a multiuser environment, all the users who are logged in to a system can read files that belong to other users. With the appropriate file permissions, users can also use files that belong to other users. For more discussion, see Chapter 6. For step-by-step instructions on setting appropriate permissions on files, see “Protecting Files (Task Map)” on page 128.

Protecting Files With Encryption

You can keep a file secure by making the file inaccessible to other users. For example, a file with permissions of 600 cannot be read except by its owner and by superuser. A directory with permissions of 700 is similarly inaccessible. However, someone who guesses your password or who discovers the root password can access that file. Also, the otherwise inaccessible file is preserved on a backup tape every time that the system files are backed up to offline media.

The Solaris Cryptographic Framework provides digest, mac, and encrypt commands to protect files. For more information, see Chapter 13.

Using Access Control Lists

ACLs, pronounced “ackkls,” can provide greater control over file permissions. You add ACLs when traditional UNIX file protections are not sufficient. Traditional UNIX file protections provide read, write, and execute permissions for the three user classes: owner, group, and other. An ACL provides finer-grained file security. ACLs enable you to define the following file permissions:

- Owner file permissions
- File permissions for the owner’s group
- File permissions for other users who are outside the owner’s group
- File permissions for specific users
- File permissions for specific groups
- Default permissions for each of the previous categories

For more information about using ACLs, see “Using Access Control Lists to Protect Files” on page 125.
Sharing Files Across Machines

A network file server can control which files are available for sharing. A network file server can also control which clients have access to the files, and what type of access is permitted for those clients. In general, the file server can grant read-write access 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.

The `/etc/dfs/dfstab` file on the file server lists the file systems that the server makes available to clients on the network. For more information about sharing file systems, see “Automatic File-System Sharing” in System Administration Guide: Network Services.

Restricting root Access to Shared Files

In general, superuser is not allowed root access to file systems that are shared across the network. The NFS system prevents root access to mounted file systems by changing the user of the requester to the user nobody with the user ID 60001. The access rights of user nobody are the same as those access rights that are given to the public. The user nobody has the access rights of a user without credentials. 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 capabilities on a shared file system on a per-host basis. To grant these privileges, use the `root=hostname` option to the `share` command. You should use this option with care. For a discussion of security options with NFS, see Chapter 6, “Accessing Network File Systems (Reference),” in System Administration Guide: Network Services.

Controlling Network Access

Computers are often part of a configuration of computers. This configuration is called a network. A network allows connected computers to exchange information. Networked computers can access data and other resources from other computers on the network. Computer networks create a powerful and sophisticated computing environment. However, networks also complicate computer security.

For example, within a network of computers, individual machines allow the sharing of information. Unauthorized access is a security risk. Because many people have access to a network, unauthorized access is more likely, especially through user error. A poor use of passwords can also allow unauthorized access.

Network Security Mechanisms

Network security is usually based on limiting or blocking operations from remote systems. The following figure describes the security restrictions that you can impose on remote operations.
The firewall restricts the types of remote operations that the systems at a particular site can perform with systems that are outside the firewall.

Remote systems use authentication to restrict access to specific users.

Remote systems use authorization to restrict authenticated users from performing operations on their file systems.

**FIGURE 2–1 Security Restrictions for Remote Operations**

### Authentication and Authorization for Remote Access

*Authentication* is a way to restrict access to specific users when these users access a remote system. Authentication can be set up at both the system level and the network level. After a user has gained access to a remote system, *authorization* is a way to restrict operations that the user can perform. The following table lists the services that provide authentication and authorization.

**TABLE 2–3 Authentication and Authorization Services for Remote Access**

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPsec</td>
<td>IPsec provides host-based and certificate-based authentication and network traffic encryption.</td>
<td>Chapter 19, &quot;IP Security Architecture (Overview),” in <em>System Administration Guide: IP Services</em></td>
</tr>
</tbody>
</table>
### Table 2-3 Authentication and Authorization Services for Remote Access (Continued)

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kerberos</td>
<td>Kerberos uses encryption to authenticate and authorize a user who is logging in to the system.</td>
<td>For an example, see “How the Kerberos Service Works” on page 340.</td>
</tr>
<tr>
<td>LDAP and NIS+</td>
<td>The LDAP directory service and the NIS+ name service can provide both authentication and authorization at the network level.</td>
<td>System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) and System Administration Guide: Naming and Directory Services (NIS+)</td>
</tr>
<tr>
<td>Remote login commands</td>
<td>The remote login commands enable users to log in to a remote system over the network and use its resources. Some of the remote login commands are rlogin, rcp, and ftp. If you are a “trusted host,” authentication is automatic. Otherwise, you are asked to authenticate yourself.</td>
<td>Chapter 29, “Accessing Remote Systems (Tasks),” in System Administration Guide: Network Services</td>
</tr>
<tr>
<td>SASL</td>
<td>The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. Plugins enable you to choose an appropriate authentication protocol.</td>
<td>“SASL (Overview)” on page 299</td>
</tr>
<tr>
<td>Secure RPC</td>
<td>Secure RPC improves the security of network environments by authenticating users who make requests on remote machines. You can use either the UNIX, DES, or Kerberos authentication system for Secure RPC. Secure RPC can also be used to provide additional security in an NFS environment. An NFS environment with secure RPC is called Secure NFS. Secure NFS uses Diffie-Hellman authentication for public keys.</td>
<td>“Overview of Secure RPC” on page 277 “NFS Services and Secure RPC” on page 277</td>
</tr>
<tr>
<td>Solaris Secure Shell</td>
<td>Solaris Secure Shell encrypts network traffic over an unsecured network. Solaris Secure Shell provides authentication by the use of passwords, public keys, or both. Solaris Secure Shell uses RSA and DSA authentication for public keys.</td>
<td>“Solaris Secure Shell (Overview)” on page 303</td>
</tr>
</tbody>
</table>

A possible substitute for Secure RPC is the Solaris privileged port mechanism. A privileged port is assigned a port number less than 1024. After a client system has authenticated the client’s credential, the client builds a connection to the server by using the privileged port. The server then verifies the client credential by examining the connection’s port number.

Clients that are not running Solaris software might be unable to communicate by using the privileged port. If the clients cannot communicate over the port, you see an error message that is similar to the following:

"Weak Authentication
NFS request from unprivileged port"
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 internal network treats every other network as untrusted. You should consider this setup as mandatory between your internal network and any external networks, such as the Internet, with which you communicate.

A firewall acts as a gateway and as a barrier. A firewall acts as a gateway that passes data between the networks. A firewall acts as a barrier that 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 first log in to the firewall system before being granted access to a host on the internal network.

A firewall can also be useful between some internal networks. For example, you can set up a firewall or a secure gateway computer to restrict the transfer of packets. The gateway can forbid packet exchange between two networks, unless the gateway computer is the source address 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 allow packets for the telnet or the rlogin command. ASET, when run at high security, disables the forwarding of Internet Protocol (IP) packets.

In addition, all electronic mail that is sent from the internal network is first sent to the firewall system. The firewall then transfers the mail to a host on an external network. The firewall system also receives all incoming electronic mail, and distributes the mail to the hosts on the internal network.

Caution – A firewall prevents unauthorized users from accessing the 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 a host from which a user can log in without being required to supply a password. A firewall system should not share any of its file systems, or mount any file systems from other servers.

The following technologies can be used to harden a system into a firewall:

- ASET enforces high security on a firewall system, as described in Chapter 7.
- The Solaris Security Toolkit, informally known as the JASS toolkit, can harden a Solaris system into a firewall. The toolkit can be downloaded from the Sun web site, http://wwws.sun.com/security/jass.
Encryption and Firewall Systems

Most local area networks transmit data between computers in blocks that are called packets. Through a procedure that is called packet smashing, unauthorized users from outside the network can corrupt or destroy data.

Packet smashing involves capturing the packets before the packets reach their destination. The intruder then injects arbitrary data into the contents, and sends 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 that all gateways on the network are protected.

The most dangerous attacks affect the integrity of the data. Such attacks involve changing the contents of the packets or impersonating a user. Attacks that involve eavesdropping do not compromise data integrity. An eavesdropper records conversations for later replay. An eavesdropper does not impersonate a user. Although eavesdropping attacks do not attack data integrity, the attacks do affect privacy. You can protect the privacy of sensitive information by encrypting data that goes over the network.

- To encrypt remote operations over an insecure network, see Chapter 18.
- To encrypt and authenticate data across a network, see Chapter 20.

Reporting Security Problems

If you experience a suspected security breach, you can contact the Computer Emergency Response Team/Coordination Center (CERT/CC). CERT/CC is a Defense Advanced Research Projects Agency (DARPA) funded project that is located at the Software Engineering Institute at Carnegie Mellon University. This agency can assist you with any security problems you are having. This agency can also direct you to other Computer Emergency Response Teams that might be more appropriate for 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.
Controlling Access to Systems (Tasks)

This chapter describes the procedures for controlling who can access Solaris systems. The following is a list of the information in this chapter.

- “Controlling System Access (Task Map)” on page 57
- “Securing Logins and Passwords (Task Map)” on page 58
- “Changing the Password Algorithm (Task Map)” on page 65
- “Monitoring and Restricting Superuser (Task Map)” on page 69
- “SPARC: Controlling Access to System Hardware (Task Map)” on page 72

For overview information about system security, see Chapter 2.

Controlling System Access (Task Map)

A computer is as secure as its weakest point of entry. The following task map shows the areas that you should monitor and secure.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor, permit, and deny user login</td>
<td>Monitors unusual login activity. Prevents logins temporarily. Manages dial-up logins.</td>
<td>“Securing Logins and Passwords (Task Map)” on page 58</td>
</tr>
<tr>
<td>Provide strong password encryption</td>
<td>Specifies algorithms to encrypt user passwords. Installs additional algorithms.</td>
<td>“Changing the Password Algorithm (Task Map)” on page 65</td>
</tr>
<tr>
<td>Monitor and restrict superuser activities</td>
<td>Regularly monitors superuser activity. Prevents remote login by a root user.</td>
<td>“Monitoring and Restricting Superuser (Task Map)” on page 69</td>
</tr>
<tr>
<td>Prevent access to hardware settings</td>
<td>Keeps ordinary users away from the PROM.</td>
<td>“SPARC: Controlling Access to System Hardware (Task Map)” on page 72</td>
</tr>
</tbody>
</table>
Securing Logins and Passwords (Task Map)

The following task map points to procedures that monitor user logins and that disable user logins.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display a user’s login status</td>
<td>Lists extensive information about a user’s login account, such as full name and password aging information.</td>
<td>“How to Display a User’s Login Status” on page 58</td>
</tr>
<tr>
<td>Find users who do not have passwords</td>
<td>Finds only those users whose accounts do not require a password.</td>
<td>“How to Display Users Without Passwords” on page 59</td>
</tr>
<tr>
<td>Disable logins temporarily</td>
<td>Denies user logins to a machine as part of system shutdown or routine maintenance.</td>
<td>“How to Temporarily Disable User Logins” on page 60</td>
</tr>
<tr>
<td>Save failed login attempts</td>
<td>Creates a log of users who failed to provide the correct password after five attempts.</td>
<td>“How to Monitor Failed Login Attempts” on page 61</td>
</tr>
<tr>
<td>Save all failed login attempts</td>
<td>Creates a log of failed attempts to log in.</td>
<td>“How to Monitor All Failed Login Attempts” on page 62</td>
</tr>
<tr>
<td>Create a dial-up password</td>
<td>Requires an additional password for users who log in remotely through a modem or dial-up port.</td>
<td>“How to Create a Dial-Up Password” on page 63</td>
</tr>
<tr>
<td>Disable dial-up logins temporarily</td>
<td>Prevents users from dialing in remotely through a modem or port.</td>
<td>“How to Temporarily Disable Dial-Up Logins” on page 65</td>
</tr>
</tbody>
</table>

Securing Logins and Passwords

You can limit remote logins and require users to have passwords. You can also monitor failed access attempts and disable logins temporarily.

How to Display a User’s Login Status

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Display a user’s login status by using the `logins` command.**
   ```bash
   # logins -x -l username
   -x  Displays an extended set of login status information.
   -l username  Displays the login status for the specified user. The variable `username` is a user’s login name. Multiple login names must be specified in a comma-separated list.
   ```
The `logins` command uses the appropriate password database to obtain a user’s login status. The database can be the local `/etc/passwd` file, or a password database for the name service. For more information, see the `logins(1M)` man page.

**Example 3-1**  
**Displaying a User’s Login Status**

In the following example, the login status for the user `rimmer` is displayed.

```
# logins -x -l rimmer
rimmer  500  staff  10  Annalee J. Rimmer
/export/home/rimmer
/bin/sh
PS 010103 10 7 -1
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rimmer</td>
<td>Identifies the user’s login name.</td>
</tr>
<tr>
<td>500</td>
<td>Identifies the user ID (UID).</td>
</tr>
<tr>
<td>staff</td>
<td>Identifies the user’s primary group.</td>
</tr>
<tr>
<td>10</td>
<td>Identifies the group ID (GID).</td>
</tr>
<tr>
<td>Annalee J. Rimmer</td>
<td>Identifies the comment.</td>
</tr>
<tr>
<td>/export/home/rimmer</td>
<td>Identifies the user’s home directory.</td>
</tr>
<tr>
<td>/bin/sh</td>
<td>Identifies the login shell.</td>
</tr>
<tr>
<td>PS 010103</td>
<td>Specifies the password aging information:</td>
</tr>
<tr>
<td>10 7 -1</td>
<td>- Last date that the password was changed</td>
</tr>
<tr>
<td></td>
<td>- Number of days that are required between changes</td>
</tr>
<tr>
<td></td>
<td>- Number of days before a change is required</td>
</tr>
<tr>
<td></td>
<td>- Warning period</td>
</tr>
</tbody>
</table>

**How to Display Users Without Passwords**

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Display all users who have no passwords by using the `logins` command.**

   ```
   # logins -p
   ```

   The `-p` option displays a list of users with no passwords. The `logins` command uses the password database from the local system unless a name service is enabled.
Example 3–2 Displaying Users Without Passwords

In the following example, the user pmorph does not have a password.

```
# logins -p
pmorph      501    other    1    Polly Morph
#
```

How to Temporarily Disable User Logins

Temporarily disable user logins during system shutdown or routine maintenance. Superuser logins are not affected. For more information, see the nologin(4) man page.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Create the /etc/nologin file in a text editor.

```
# vi /etc/nologin
```

3 Include a message about system availability.

4 Close and save the file.

Example 3–3 Disabling User Logins

In this example, users are notified of system unavailability.

```
# vi /etc/nologin
(Add system message here)

# cat /etc/nologin
***No logins permitted.***

***The system will be unavailable until 12 noon.***
```

You can also bring the system to run level 0, single-user mode, to disable logins. For information on bringing the system to single-user mode, see Chapter 9, “Shutting Down a System (Tasks),” in System Administration Guide: Basic Administration.
How to Monitor Failed Login Attempts

This procedure captures failed login attempts from terminal windows. This procedure does not capture failed logins from a CDE or GNOME login attempt.

1 Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Create the loginlog file in the /var/adm directory.
   # touch /var/adm/loginlog

3 Set read-and-write permissions for root user 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 Verify that the log works.
For example, log in to the system five times with the wrong password. Then, display the /var/adm/loginlog file.
   # more /var/adm/loginlog
   jdoe:/dev/pts/2:Tue Nov 4 10:21:10 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:21 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:30 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:40 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:49 2003
   #

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, no failed attempts are logged.

A growing loginlog file can indicate an attempt to break into the computer system. Therefore, check and clear the contents of this file regularly. For more information, see the loginlog(4) man page.
**How to Monitor All Failed Login Attempts**

This procedure captures in a syslog file all failed login attempts.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Set up the /etc/default/login file with the desired values for SYSLOG and SYSLOG_FAILED_LOGINS**
   Edit the /etc/default/login file to change the entry. Make sure that SYSLOG=Yes is uncommented.

   ```
   # grep SYSLOG /etc/default/login
   # SYSLOG determines whether the syslog(3) LOG_AUTH facility
   # should be used
   SYSLOG=YES
   ...
   SYSLOG_FAILED_LOGINS=0
   #
   ```

3. **Create a file with the correct permissions to hold the logging information.**
   a. Create the authlog file in the /var/adm directory.
      ```
      # touch /var/adm/authlog
      ```
   b. Set read-and-write permissions for root user on the authlog file.
      ```
      # chmod 600 /var/adm/authlog
      ```
   c. Change group membership to sys on the authlog file.
      ```
      # chgrp sys /var/adm/authlog
      ```

4. **Edit the syslog.conf file to log failed password attempts.**
   The failures should be sent to the authlog file.

   a. Type the following entry into the syslog.conf file.
      Fields on the same line in syslog.conf are separated by tabs.
      ```
      auth notice <Press Tab> /var/adm/authlog
      ```
   b. Refresh the configuration information for the syslog daemon.
      ```
      # svcadm refresh system/system-log
      ```
5 Verify that the log works.
   For example, as an ordinary user, log in to the system with the wrong password. Then, in the Primary Administrator role or as superuser, display the `/var/adm/authlog` file.
   
   ```
   # more /var/adm/authlog
   Nov 4 14:46:11 example1 login: [ID 143248 auth.notice]  
   Login failure on /dev/pts/8 from example2, stacey
   #
   ```

6 Monitor the `/var/adm/authlog` file on a regular basis.

**Example 3–4** Logging Access Attempts After Three Login Failures

Follow the preceding procedure, except set the value of `SYSLOG_FAILED_LOGINS` to 3 in the `/etc/default/login` file.

**Example 3–5** Closing Connection After Three Login Failures

Uncomment the `RETRIES` entry in the `/etc/default/login` file, then set the value of `RETRIES` to 3. Your edits take effect immediately. After three login retries in one session, the system closes the connection.

▼ How to Create a Dial-Up Password

Caution – When you first establish a dial-up password, be sure to remain logged in to at least one port. Test the password on a different port. If you log off to test the new password, you might not be able to log back in. If you are still logged in to another port, you can go back and fix your mistake.

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Create an `/etc/dialups` file that contains a list of serial devices.
   Include all the ports that are being protected with dial-up passwords. The `/etc/dialups` file should appear similar to the following:
   ```
   /dev/term/a
   /dev/term/b
   /dev/term/c
   ```
Create an `/etc/d_passwd` file that contains the login programs that you are requiring to have a 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 appear similar to the following:

```
/usr/lib/uucp/uucico:encrypted-password:
/usr/bin/csh:encrypted-password:
/usr/bin/ksh:encrypted-password:
/usr/bin/sh:encrypted-password:
```

Later in the procedure, you are going to add the encrypted password for each login program.

Set ownership to root on the two files.

```bash
# chown root /etc/dialups /etc/d_passwd
```

Set group ownership to root on the two files.

```bash
# chgrp root /etc/dialups /etc/d_passwd
```

Set read-and-write permissions for root on the two files.

```bash
# chmod 600 /etc/dialups /etc/d_passwd
```

Create the encrypted passwords.

a. Create a temporary user.

```bash
# useradd username
```

b. Create a password for the temporary user.

```bash
# passwd username
New Password: <Type password>
Re-enter new Password: <Retype password>
```

```bash
passwd: password successfully changed for username
```

c. Capture the encrypted password.

```bash
# grep username /etc/shadow > username.temp
```

d. Edit the `username.temp` file.

Delete all fields except the encrypted password. The second field holds the encrypted password.

For example, in the following line, the encrypted password is `U9gp9SyA/31Sk`.

```
temp:U9gp9SyA/31Sk:7967:7988:
```

e. Delete the temporary user.

```bash
# userdel username
```
Copy the encrypted password from **username.temp** file into the **/etc/d_passwd** file.

You can create a different password for each login shell. Alternatively, use the same password for each login shell.

Inform your dial-up users of the password.

You should ensure that your means of informing the users cannot be tampered with.

**How to Temporarily Disable Dial-Up Logins**

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Put the following single-line entry into the **/etc/d_passwd** file:**
   
   ```
   /usr/bin/sh:*:
   ```

**Changing the Password Algorithm (Task Map)**

The following task map points to procedures to administer password algorithms.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide strong password encryption</td>
<td>“How to Specify an Algorithm for Password Encryption” on page 66</td>
</tr>
<tr>
<td>Provide strong password encryption with a name service</td>
<td>“How to Specify a New Password Algorithm for an NIS Domain” on page 67</td>
</tr>
<tr>
<td></td>
<td>“How to Specify a New Password Algorithm for an NIS+ Domain” on page 67</td>
</tr>
<tr>
<td></td>
<td>“How to Specify a New Password Algorithm for an LDAP Domain” on page 67</td>
</tr>
<tr>
<td>Add new password encryption module</td>
<td>“How to Install a Password Encryption Module From a Third Party” on page 68</td>
</tr>
</tbody>
</table>
Changing the Default Algorithm for Password Encryption

By default, user passwords are encrypted with the crypt_unix algorithm. You can use a stronger encryption algorithm, such as MD5 or Blowfish, by changing the default password encryption algorithm.

How to Specify an Algorithm for Password Encryption

In this procedure, the BSD-Linux version of the MD5 algorithm is the default encryption algorithm that is used when users change their passwords. This algorithm is suitable for a mixed network of machines that run the Solaris, BSD, and Linux versions of UNIX. For a list of password encryption algorithms and algorithm identifiers, see Table 2–1.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Specify the identifier for your chosen encryption algorithm.

Type the identifier as the value for the CRYPT_DEFAULT variable in the /etc/security/policy.conf file.

You might want to comment the file to explain your choice.

```
# cat /etc/security/policy.conf
...
CRYPT_ALGORITHMS_ALLOW=1,2a,md5
# # Use the version of MD5 that works with Linux and BSD systems.
# Passwords previously encrypted with __unix__ will be encrypted with MD5
# when users change their passwords.
#
# CRYPT_DEFAULT=__unix__
CRYPT_DEFAULT=1
```

In this example, the algorithms configuration ensures that the weakest algorithm, crypt_unix, is never used to encrypt a password. Users whose passwords were encrypted with the crypt_unix module get a crypt_bsdmd5-encrypted password when they change their passwords.

For more information on configuring the algorithm choices, see the policy.conf(4) man page.

Example 3–6 Using the Blowfish Algorithm for Password Encryption

In this example, the identifier for the Blowfish algorithm, 2a, is specified as the value for the CRYPT_DEFAULT variable in the policy.conf file:
This configuration is compatible with BSD systems that use the Blowfish algorithm.

**How to Specify a New Password Algorithm for an NIS Domain**

When users in an NIS domain change their passwords, the NIS client consults its local algorithms configuration in the `/etc/security/policy.conf` file. The NIS client machine encrypts the password.

1. Specify the password encryption algorithm in the `/etc/security/policy.conf` file on the NIS client.

2. Copy the modified `/etc/security/policy.conf` file to every client machine in the NIS domain.

3. To minimize confusion, copy the modified `/etc/security/policy.conf` file to the NIS root server and to the slave servers.

**How to Specify a New Password Algorithm for an NIS+ Domain**

When users in an NIS+ domain change their passwords, the NIS+ name service consults the algorithms configuration in the `/etc/security/policy.conf` file on the NIS+ master. The NIS+ master, which is running the `rpc.nispasswd` daemon, creates the encrypted password.

1. Specify the password encryption algorithm in the `/etc/security/policy.conf` file on the NIS+ master.

2. To minimize confusion, copy the NIS+ master’s `/etc/security/policy.conf` file to every host in the NIS+ domain.

**How to Specify a New Password Algorithm for an LDAP Domain**

When the LDAP client is properly configured, the LDAP client can use the new password algorithms. The LDAP client behaves just as an NIS client behaves.

1. Specify a password encryption algorithm in the `/etc/security/policy.conf` file on the LDAP client.
2 Copy the modified policy.conf file to every client machine in the LDAP domain.

3 Ensure that the client’s /etc/pam.conf file does not use a pam_ldap module.

Ensure that a comment sign (#) precedes entries that include pam_ldap.so.1. Also, do not use the new server_policy option with the pam_authtok_store.so.1 module.

The PAM entries in the client’s pam.conf file enable the password to be encrypted according to the local algorithms configuration. The PAM entries also enable the password to be authenticated.

When users in the LDAP domain change their passwords, the LDAP client consults its local algorithms configuration in the /etc/security/policy.conf file. The LDAP client machine encrypts the password. Then, the client sends the encrypted password, with a {crypt} tag, to the server. The tag tells the server that the password is already encrypted. The password is then stored, as is, on the server. For authentication, the client retrieves the stored password from the server. The client then compares the stored password with the encrypted version that the client has just generated from the user’s typed password.

Note – To take advantage of password policy controls on the LDAP server, use the server_policy option with the pam_authtok_store entries in the pam.conf file. Passwords are then encrypted on the server by using the Sun Java™ System Directory Server’s cryptographic mechanism. For the procedure, see Chapter 11, “Setting Up Sun Java System Directory Server With LDAP Clients (Tasks),” in System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP).

▼ How to Install a Password Encryption Module From a Third Party

A third-party password encryption algorithm is typically delivered as a module in a software package. When you run the pkgadd command, scripts from the vendor should modify the /etc/security/crypt.conf file. You then modify the /etc/security/policy.conf file to include the new module and its identifier.

1 Add the software by using the pkgadd command.

For detailed instructions on how to add software, see “Adding or Removing a Software Package (pkgadd)” in System Administration Guide: Basic Administration.

2 Confirm that the new module and module identifier have been added.

Read the list of encryption algorithms in the /etc/security/crypt.conf file.

For example, the following lines show that a module that implements the crypt_rot13 algorithm has been installed.

```bash
# crypt.conf
#
md5 /usr/lib/security/$ISA/crypt_md5.so
```
rot13 /usr/lib/security/$ISA/crypt_rot13.so

# For *BSD - Linux compatibility
# 1 is MD5, 2a is Blowfish
1 /usr/lib/security/$ISA/crypt_bsdmd5.so
2a /usr/lib/security/$ISA/crypt_bsdbf.so

3 Add the identifier of the newly installed algorithm to the /etc/security/policy.conf file.
The following lines show excerpts from the policy.conf file that would need to be modified to add
the rot13 identifier.

# Copyright 1999-2002 Sun Microsystems, Inc. All rights reserved.
# ...
#ident "(#)policy.conf 1.6 02/06/07 SMI"
# ...
# crypt(3c) Algorithms Configuration
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,rot13
#CRYPT_ALGORITHMS_DEPRECATE=unix
CRYPT_DEFAULT=md5

In this example, the rot13 algorithm is used if the current password was encrypted with the
crypt_rot13 algorithm. New user passwords are encrypted with the crypt_sunmd5 algorithm. This
algorithms configuration works on Solaris-only networks.

**Monitoring and Restricting Superuser (Task Map)**

The following task map describes how to monitor and restrict the root user login.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor who is using the <code>su</code> command</td>
<td>Scans the su.log file on a regular basis.</td>
<td>&quot;How to Monitor Who Is Using the su Command&quot; on page 70</td>
</tr>
<tr>
<td>Display superuser activity on the console</td>
<td>Monitors superuser access attempts as the attempts occur.</td>
<td>&quot;How to Restrict and Monitor Superuser Logins&quot; on page 70</td>
</tr>
</tbody>
</table>

**Monitoring and Restricting Superuser**

An alternative to using the superuser account is to set up role-based access control. Role-based access
control is called RBAC. For overview information on RBAC, see “Role-Based Access Control
(Overview)” on page 169. To set up RBAC, see Chapter 9.
How to Monitor Who Is Using the su Command

The sulog file lists every use of the su command, not only the su attempts that are used to switch from user to superuser.

- Monitor the contents of the /var/adm/sulog file on a regular basis.

  # more /var/adm/sulog
  SU 12/20 16:26 + pts/0 stacey-root
  SU 12/20 16:27 + pts/0 stacey-root
  SU 12/20 16:27 + pts/0 stacey-root
  SU 01/12 11:11 + pts/0 root-rimmer
  SU 01/12 14:56 + pts/0 pmorph-root
  SU 01/12 14:57 + pts/0 pmorph-root

  The entries display the following information:
  - The date and time that the command was entered.
  - If the attempt was successful. A plus sign (+) indicates a successful attempt. A minus sign (-) indicates an unsuccessful attempt.
  - The port from which the command was issued.
  - The name of the user and the name of the switched identity.

  The su logging in this file is enabled by default through the following entry in the /etc/default/su file:

  SULOG=/var/adm/sulog

How to Restrict and Monitor Superuser Logins

This method immediately detects superuser attempts to access the local system.

1. View the CONSOLE entry in the /etc/default/login file.

   CONSOLE=/dev/console

   By default, the console device is set to /dev/console. With this setting, root can log in to the console. root cannot log in remotely.

2. Verify that root cannot log in remotely.

   From a remote system, try to log in as superuser.

   mach2 % rlogin -l root mach1
   Password: <Type root password of mach1>
   Not on system console
   Connection closed.
3 Monitor attempts to become superuser.
By default, attempts to become superuser are printed to the console by the SYSLOG utility.

a. Open a terminal console on your desktop.

b. In another window, use the su command to become superuser.

```
% su
Password: <Type root password>
#
```

A message is printed on the terminal console.

Sep 7 13:22:57 mach1 su: 'su root' succeeded for jdoe on /dev/pts/6

### Example 3–7 Logging Superuser Access Attempts

In this example, superuser attempts are not being logged by SYSLOG. Therefore, the administrator is logging those attempts by removing the comment from the #CONSOLE=/dev/console entry in the /etc/default/su file.

```
# CONSOLE determines whether attempts to su to root should be logged
# to the named device
#
CONSOLE=/dev/console
```

When a user attempts to become superuser, the attempt is printed on the terminal console.

```
SU 09/07 16:38 + pts/8 jdoe-root
```

### Troubleshooting

To become superuser from a remote system when the /etc/default/login file contains the default CONSOLE entry, users must first log in with their user name. After logging in with their user name, users then can use the su command to become superuser.

If the console displays an entry similar to Mar 16 16:20:36 mach1 login: ROOT LOGIN /dev/pts/14 FROM mach2.Example.COM, then the system is permitting remote root logins. To prevent remote superuser access, change the #CONSOLE=/dev/console entry to CONSOLE=/dev/console in the /etc/default/login file.
The following task map describes how to protect the PROM from unwanted access.

<table>
<thead>
<tr>
<th>Task Description For Instructions</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevent users from changing system hardware settings</td>
<td>Requires a password to modify PROM settings.</td>
<td>&quot;How to Require a Password for Hardware Access&quot; on page 72</td>
</tr>
<tr>
<td>Disable the abort sequence</td>
<td>Prevents users from accessing the PROM.</td>
<td>&quot;How to Disable a System’s Abort Sequence&quot; on page 73</td>
</tr>
</tbody>
</table>

**Controlling Access to System Hardware**

You can protect the physical machine by requiring a password to gain access to the hardware settings. You can also protect the machine by preventing a user from using the abort sequence to leave the windowing system.

**How to Require a Password for Hardware Access**

On an x86 system, the equivalent to protecting the PROM is to protect the BIOS. Refer to your machine’s manuals for how to protect the BIOS.

1. **Become superuser or assume a role that includes the Device Security profile, the Maintenance and Repair profile, or the System Administrator profile.**

   The System Administrator profile includes the Maintenance and Repair profile. To create a role that includes the System Administrator profile and to assign the role to a user, see “Configuring RBAC (Task Map)” on page 188.

2. **In a terminal window, type the PROM security mode.**

   ```
   # eeprom security-mode=command
   ```

   Changing PROM password:
   New password:  <Type password>
   Retype new password:  <Retype password>

   Choose the value command or full. For more details, see the eeprom(1M) man page.

   If, when you type the preceding command, you are not prompted for a PROM password, the system already has a PROM password.

3. **(Optional) To change the PROM password, type the following command:**

   ```
   # eeprom security-password= Press Return
   ```

   Changing PROM password:
New password: <Type password>
Retype new password: <Retype password>

The new PROM security mode and password are in effect immediately. However, they are most likely to be noticed at the next boot.

Caution – Do not forget the PROM password. The hardware is unusable without this password.

How to Disable a System’s Abort Sequence

Some server systems have a key switch. When the key switch is set in the secure position, the switch overrides the software keyboard abort settings. So, any changes that you make with the following procedure might not be implemented.

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Change the value of KEYBOARD_ABORT to disable.
   Comment out the enable line in the /etc/default/kbd file. Then, add a disable line:
   
   ```
   # cat /etc/default/kbd
   ...
   # KEYBOARD_ABORT affects the default behavior of the keyboard abort
   # sequence, see kbd(1) for details. The default value is "enable".
   # The optional value is "disable". Any other value is ignored.
   ...
   #KEYBOARD_ABORT=enable
   KEYBOARD_ABORT=disable
   ```

3 Update the keyboard defaults.
   
   ```
   # kbd -i
   ```
This chapter provides step-by-step instructions for protecting devices, in addition to a reference section. The following is a list of the information in this chapter.

- “Configuring Devices (Task Map)” on page 75
- “Configuring Device Policy (Task Map)” on page 76
- “Managing Device Allocation (Task Map)” on page 79
- “Allocating Devices (Task Map)” on page 84
- “Device Protection (Reference)” on page 89

For overview information about device protection, see “Controlling Access to Devices” on page 44.

### Configuring Devices (Task Map)

The following task map points to tasks for managing access to devices.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage device policy</td>
<td>“Configuring Device Policy (Task Map)” on page 76</td>
</tr>
<tr>
<td>Manage device allocation</td>
<td>“Managing Device Allocation (Task Map)” on page 79</td>
</tr>
<tr>
<td>Use device allocation</td>
<td>“Allocating Devices (Task Map)” on page 84</td>
</tr>
</tbody>
</table>
Configuring Device Policy (Task Map)

The following task map points to device configuration procedures that are related to device policy.

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the device policy for the devices on your system</td>
<td>Lists the devices and their device policy.</td>
<td>“How to View Device Policy” on page 76</td>
</tr>
<tr>
<td>Require privilege for device use</td>
<td>Uses privileges to protect a device.</td>
<td>“How to Change the Device Policy on an Existing Device” on page 77</td>
</tr>
<tr>
<td>Remove privilege requirements from a device</td>
<td>Removes or lessens the privileges that are required to access a device.</td>
<td>Example 4–3</td>
</tr>
<tr>
<td>Audit changes in device policy</td>
<td>Records changes in device policy in the audit trail</td>
<td>“How to Audit Changes in Device Policy” on page 78</td>
</tr>
<tr>
<td>Access /dev/arp</td>
<td>Gets Solaris IP MIB-II information.</td>
<td>“How to Retrieve IP MIB-II Information From a /dev/* Device” on page 78</td>
</tr>
</tbody>
</table>

Configuring Device Policy

Device policy restricts or prevents access to devices that are integral to the system. The policy is enforced in the kernel.

▼ How to View Device Policy

- Display the device policy for all devices on your system.

  `%% getdevpolicy | more`

  DEFAULT

  read_priv_set=none
  write_priv_set=none
  ip:* read_priv_set=net_rawaccess
  write_priv_set=net_rawaccess

  ...

Example 4–1 Viewing the Device Policy for a Specific Device

In this example, the device policy for three devices is displayed.

  `% getdevpolicy /dev/allkmem /dev/ipsecesp /dev/hme`  
  /dev/allkmem
  read_priv_set=all
How to Change the Device Policy on an Existing Device

1. Assume a role that includes the Device Security rights profile, or become superuser.
   The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2. Add policy to a device.
   
   ```
   # update_drv -a -p policy device-driver
   
   -a     Specifies a policy for device-driver.
   
   -p policy Is the device policy for device-driver. Device policy specifies two sets of privileges.
          One set is required to read the device. The other set is required to write to the device.

   device-driver Is the device driver.
   
   For more information, see the update_drv(1M) man page.
   ```

Example 4–2 Adding Policy to an Existing Device

In the following example, device policy is added to the ipnat device.

```bash
# getdevpolicy /dev/ipnat
/dev/ipnat
   read_priv_set=none
   write_priv_set=none
# update_drv -a \
   -p 'read_priv_set=net_rawaccess write_priv_set=net_rawaccess' ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
   read_priv_set=net_rawaccess
   write_priv_set=net_rawaccess
```
Example 4–3  Removing Policy From a Device

In the following example, the read set of privileges is removed from the device policy for the ipnat device.

```bash
# getdevpolicy /dev/ipnat
/dev/ipnat
     read_priv_set=net_rawaccess
     write_priv_set=net_rawaccess
# update_drv -a -p write_priv_set=net_rawaccess ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
     read_priv_set=none
     write_priv_set=net_rawaccess
```

How to Audit Changes in Device Policy

By default, the as audit class includes the AUE_MODDEVPLCY audit event.

1  Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2  Preselect the audit class that includes AUE_MODDEVPLCY audit event.
   Add the as class to the flags line of the audit_control file. The file would appear similar to the following:

   ```bash
   # audit_control file
dir:/var/audit
   flags:lo,as
   minfree:20
   naflags:lo
   ```

   For detailed instructions, see “How to Modify the audit_control File” on page 524.

How to Retrieve IP MIB-II Information From a /dev/* Device

Applications that retrieve Solaris IP MIB-II information should open /dev/arp, not /dev/ip.

1  Determine the device policy on /dev/ip and /dev/arp.

   ```bash
   % getdevpolicy /dev/ip /dev/arp
   /dev/ip
   ```
read_priv_set=net_rawaccess
write_priv_set=net_rawaccess
/dev/arp
read_priv_set=none
write_priv_set=none

Note that the net_rawaccess privilege is required for reading and writing to /dev/ip. No privileges are required for /dev/arp.

2 Open /dev/arp and push the tcp and udp modules.

No privileges are required. This method is equivalent to opening /dev/ip and pushing the arp, tcp and udp modules. Because opening /dev/ip now requires a privilege, the /dev/arp method is preferred.

Managing Device Allocation (Task Map)

The following task map points to procedures that enable and configure device allocation. Device allocation is not enabled by default. After device allocation is enabled, see “Allocating Devices (Task Map)” on page 84.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make a device allocatable</td>
<td>Enables a device to be allocated to one user at a time.</td>
<td>“How to Make a Device Allocatable” on page 80</td>
</tr>
<tr>
<td>Authorize users to allocate a device</td>
<td>Assigns device allocation authorizations to users.</td>
<td>“How to Authorize Users to Allocate a Device” on page 80</td>
</tr>
<tr>
<td>View the allocatable devices on your system</td>
<td>Lists the devices that are allocatable, and the state of the device.</td>
<td>“How to View Allocation Information About a Device” on page 81</td>
</tr>
<tr>
<td>Forcibly allocate a device</td>
<td>Allocates a device to a user who has an immediate need.</td>
<td>“Forcibly Allocating a Device” on page 82</td>
</tr>
<tr>
<td>Forcibly deallocate a device</td>
<td>Deallocates a device that is currently allocated to a user.</td>
<td>“Forcibly Deallocating a Device” on page 82</td>
</tr>
<tr>
<td>Change the allocation properties of a device</td>
<td>Changes the requirements for allocating a device</td>
<td>“How to Change Which Devices Can Be Allocated” on page 83</td>
</tr>
<tr>
<td>Create a device-clean script</td>
<td>Purges data from a physical device.</td>
<td>“Writing New Device-Clean Scripts” on page 94</td>
</tr>
<tr>
<td>Disable device allocation</td>
<td>Removes allocation restrictions from all devices.</td>
<td>“How to Disable Auditing” on page 540</td>
</tr>
<tr>
<td>Audit device allocation</td>
<td>Records device allocation in the audit trail</td>
<td>“How to Audit Device Allocation” on page 84</td>
</tr>
</tbody>
</table>
Managing Device Allocation

Device allocation restricts or prevents access to peripheral devices. Restrictions are enforced at user allocation time. By default, users must have authorization to access allocatable devices.

How to Make a Device Allocatable

If you have already run the bsmconv command to enable auditing, then device allocation is already enabled on your system. For more information, see the bsmconv(1M) man page.

1 Assume a role that includes the Audit Control rights profile, or become superuser.

   The Primary Administrator role includes the Audit Control rights profile. You can also assign the Audit Control rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2 Enable device allocation.

   # bsmconv
   
   This script is used to enable the Basic Security Module (BSM).
   
   Shall we continue with the conversion now? [y/n] y
   bsmconv: INFO: turning on audit module.
   bsmconv: INFO: initializing device allocation files.
   
   The Basic Security Module is ready.
   If there were any errors, please fix them now.
   Configure BSM by editing files located in /etc/security.
   Reboot this system now to come up with BSM enabled.

Note – The Volume Management daemon (/etc/rc3.d/S81volmgt) is disabled by this command.

How to Authorize Users to Allocate a Device

1 Assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Create a rights profile that contains the appropriate authorization and commands.

   Typically, you would create a rights profile that includes the solaris.device.allocate authorization. Follow the instructions in “How to Create or Change a Rights Profile” on page 206. Give the rights profile appropriate properties, such as the following:
Create a role for the rights profile.
Follow the instructions in “How to Create and Assign a Role By Using the GUI” on page 191. Use the following role properties as a guide:

- Role name: devicealloc
- Role full name: Device Allocator
- Role description: Allocates and mounts allocated devices
- Rights profile: Device Allocation
  - This rights profile must be at the top of the list of profiles that are included in the role.

Assign the role to every user who is permitted to allocate a device.

Teach the users how to use device allocation.
For examples of allocating removable media, see “How to Allocate a Device” on page 85.
Because the Volume Management daemon (vold) is not running, removable media are not automatically mounted. For examples of mounting a device that has been allocated, see “How to Mount an Allocated Device” on page 86.

How to View Allocation Information About a Device

Before You Begin
Device allocation must be enabled for this procedure to succeed. To enable device allocation, see “How to Make a Device Allocatable” on page 80.

1 Assume a role that includes the Device Security rights profile, or become superuser.
The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2 Display information about allocatable devices on your system.
   # list_devices device-name
   where device-name is one of the following:
   - audio[n] – Is a microphone and speaker.
   - fd[n] – Is a diskette drive.
   - st[n] – Is a tape drive.
If the list_devices command returns an error message similar to the following, then either device allocation is not enabled, or you do not have sufficient permissions to retrieve the information.

```
list_devices: No device maps file entry for specified device.
```

For the command to succeed, enable device allocation and assume a role with the solaris.device.revoke authorization.

**Forcibly Allocating a Device**

Forcible allocation is used when someone has forgotten to deallocate a device. Forcible allocation can also be used when a user has an immediate need for a device.

**Before You Begin**

The user or role must have the solaris.device.revoke authorization.

1. Determine if you have the appropriate authorizations in your role.
   
   ```
   $ auths
   solaris.device.allocate solaris.device.revoke
   ```

2. Forcibly allocate the device to the user who needs the device.
   
   In this example, the tape drive is forcibly allocated to the user jdoe.
   
   ```
   $ allocate -U jdoe
   ```

**Forcibly Deallocating a Device**

Devices that a user has allocated are not automatically deallocated when the process terminates or when the user logs out. Forcible deallocation is used when a user has forgotten to deallocate a device.

**Before You Begin**

The user or role must have the solaris.device.revoke authorization.

1. Determine if you have the appropriate authorizations in your role.
   
   ```
   $ auths
   solaris.device.allocate solaris.device.revoke
   ```

2. Forcibly deallocate the device.
   
   In this example, the printer is forcibly deallocated. The printer is now available for allocation by another user.
   
   ```
   $ deallocate -f /dev/lp/printer-1
   ```
How to Change Which Devices Can Be Allocated

1. **Assume a role that includes the Device Security rights profile, or become superuser.**
   - The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2. **Specify if authorization is required, or specify the `solaris.device.allocate` authorization.**
   - Change the fifth field in the device entry in the `device_allocate` file.
     ```
     audio;audio;reserved;reserved;@;/etc/security/lib/audio_clean
     fd0;fd;reserved;reserved;@;/etc/security/lib/fd_clean
     sr0;sr;reserved;reserved;@;/etc/security/lib/sr_clean
     ```
   - Where `solaris.device.allocate` indicates that a user must have the `solaris.device.allocate` authorization to use the device.

**Example 4–4 Permitting Any User to Allocate a Device**

In the following example, any user on the system can allocate any device. The fifth field in every device entry in the `device_allocate` file has been changed to an at sign (`@`).

```bash
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;@;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;@;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;@;/etc/security/lib/sr_clean
...```

**Example 4–5 Preventing Some Peripheral Devices From Being Used**

In the following example, the audio device cannot be used. The fifth field in the audio device entry in the `device_allocate` file has been changed to an asterisk (`*`).

```bash
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;*/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;*/etc/security/lib/sr_clean
...```

**Example 4–6 Preventing All Peripheral Devices From Being Used**

In the following example, no peripheral device can be used. The fifth field in every device entry in the `device_allocate` file has been changed to an asterisk (`*`).
$ whoami
deviceSec

$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;*/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;*/etc/security/lib/sr_clean
...

▶ How to Audit Device Allocation

By default, the device allocation commands are in the other audit class.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Preselect the ot class for auditing.

Add the ot class to the flags line of the audit_control file. The file would appear similar to the following:

```
# audit_control file
dir:/var/audit
flags:lo,ot
minfree:20
naflags:lo
```

For detailed instructions, see “How to Modify the audit_control File” on page 524.

Allocating Devices (Task Map)

The following task map points to procedures that show users how to allocate devices.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocate a device</td>
<td>Enables a user to use a device, while preventing any other user from using the device.</td>
<td>“How to Allocate a Device” on page 85</td>
</tr>
<tr>
<td>Mount an allocated device</td>
<td>Enables a user to view a device that requires mounting, such as a CD-ROM or a diskette.</td>
<td>“How to Mount an Allocated Device” on page 86</td>
</tr>
<tr>
<td>Deallocate a device</td>
<td>Makes an allocatable device available for use by another user.</td>
<td>“How to Deallocate a Device” on page 88</td>
</tr>
</tbody>
</table>
Allocating Devices

Device allocation reserves the use of a device to one user at a time. Devices that require a mount point must be mounted.

▼ How to Allocate a Device

Before You Begin

Device allocation must be enabled, as described in “How to Make a Device Allocatable” on page 80. If authorization is required, the user must have the authorization.

1 Allocate the device.
   Specify the device by device name.
   `% allocate device-name`

2 Verify that the device is allocated.
   Run the identical command.
   `% allocate device-name`
   `allocate. Device already allocated`.

Example 4–7 Allocating a Microphone

In this example, the user jdoe allocates a microphone, audio.

% whoami
jdoe
% allocate audio

Example 4–8 Allocating a Printer

In this example, a user allocates a printer. No one else can print to printer-1 until the user deallocates it, or until the printer is forcibly allocated to another user.

% allocate /dev/lp/printer-1

For an example of forcible deallocation, see “Forcibly Deallocating a Device” on page 82.

Example 4–9 Allocating a Tape Drive

In this example, the user jdoe allocates a tape drive, st0.

% whoami
jdoe
% allocate st0

Troubleshooting

If the allocate command cannot allocate the device, an error message is displayed in the console window. For a list of allocation error messages, see the allocate(1) man page.
## How to Mount an Allocated Device

### Before You Begin
The user or role has allocated the device. To mount a device, the user or role must have the privileges that are required for mounting the device. To give the required privileges, see “How to Authorize Users to Allocate a Device” on page 80.

1. **Assume a role that can allocate and mount a device.**
   ```
   % su role-name
   Password: <Type role-name password>
   $ 
   ```

2. **Create and protect a mount point in the role’s home directory.**
   You only need to do this step the first time you need a mount point.
   ```
   $ mkdir mount-point ; chmod 700 mount-point 
   ```

3. **List the allocatable devices.**
   ```
   $ list_devices -l
   List of allocatable devices
   ```

4. **Allocate the device.**
   Specify the device by device name.
   ```
   $ allocate device-name 
   ```

5. **Mount the device.**
   ```
   $ mount -o ro -F filesystem-type device-path mount-point
   ```
   - `-o ro` indicates that the device is to be mounted read-only. Use `-o rw` to indicate that you should be able to write to the device.
   - `-F filesystem-type` indicates the file system format of the device. Typically, a CD-ROM is formatted with an HSFS file system. A diskette is typically formatted with a PCFS file system.
   - `device-path` indicates the path to the device. The output of the `list_devices -l` command includes the `device-path`.
   - `mount-point` indicates the mount point that you created in Step 2.

### Example 4–10 Allocating a Diskette Drive

In this example, a user assumes a role that can allocate and mount a diskette drive, `/dev/fd0`. The diskette is formatted with a PCFS file system.

```
% roles
devicealloc
```
% su devicealloc
Password: <Type devicealloc password>
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
...  
device: fd0 type: fd files: /dev/diskette /dev/rdiskette /dev/fd0a  
...  
$ allocate fd0
$ mount -o ro -F pcfs /dev/diskette /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
List of the contents of diskette

Example 4–11 Allocating a CD-ROM Drive

In this example, a user assumes a role that can allocate and mount a CD-ROM drive, sr0. The drive is formatted as an HSFS file system.

% roles
devicealloc
% su devicealloc
Password: <Type devicealloc password>
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
...  
device: sr0 type: sr files: /dev/sr0 /dev/rsr0 /dev/dsk/c0t2d0s0 ...  
...  
$ allocate sr0
$ mount -o ro -F hsfs /dev/sr0 /home/devicealloc/mymnt
$ cd /home/devicealloc/mymnt ; ls
List of the contents of CD-ROM

Troubleshooting If the mount command cannot mount the device, an error message is displayed: mount: insufficient privileges. Check the following:

- Make sure that you are executing the mount command in a profile shell. If you have assumed a role, the role has a profile shell. If you are a user who has been assigned a profile with the mount command, you must create a profile shell. The commands pfsh, pfksh, and pfcsh create a profile shell.

- Make sure that you own the specified mount point. You should have read, write, and execute access to the mount point.

Contact your administrator if you still cannot mount the allocated device.
How to Deallocate a Device

Deallocation enables other users to allocate and use the device when you are finished.

Before You Begin

You must have allocated the device.

1. If the device is mounted, unmount the device.
   
   $ cd $HOME
   $ umount mount-point

2. Deallocate the device.
   
   $ deallocate device-name

Example 4–12 Deallocating a Microphone

In this example, the user jdoe deallocates the microphone, audio.

% whoami
jdoe
%
deallocate audio

Example 4–13 Deallocating a CD-ROM Drive

In this example, the Device Allocator role deallocates a CD-ROM drive. After the message is printed, the CD-ROM is ejected.

$ whoami
devicealloc
$ cd /home/devicealloc
$ umount /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
$
$ deallocate sr0
/dev/sr0: 326o
/dev/rsr0: 326o
...

sr_clean: Media in sr0 is ready. Please, label and store safely.
Device Protection (Reference)

Devices in the Solaris OS are protected by device policy. Peripheral devices can be protected by device allocation. Device policy is enforced by the kernel. Device allocation is optionally enabled, and is enforced at the user level.

Device Policy Commands

Device management commands administer the device policy on local files. Device policy can include privilege requirements. Only superuser or a role of equivalent capabilities can manage devices.

The following table lists the device management commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>devfsadm</td>
<td>Administers devices and device drivers on a running system. Also loads device policy. The devfsadm command enables the cleanup of dangling /dev links to disk, tape, port, audio, and pseudo devices. Devices for a named driver can also be reconfigured.</td>
<td>devfsadm(1M)</td>
</tr>
<tr>
<td>getdevpolicy</td>
<td>Displays the policy associated with one or more devices. This command can be run by any user.</td>
<td>getdevpolicy(1M)</td>
</tr>
<tr>
<td>add_drv</td>
<td>Adds a new device driver to a running system. Contains options to add device policy to the new device. Typically, this command is called in a script when a device driver is being installed.</td>
<td>add_drv(1M)</td>
</tr>
<tr>
<td>update_drv</td>
<td>Updates the attributes of an existing device driver. Contains options to update the device policy for the device. Typically, this command is called in a script when a device driver is being installed.</td>
<td>update_drv(1M)</td>
</tr>
<tr>
<td>rem_drv</td>
<td>Removes a device or device driver.</td>
<td>rem_drv(1M)</td>
</tr>
</tbody>
</table>

Device Allocation

Device allocation can protect your site from loss of data, computer viruses, and other security breaches. Unlike device policy, device allocation is optional. Devices are not allocatable until the bsmconv script is run. Device allocation uses authorizations to limit access to allocatable devices.

Components of Device Allocation

The components of the device allocation mechanism are as follows:
The allocate, deallocate, dminfo, and list_devices commands. For more information, see “Device Allocation Commands” on page 90.

Device-cleanup scripts for each allocatable device.

These commands and scripts use the following local files to implement device allocation:

- The /etc/security/device_allocate file. For more information, see the device_allocate(4) man page.
- The /etc/security/device_maps file. For more information, see the device_maps(4) man page.
- A lock file, in the /etc/security/dev directory, for each allocatable device.
- The changed attributes of the lock files that are associated with each allocatable device.

Note – The /etc/security/dev directory might not be supported in future releases of the Solaris OS.

### Device Allocation Commands

With uppercase options, the allocate, deallocate, and list_devices commands are administrative commands. Otherwise, these commands are user commands. The following table lists the device allocation commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>bsmconv</td>
<td>Creates databases to handle device allocation. Also enables the auditing service. You must be superuser or in the Primary Administrator role.</td>
<td>bsmconv(1M)</td>
</tr>
<tr>
<td>dminfo</td>
<td>Searches for an allocatable device by device type, by device name, and by full path name.</td>
<td>dminfo(1M)</td>
</tr>
<tr>
<td>list_devices</td>
<td>Lists the status of allocatable devices.</td>
<td>list_devices(1)</td>
</tr>
<tr>
<td></td>
<td>Lists all the device-special files that are associated with any device that is listed in the device_maps file.</td>
<td></td>
</tr>
<tr>
<td>list_devices -U</td>
<td>Lists the devices that are allocatable or allocated to the specified user ID. This option allows you to check which devices are allocatable or allocated to another user. You must have the solaris.device.revoke authorization.</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 4–2 Device Allocation Commands  
(Continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocate</td>
<td>Reserves an allocatable device for use by one user.</td>
<td>allocate(1)</td>
</tr>
<tr>
<td></td>
<td>By default, a user must have the solaris.device.allocate authorization to allocate a device. You can modify the device_allocate file to not require user authorization. Then, any user on the system can request the device to be allocated for use.</td>
<td></td>
</tr>
<tr>
<td>deallocale</td>
<td>Removes the allocation reservation from a device.</td>
<td>deallocale(1)</td>
</tr>
</tbody>
</table>

### Authorizations for the Allocation Commands

By default, users must have the solaris.device.allocate authorization to reserve an allocatable device. To create a rights profile to include the solaris.device.allocate authorization, see “How to Authorize Users to Allocate a Device” on page 80.

Administrators must have the solaris.device.revoke authorization to change the allocation state of any device. For example, the -U option to the allocate and list_devices commands, and the -F option to the deallocale command require the solaris.device.revoke authorization.

For more information, see “Commands That Require Authorizations” on page 226.

### Allocate Error State

A device is put in an allocate error state when the deallocale command fails to deallocale, or when the allocate command fails to allocate. When an allocatable device is in an allocate error state, then the device must be forcibly deallocale. Only superuser or a role with the Device Management rights profile or the Device Security rights profile can handle an allocate error state.

The deallocale command with the -F option forces deallocale. Or, you can use allocate -U to assign the device to a user. Once the device is allocated, you can investigate any error messages that appear. After any problems with the device are corrected, you can forcibly deallocale it.

### device_maps File

Device maps are created when you set up device allocation. A default /etc/security/device_maps file is created by the bsmconv command when the auditing service is enabled. This initial device_maps file can be customized for your site. The device_maps file includes the device names, device types, and device-special files that are associated with each allocatable device.

The device_maps file defines the device-special file mappings for each device, which in many cases is not intuitive. This file allows programs to discover which device-special files map to which devices. You can use the dminfo command, for example, to retrieve the device name, the device type, and the device-special files to specify when you set up an allocatable device. The dminfo command uses the device_maps file to report this information.

Each device is represented by a one-line entry of the form:
The following is an example of an entry in a `device_maps` file for a diskette drive, `fd0`:

```
fd0:\
   fd:\
      /dev/diskette /dev/rdiskette /dev/fd0a /dev/rfd0a \
      /dev/fd0b /dev/rfd0b /dev/fd0c /dev/fd0 /dev/rfd0c /dev/rfd0:
```

Lines in the `device_maps` file can end with a backslash (\) to continue an entry on the next line. Comments can also be included. A pound sign (#) comments all subsequent text until the next newline that is not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field. The fields are defined as follows:

- **device-name**: Specifies the name of the device. For a list of current device names, see "How to View Allocation Information About a Device" on page 81.
- **device-type**: Specifies the generic device type. The generic name is the name for the class of devices, such as `st`, `fd`, or `audio`. The **device-type** field logically groups related devices.
- **device-list**: Lists the device-special files that are associated with the physical device. The **device-list** must contain all of the special files that allow access to a particular device. If the list is incomplete, a malevolent user can still obtain or modify private information. Valid entries for the **device-list** field reflect the device files that are located in the `/dev` directory.

### **device_allocate** File

An initial `/etc/security/device_allocate` file is created by the `bsmconv` command when the auditing service is enabled. This initial `device_allocate` file can be used as a starting point. You can modify the `device_allocate` file to change devices from allocatable to nonallocatable, or to add new devices. A sample `device_allocate` file follows.

```
st0:st;;;;/etc/security/lib/st_clean
fd0:fd;;;;/etc/security/lib/fd_clean
sr0:sr;;;;/etc/security/lib/sr_clean
audio:audio;;;;*;/etc/security/lib/audio_clean
```

An entry in the `device_allocate` file does not mean that the device is allocatable, unless the entry specifically states that the device is allocatable. In the sample `device_allocate` file, note the asterisk (*) in the fifth field of the audio device entry. An asterisk in the fifth field indicates to the system that the device is not allocatable. Therefore, the device cannot be used. Other values or no value in this field indicates that the device can be used.

In the `device_allocate` file, each device is represented by a one-line entry of the form:

```
device-name;device-type;reserved;reserved;auths;device-exec
```
Lines in the `device_allocate` file can end with a backslash (\) to continue an entry on the next line. Comments can also be included. A pound sign (#) comments all subsequent text until the next newline that is not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field. The fields are defined as follows:

- **device-name**: Specifies the name of the device. For a list of current device names, see “How to View Allocation Information About a Device” on page 81.
- **device-type**: Specifies the generic device type. The generic name is the name for the class of devices, such as `st`, `fd`, and `sr`. The `device-type` field logically groups related devices. When you make a device allocatable, retrieve the device name from the `device-type` field in the `device_maps` file.
- **reserved**: Sun reserves the two fields that are marked reserved for future use.
- **auths**: Specifies whether the device is allocatable. An asterisk (*) in this field indicates that the device is not allocatable. An authorization string, or an empty field, indicates that the device is allocatable. For example, the string `solaris.device.allocate` in the `auths` field indicates that the `solaris.device.allocate` authorization is required to allocate the device. An at sign (@) in this file indicates that the device is allocatable by any user.
- **device-exec**: Supplies the path name of a script to be invoked for special handling, such as cleanup and object-reuse protection during the allocation process. The `device-exec` script is run any time that the device is acted on by the `deallocate` command.

For example, the following entry for the `sr0` device indicates that the CD-ROM drive is allocatable by a user with the `solaris.device.allocate` authorization:

```
sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean
```

You can decide to accept the default devices and their defined characteristics. After you install a new device, you can modify the entries. Any device that needs to be allocated before use must be defined in the `device_allocate` and `device_maps` files for that device’s system. Currently, cartridge tape drives, diskette drives, CD-ROM drives, and audio chips are considered allocatable. These device types have device-clean scripts.

---

**Note** – Xylogics™ tape drives or Archive tape drives also use the `st_clean` script that is supplied for SCSI devices. You need to create your own device-clean scripts for other devices, such as modems, terminals, graphics tablets, and other allocatable devices. The script must fulfill object-reuse requirements for that type of device.

---

**Device-Clean Scripts**

Device allocation satisfies part of what is called the object reuse requirement. The `device-clean` scripts address the security requirement that all usable data be purged from a physical device before reuse. The data is cleared before the device is allocatable by another user. By default, cartridge tape drives,
diskette drives, CD-ROM drives, and audio devices require device-clean scripts. The Solaris OS provides the scripts. This section describes what device-clean scripts do.

**Device-Clean Script for Tapes**

The `st_clean` device-clean script supports three tape devices:

- **SCSI ¼-inch tape**
- **Archive ¼-inch tape**
- **Open-reel ½-inch tape**

The `st_clean` script uses the `rewoffl` option to the `mt` command to clean up the device. For more information, see the `mt(1)` man page. If the script runs during system boot, the script queries the device to determine if the device is online. If the device is online, the script determines if the device has media in it. The ¼-inch tape devices that have media in them are placed in the allocate error state. The allocate error state forces the administrator to manually clean up the device.

During normal system operation, when the `deallocate` command is executed in interactive mode, the user is prompted to remove the media.Deallocation is delayed until the media is removed from the device.

**Device-Clean Scripts for Diskettes and CD-ROM Drives**

The following device-clean scripts are provided for diskettes and CD-ROM drives:

- **fd_clean** script – Is a device-clean script for diskettes.
- **sr_clean** script – Is a device-clean script for CD-ROM drives.

The scripts use the `eject` command to remove the media from the drive. If the `eject` command fails, the device is placed in the allocate error state. For more information, see the `eject(1)` man page.

**Device-Clean Script for Audio**

Audio devices are cleaned up with an `audio_clean` script. The script performs an `AUDIO_GETINFO` ioctl system call to read the device. The script then performs an `AUDIO_SETINFO` ioctl system call to reset the device configuration to the default.

**Writing New Device-Clean Scripts**

If you add more allocatable devices to the system, you might need to create your own device-clean scripts. The `deallocate` command passes a parameter to the device-clean scripts. The parameter, which is shown here, is a string that contains the device name. For more information, see the `device_allocate(4)` man page.

`clean-script [-1|f|s] device-name`
Device-cleanscripts must return "0" for success and greater than "0" for failure. The options -I, -f, and -S determine the running mode of the script:

- **-I** Is needed during system boot only. All output must go to the system console. Failure or inability to forcibly eject the media must put the device in the allocate error state.

- **-i** Similar to the -I option, except that output is suppressed.

- **-f** Is for forced cleanup. The option is interactive and assumes that the user is available to respond to prompts. A script with this option must attempt to complete the cleanup if one part of the cleanup fails.

- **-S** Is for standard cleanup. The option is interactive and assumes that the user is available to respond to prompts.
Using the Basic Audit Reporting Tool (Tasks)

This chapter describes how to create a manifest of the files on a system and how to use that manifest to check the integrity of the system. The Basic Audit Reporting Tool (BART) enables you to comprehensively validate systems by performing file-level checks of a system over time.

The following is a list of the information in this chapter:

- “Using BART (Task Map)” on page 100
- “Basic Audit Reporting Tool (Overview)” on page 97
- “Using BART (Tasks)” on page 101
- “BART Manifest, Rules File, and Reporting (Reference)” on page 114

Basic Audit Reporting Tool (Overview)

BART is a file tracking tool that operates entirely at the file system level. Using BART gives you the ability to quickly, easily, and reliably gather information about the components of the software stack that is installed on deployed systems. Using BART can greatly reduce the costs of administering a network of systems by simplifying time-consuming administrative tasks.

BART enables you to determine what file-level changes have occurred on a system, relative to a known baseline. You use BART to create a baseline or control manifest from a fully installed and configured system. You can then compare this baseline with a snapshot of the system at a later time, generating a report that lists file-level changes that have occurred on the system since it was installed.

The bart command is a standard UNIX command. You can redirect the output of the bart command to a file for later processing.

BART Features

BART has been designed with an emphasis on a simple syntax that is both powerful and flexible. The tool enables you to generate manifests of a given system over time. Then, when the system’s files need to be validated, you can generate a report by comparing the old and new manifests. Another way to
use BART is to generate manifests of several similar systems and run system-to-system comparisons. The main difference between BART and existing auditing tools is that BART is flexible, both in terms of what information is tracked and what information is reported.

Additional benefits and uses of BART include the following:

- Provides an efficient and easy method for cataloging a system that is running the Solaris software at the file level.
- Enables you to define which files to monitor and gives you the ability to modify profiles when necessary. This flexibility allows you to monitor local customizations and enables you to reconfigure software easily and efficiently.
- Ensures that systems are running reliable software.
- Allows you to monitor file-level changes of a system over time, which can help you locate corrupted or unusual files.
- Helps you troubleshoot system performance issues.

**BART Components**

BART has two main components and one optional component:

- BART Manifest
- BART Report
- BART Rules File

**BART Manifest**

You use the `bart create` command to take a file-level snapshot of a system at a particular time. The output is a catalog of files and file attributes called a *manifest*. The manifest lists information about all the files or specific files on a system. It contains information about attributes of files, which can include some uniquely identifying information, such as an MD5 checksum. For more information about the MD5 checksum, see the `md5(3EXT)` man page. A manifest can be stored and transferred between client and server systems.

**Note** – BART does not cross file system boundaries, with the exception of file systems of the same type. This constraint makes the output of the `bart create` command more predictable. For example, without arguments, the `bart create` command catalogs all UFS file systems under the root `(/)` directory. However, no NFS or TMPFS file systems or mounted CD-ROMs would be cataloged. When creating a manifest, do not attempt to audit file systems on a network. Note that using BART to monitor networked file systems can consume large resources to generate manifests that will have little value.

For more information about BART manifests, see “BART Manifest File Format” on page 114.
**BART Report**

The report tool has three inputs: the two manifests to be compared and an optional user-provided rules file that indicates which discrepancies are to be flagged.

You use the `bart compare` command to compare two manifests, a *control manifest* and a *test manifest*. These manifests must be prepared with the same file systems, options, and rules file that you use with the `bart create` command.

The output of the `bart compare` command is a report that lists per-file discrepancies between the two manifests. A *discrepancy* is a change to any attribute for a given file that is cataloged for both manifests. Additions or deletions of file entries between the two manifests are also considered discrepancies.

There are two levels of control when reporting discrepancies:

- When generating a manifest
- When producing reports

These levels of control are intentional, since generating a manifest is more costly than reporting discrepancies between two manifests. Once you have created manifests, you have the ability to compare manifests from different perspectives by running the `bart compare` command with different rules files.

For more information about BART reports, see “BART Reporting” on page 116.

**BART Rules File**

The *rules file* is a text file that you can optionally use as input to the `bart` command. This file uses inclusion and exclusion rules. A rules file is used to create custom manifests and reports. A rules file enables you to express in a concise syntax which sets of files you want to catalog, as well as which attributes to monitor for any given set of files. When you compare manifests, the rules file aids in flagging discrepancies between the manifests. Using a rules file is an effective way to gather specific information about files on a system.

You create a rules file by using a text editor. With a rules file, you can perform the following tasks:

- Use the `bart create` command to create a manifest that lists information about all or specific files on a system.
- Use the `bart compare` command to generate a report that monitors specific attributes of a file system.
Using BART (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a manifest.</td>
<td>Obtain a manifest that lists information about every file that is installed on a system.</td>
<td>&quot;How to Create a Manifest&quot; on page 101</td>
</tr>
<tr>
<td>Create a custom manifest.</td>
<td>Obtain a manifest that lists information about specific files that are installed on a system in one of the following ways:</td>
<td>&quot;How to Customize a Manifest&quot; on page 103</td>
</tr>
<tr>
<td></td>
<td>- By specifying a subtree</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- By specifying a file name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- By using a rules file</td>
<td></td>
</tr>
<tr>
<td>Compare manifests for the same system over time. Or, compare manifests for different systems with a control system manifest.</td>
<td>Obtain a report that compares changes to a system over time. Or, obtain a report that compares one or several systems to control system.</td>
<td>&quot;How to Compare Manifests for the Same System Over Time&quot; on page 107 &quot;How to Compare Manifests From a Different System With the Manifest of a Control System&quot; on page 109</td>
</tr>
<tr>
<td>(Optional) Customize a BART report.</td>
<td>Obtain a custom BART report in one of the following ways:</td>
<td>&quot;How to Customize a BART Report by Specifying File Attributes&quot; on page 112 &quot;How to Customize a BART Report by Using a Rules File&quot; on page 112</td>
</tr>
<tr>
<td></td>
<td>- By specifying attributes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- By using a rules file</td>
<td></td>
</tr>
</tbody>
</table>

Note – You can create several rules files for different purposes. However, if you create a manifest by using a rules file, you must use the same rules file when you compare the manifests. If you do not use the same rules file when comparing manifests that were created with a rules file, the output of the bart compare command will list many invalid discrepancies.

A rules file can also contain syntax errors and other ambiguous information as a result of user error. If a rules file does contain misinformation, these errors will also be reported.

Using a rules file to monitor specific files and file attributes on a system requires planning. Before you create a rules file, decide which files and file attributes on the system you want to monitor. Depending on what you are trying to accomplish, you might use a rules file to create manifests, compare manifests, or for purposes.

For more information about the BART rules file, see “BART Rules File Format” on page 115 and the bart_rules (4) man page.
Using BART (Tasks)

You can run the bart command as a regular user, superuser, or a user who has assumed the Primary Administrator role. If you run the bart command as a regular user, you will only be able to catalog and monitor files that you have permission to access, for example, information about files in your home directory. The advantage of becoming superuser when you run the bart command is that the manifests you create will contain information about hidden and private files that you might want to monitor. If you need to catalog and monitor information about files that have restricted permissions, for example, the /etc/passwd or /etc/shadow file, run the bart command as superuser or assume an equivalent role. For more information about using role-based access control, see "Configuring RBAC (Task Map)" on page 188.

BART Security Considerations

Running the bart command as superuser makes the output readable by anyone. This output might contain file names that are intended to be private. If you become superuser when you run the bart command, take appropriate measures to protect the output. For example, use options that generate output files with restrictive permissions.

Note – The procedures and examples in this chapter show the bart command run by superuser. Unless otherwise specified, running the bart command as superuser is optional.

How to Create a Manifest

You can create a manifest of a system immediately after an initial Solaris software installation. This type of manifest will provide you with a baseline for comparing changes to the same system over time. Or, you can use this manifest to compare with the manifests for different systems. For example, if you take a snapshot of each system on your network, and then compare each test manifest with the control manifest, you can quickly determine what you need to do to synchronize the test system with the baseline configuration.

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 After installing the Solaris software, create a control manifest and redirect the output to a file.
   # bart create options > control-manifest
   -R Specifies the root directory for the manifest. All paths specified by the rules will be interpreted relative to this directory. All paths reported in the manifest will be relative to this directory.
-I Accepts a list of individual files to be cataloged, either on the command line or read from standard input.

-\texttt{r} Is the name of the rules file for this manifest. Note that –, when used with the -r option, will be read the rules file from standard input.

-\texttt{n} Turns off content signatures for all regular files in the file list. This option can be used to improve performance. Or, you can use this option if the contents of the file list are expected to change, as in the case of system log files.

3 Examine the contents of the manifest.

4 Save the manifest for future use.

Choose a meaningful name for the manifest. For example, use the system name and date that the manifest was created.

**Example 5–1 Creating a Manifest That Lists Information About Every File on a System**

If you run the bart create command without any options, information about every file that is installed on the system will be cataloged. Use this type of manifest as a baseline when you are installing many systems from a central image. Or, use this type of manifest to run comparisons when you want to ensure that the installations are identical.

For example:

```
# bart create
! Version 1.0
! Thursday, December 04, 2003 (16:17:39)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl \nmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 1024 40755 user::rwx,group::r-x,other:r-x 3fd9ea47 0 0
/.java D 512 40755 user::rwx,group::r-x,other:r-x 3f8dc04d 0 10
/.java/.userPrefs D 512 40700 user::rwx,group::r--,mask:---
other:--- 3f8dc06b 0 10
/.java/.userPrefs/.user.lock.root F 0 100600 user::rw-
group::r---,other:--- 3f8dc06b 0 10 -
/.java/.userPrefs/.userRootModFile.root F 0 100600 user::rw-
group::r---,mask:---,other:--- 3f8dc0a1 0 10 -
/.smc.properties F 1389 100644 user::rw-,group::r--,mask:r--
other:r-- 3f8dca0c0 10
```


Each manifest consists of a header and entries. Each manifest file entry is a single line, depending on the file type. For example, for each manifest entry in the preceding output, type F specifies a file and type D specifies a directory. Also listed is information about size, content, user ID, group ID, and permissions. File entries in the output are sorted by the encoded versions of the file names to correctly handle special characters. All entries are sorted in ascending order by file name. All nonstandard file names, such as those that contain embedded newline or tab characters, have the nonstandard characters quoted before being sorted.

Lines that begin with ! supply metadata about the manifest. The manifest version line indicates the manifest specification version. The date line shows the date on which the manifest was created, in date form. See the date(1) man page. Some lines are ignored by the manifest comparison tool. Ignored lines include blank lines, lines that consist only of white space, and comments that begin with #.

### How to Customize a Manifest

You can customize a manifest in one of the following ways:

- **By specifying a subtree**

  Creating a manifest for an individual subtree on a system is an efficient way to monitor changes to specific files, rather than the entire contents of a large directory. You can create a baseline manifest of a specific subtree on your system, then periodically create test manifests of the same subtree. Use the bart compare command to compare the control manifest with the test manifest. By using this option, you are able to efficiently monitor important file systems to determine whether any files have been compromised by an intruder.

- **By specifying a file name**

  Since creating a manifest that catalogs the entire system is more time-consuming, takes up more space, and is more costly, you might choose to use this option of the bart command when you want to only list information about a specific file or files on a system.
By using a rules file
You use a rules file to create custom manifests that list information about specific files and specific subtrees on a given system. You can also use a rules file to monitor specific file attributes. Using a rules file to create and compare manifests gives you the flexibility to specify multiple attributes for more than one file or subtree. Whereas, from the command line, you can only specify a global attribute definition that applies to all files for each manifest you create or report you generate.

1. Determine which files you want to catalog and monitor.

2. Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

3. After installing the Solaris software, create a custom manifest by using one of the following options:
   - By specifying a subtree:
     \# bart create -R root-directory
   - By specifying a file name or file names:
     \# bart create -I filename...
     For example:
     \# bart create -I /etc/system /etc/passwd /etc/shadow
   - By using a rules file:
     \# bart create -r rules-file

4. Examine the contents of the manifest.

5. Save the manifest for future use.

Example 5–2 Creating a Manifest by Specifying a Subtree
This example shows how to create a manifest that contains information about the files in the /etc/ssh subtree only.

\# bart create -R /etc/ssh
! Version 1.0
! Saturday, November 29, 2003 (14:05:36)
! Format:
#fname O size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
Customizing a Manifest by Specifying a File Name

This example shows how to create a manifest that lists only information about the /etc/passwd and /etc/shadow files on a system.

```
# bart create -I /etc/passwd /etc/shadow

# Example 5–3

By comparison, the following is the standard output of the `ls -al` command for the /etc/passwd and the /etc/shadow files on the same system.

```
# ls -al /etc/passwd
-r--r--r-- 1 root sys 542 Dec 4 17:42 /etc/passwd
```
Example 5–4  Customizing a Manifest by Using a Rules File

This example shows how to create a manifest by using a rules file to catalog only those files in the /etc directory. The same rules file includes directives to be used by the bart compare command for monitoring changes to the acl attribute of the /etc/system file.

- Use a text editor to create a rules file that catalogs only those files in the /etc directory.
  
  `# List information about all the files in the /etc directory.`
  
  ```
  CHECK all
  /etc
  # Check only acl changes in the /etc/system file
  IGNORE all
  CHECK acl
  /etc/system
  ```

  For more information about creating a rules file, see “BART Rules File” on page 99.

- Create a control manifest by using the rules file you created.

  ```
  # bart create -r etc.rules-file > etc.system.control-manifest
  ! Version 1.0
  ! Thursday, December 11, 2003 (21:51:32)
  # Format:
  #fname D size mode acl dirmtime uid gid
  #fname P size mode acl mtime uid gid
  #fname S size mode acl mtime uid gid
  #fname F size mode acl mtime uid gid contents
  #fname L size mode acl lnmtime uid gid dest
  #fname B size mode acl mtime uid gid devnode
  #fname C size mode acl mtime uid gid devnode
  /etc/system F 1883 100644 user::rw-,group::r--,mask:r--, other:r-- 3f81db61 0 3
  ```

- Create a test manifest whenever you want to monitor changes to the system. Prepare the test manifest identically to the control manifest by using the same bart options and the same rules file.

- Compare manifests by using the same rules file.
How to Compare Manifests for the Same System Over Time

Use this procedure when you want to monitor file-level changes to the same system over time. This type of manifest can assist you in locating corrupted or unusual files, detecting security breaches, or in troubleshooting performance issues on a system.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **After installing the Solaris software, create a control manifest of the files that you want to monitor on the system.**
   ```bash
   # bart create -R /etc > control-manifest
   ```

3. **Create a test manifest that is prepared identically to the control manifest whenever you want monitor changes to the system.**
   ```bash
   # bart create -R /etc > test-manifest
   ```

4. **Compare the control manifest with the test manifest.**
   ```bash
   # bart compare options control-manifest test-manifest > bart-report
   ```
   - `r` Is the name of the rules file for this comparison. Using the `-r` option with the – means that the directives will be read from standard input.
   - `i` Allows the user to set global IGNORE directives from the command line.
   - `p` Is the programmatic mode that generates standard non-localized output for programmatic parsing.
   - `control-manifest` Is the output from the `bart create` command for the control system.
   - `test-manifest` Is the output from the `bart create` command of the test system.

5. **Examine the BART report for oddities.**

Example 5–5 Comparing Manifests for the Same System Over Time

This example shows how to monitor changes that have occurred in the `/etc` directory between two points in time. This type of comparison enables you to quickly determine whether important files on the system have been compromised.

- Create a control manifest.
  ```bash
  # bart create -R /etc > system1.control.121203
  ! Version 1.0
  ```
Create a test manifest when you want to monitor changes to the /etc directory.

```
# bart create -R /etc > system1.test.121503
```
Version 1.0

Create a test manifest when you want to monitor changes to the /etc directory.

```
# bart create -R /etc > system1.test.121503
```
Compare the control manifest with the test manifest.

```
# bart compare system1.control.121203 system1.test.121503
/vfstab:
  mode control:100644 test:100777
  acl control:user::rw-,group::r--,mask:r--,other:r-- test:user::rwx,
    group::rwx,mask::rwx,other::rwx
```

The preceding output indicates permissions on the `vfstab` file have changed since the control manifest was created. This report can be used to investigate whether ownership, date, content, or any other file attributes have changed. Having this type of information readily available can assist you in tracking down who might have tampered with the file and when the change might have occurred.

## How to Compare Manifests From a Different System With the Manifest of a Control System

You can run system to system comparisons, thereby enabling you to quickly determine whether there are any file-level differences between a baseline system and the other systems. For example, if you have installed a particular version of the Solaris software on a baseline system, and you want to know whether other systems have identical packages installed, you can create manifests for those systems and then compare the test manifests with the control manifest. This type of comparison will list any discrepancies in the file contents for each test system that you compare with the control system.

1. **Assume the Primary Administrator role, or become superuser.**
   
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2. **After installing the Solaris software, create a control manifest.**
   
   ```
   # bart create options > control-manifest
   ```

3. **Save the control manifest.**

4. **On the test system, use the same bart options to create a manifest, and redirect the output to a file.**
   
   ```
   # bart create options > test1-manifest
   ```
   Choose a distinct and meaningful name for the test manifest.
5. Save the test manifest to a central location on the system until you are ready to compare manifests.

6. When you want to compare manifests, copy the control manifest to the location of the test manifest.
   Or, copy the test manifest to the control system.
   For example:
   
   ```bash
   # cp control-manifest /net/test-server/bart/manifests
   ```
   If the test system is not an NFS-mounted system, use FTP or some other reliable means to copy the control manifest to the test system.

7. Compare the control manifest with the test manifest and redirect the output to a file.
   
   ```bash
   # bart compare control-manifest test1-manifest > test1.report
   ```

8. Examine the BART report for oddities.

9. Repeat Step 4 through Step 9 for each test manifest that you want to compare with the control manifest.
   Use the same `bart` options for each test system.

Example 5–6  Comparing Manifests From Different Systems With the Manifest of a Control System

This example describes how to monitor changes to the contents of the `/usr/bin` directory by comparing a control manifest with a test manifest from a different system.

- Create a control manifest.

```bash
# bart create -R /usr/bin > control-manifest.121203
! Version 1.0
! Friday, December 12, 2003 (09:19:00)
# Format:
# fname D size mode acl dirmtime uid gid
# fname P size mode acl mtime uid gid
# fname S size mode acl mtime uid gid
# fname F size mode acl mtime uid gid contents
# fname L size mode acl lntime uid gid dest
# fname B size mode acl mtime uid gid devnode
# fname C size mode acl mtime uid gid devnode
/D 13312 40755 user::rwx,group::r-x,mask::r-x,other::r-x 3fd9e925 0 2
/.s F 14200 104711 user::rwx,group::--x,mask::--x,other::--x
3f8bd6b0d 1 8ec7e52d8a35b054a6394cbf71cf6
/ControlPanel L 28 120777 - 3f81dcdc 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcd5 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask::r-x,other::r-x
3f657550 0 2 d6e99b19c847ab4ec084d9088c7c7608
/activation-client F 9172 100755 user::rwx,group::r-x,mask::r-x,
```
Create a test manifest for each system that you want to compare with the control system.

```bash
# bart create -R /usr/bin > system2-manifest.121503
! Version 1.0
! Friday, December 15, 2003 (13:30:58)
# Format:
# fname D size mode acl dirmtime uid gid
# fname P size mode acl mt ime uid gid
# fname S size mode acl mt ime uid gid
# fname F size mode acl mt ime uid gid contents
# fname L size mode acl lnmtime uid gid dest
# fname B size mode acl mt ime uid gid devnode
# fname C size mode acl mt ime uid gid devnode
/D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9ea9c 0 2
/.s F 14200 104711 user::rwx,group::--x,mask:--x,other:--x
3f8dbfd6 0 1 8ec7e52d8a35ba3b054a6394cbf71cf6
/ControlPanel L 20 120777 - 3f81dc71 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcdc 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask::r-x,other::r-x
3f6b5750 0 2 d6e99b19c847ab4ec084d9088c7c7608

When you want to compare manifests, copy the manifests to the same location.

```bash
# cp control-manifest /net/system2.central/bart/manifests
```

Compare the control manifest with the test manifest.

```bash
# bart compare control-manifest system2.test > system2.report
```

When you want to compare manifests, copy the manifests to the same location.

```bash
# cp control-manifest /net/system2.central/bart/manifests
```

Compare the control manifest with the test manifest.

```bash
# bart compare control-manifest system2.test > system2.report
```

The previous output indicates that the group ID of the `su` file in the `/usr/bin` directory is not the same as that of the control system. This information can be helpful in determining whether a different version of the software was installed on the test system or if possibly someone has tampered with the file.
How to Customize a BART Report by Specifying File Attributes

This procedure is optional and explains how to customize a BART report by specifying file attributes from the command line. If you create a baseline manifest that lists information about all the files or specific on your system, you can run the `bart compare` command, specifying different attributes, whenever you need to monitor changes to a particular directory, subdirectory, file or files. You can run different types of comparisons for the same manifests by specifying different file attributes from the command line.

1 Determine which file attributes you want to monitor.

2 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

3 After installing the Solaris software, create a control manifest.

4 Create a test manifest when you want to monitor changes.
   Prepare the test manifest identically to the control manifest.

5 Compare the manifests.
   For example:
   ```sh
   # bart compare -i dirtime,lnmtime,mtime control-manifest.121503 \
   test-manifest.010504 > bart.report.010504
   ```
   Note that a comma separates each attribute you specify in the command-line syntax.

6 Examine the BART report for oddities.

How to Customize a BART Report by Using a Rules File

This procedure is also optional and explains how to customize a BART report by using a rules file as input to the `bart compare` command. By using a rules file, you can customize a BART report, which allows you the flexibility of specifying multiple attributes for more than one file or subtree. You can run different comparisons for the same manifests by using different rules files.

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*. 
Determine which files and file attributes you want to monitor.

Use a text editor to create a rules file with the appropriate directives.

After installing the Solaris software, create a control manifest by using the rules file you created.

```
# bart create -r rules-file > control-manifest
```

Create a test manifest that is prepared identically to the control manifest.

```
# bart create -r rules-file > test-manifest
```

Compare the control manifest with the test manifest by using the same rules file.

```
# bart compare -r rules-file control-manifest test-manifest > bart.report
```

Examine the BART report for oddities.

Example 5–7 Customizing a BART Report by Using a Rules File

The following rules file includes directives for both the `bart create` and the `bart compare` commands. The rules file directs the `bart create` command to list information about the contents of the `/usr/bin` directory. In addition, the rules file directs the `bart compare` command to track only size and content changes in the same directory.

```
# Check size and content changes in the /usr/bin directory.
# This rules file only checks size and content changes.
# See rules file example.

IGNORE all
CHECK size contents
/usr/bin

- Create a control manifest by using the rules file you created.
  
  ```
  # bart create -r bartrules.txt > usr_bin.control-manifest.121003
  ```

- Create a test manifest whenever you want to monitor changes to the `/usr/bin` directory.

  ```
  # bart create -r bartrules.txt > usr_bin.test-manifest.121103
  ```

- Compare the manifests by using the same rules file.

  ```
  # bart compare -r bartrules.txt usr_bin.control-manifest
  \usr_bin.test-manifest
  ```

- Examine the output of the `bart compare` command.

  ```
  /usr/bin/gunzip: add
  /usr/bin/ypcat: delete
  ```
In the preceding output, the `bart compare` command reported a discrepancy in the `/usr/bin` directory. This output indicates that `/usr/bin/ypcat` file was deleted, and the `/usr/bin/gunzip` file was added.

BART Manifest, Rules File, and Reporting (Reference)

This section includes the following reference information:
- “BART Manifest File Format” on page 114
- “BART Rules File Format” on page 115
- “BART Reporting” on page 116

BART Manifest File Format

Each manifest file entry is a single line, depending on the file type. Each entry begins with `fname`, which is the name of the file. To prevent parsing problems that are caused by special characters embedded in file names, the file names are encoded. For more information, see “BART Rules File Format” on page 115.

Subsequent fields represent the following file attributes:

- **type**: Type of file with the following possible values:
  - B for a block device node
  - C for a character device node
  - D for a directory
  - F for a file
  - L for a symbolic link
  - P for a pipe
  - S for a socket

- **size**: File size in bytes.
- **mode**: Octal number that represents the permissions of the file.
- **acl**: ACL attributes for the file. For a file with ACL attributes, this contains the output from `acl totext()`.
- **uid**: Numerical user ID of the owner of this entry.
- **gid**: Numerical group ID of the owner of this entry.
- **dirmtime**: Last modification time, in seconds, since 00:00:00 UTC, January 1, 1970, for directories.
- **lnmtime**: Last modification time, in seconds, since 00:00:00 UTC, January 1, 1970, for links.
- **mtime**: Last modification time, in seconds, since 00:00:00 UTC January 1, 1970, for files.
 checksum value of the file. This attribute is only specified for regular files. If you turn
off context checking, or if checksums cannot be computed, the value of this field is –.

 dest Destination of a symbolic link.

 devnode Value of the devicenode. This attribute is for character device files and block device
 files only.

 For more information about BART manifests, see the bart_manifest(4) man page.

 **BART Rules File Format**

 The input files to the bart command are text files. These files consist of lines that specify which files
 are to be included in the manifest and which file attributes are to be included in the report. The same
 input file can be used across both pieces of BART functionality. Lines that begin with #, blank lines,
 and lines that contain white space are ignored by the tool.

 The input files have three types of directives:

 - Subtree directive, with optional pattern matching modifiers
 - CHECK directive
 - IGNORE directive

 **EXAMPLE 5-8 Rules File Format**

 `<Global CHECK/IGNORE Directives>`
 `<subtree1> [pattern1..]`
 `<IGNORE/CHECK Directives for subtree1>`

 `<subtree2> [pattern2..]`
 `subtree3 [pattern3..]`
 `subtree4 [pattern4..]`
 `<IGNORE/CHECK Directives for subtree2, subtree3, subtree4>`

 **Note** – All directives are read in order, with later directives possibly overriding earlier directives.

 There is one subtree directive per line. The directive must begin with an absolute pathname, followed
 by zero or more pattern matching statements.

 **Rules File Attributes**

 The bart command uses CHECK and IGNORE statements to define which attributes to track or ignore.
 Each attribute has an associated keyword.

 The attribute keywords are as follows:
- acl
- all
- contents
- dest
- devnode
- dirmtime
- gid
- lnmtime
- mode
- mtime
- size
- type
- uid

The all keyword refers to all file attributes.

**Quoting Syntax**

The rules file specification language that BART uses is the standard UNIX quoting syntax for representing nonstandard filenames. Embedded tab, space, newline, or special characters are encoded in their octal forms to enable the tool to read file names. This nonuniform quoting syntax prevents certain filenames, such as those containing an embedded carriage return, from being processed correctly in a command pipeline. The rules specification language allows the expression of complex file name filtering criteria that would be difficult and inefficient to describe by using shell syntax alone.

For more information about the BART rules file or the quoting syntax used by BART, see the bart_rules(4) man page.

**BART Reporting**

In default mode, the bart compare command, as shown in the following example, will check all the files installed on the system, with the exception of modified directory timestamps (dirmtime):

```plaintext
CHECK all
IGNORE  dirmtime
```

If you supply a rules file, then the global directives of CHECK all and IGNORE dirmtime, in that order, are automatically prepended to the rules file.

**BART Output**

The following exit values are returned:

0       Success
1 Nonfatal error when processing files, such as permission problems

>1 Fatal error, such as an invalid command-line option

The reporting mechanism provides two types of output: verbose and programmatic:

- Verbose output is the default output and is localized and presented on multiple lines. Verbose output is internationalized and is human-readable. When the bart compare command compares two system manifests, a list of file differences is generated.

  For example:

  `filename attribute control:xxxx test:yyyy`

  `filename` Name of the file that differs between the control manifest and the test manifest.

  `attribute` Name of the file attribute that differs between the manifests that are compared. `xxxx` is the attribute value from the control manifest, and `yyyy` is the attribute value from the test manifest. When discrepancies for multiple attributes occur in the same file, each difference is noted on a separate line.

Following is an example of the default output for the bart compare command. The attribute differences are for the `/etc/passwd` file. The output indicates that the size, mtime, and contents attributes have changed.

  `/etc/passwd`:

  `size control:74 test:81`

  `mtime control:3c165879 test:3c165979`

  `contents control:daca28ae80de97afdf7a6b91fde8d57afa test:84b2b32c4165887355317207b48a6ec7`

- Programmatic output is generated if you use the `-p` option when you run the bart compare command. This output is generated in a form that is suitable for programmatic manipulation. Programmatic output can be easily parsed by other programs and is designed to be used as input for other tools.

  For example:

  `filename attribute control-val test-val [attribute control-val test-val]*`

  `filename` Same as the `filename` attribute in the default format

  `attribute control-val test-val` A description of the file attributes that differ between the control and test manifests for each file

For a list of attributes that are supported by the bart command, see “Rules File Attributes” on page 115.

For more information about BART, see the `bart(1M)` man page.
Controlling Access to Files (Tasks)

This chapter describes how to protect files in the Solaris Operating System (Solaris OS). The chapter also describes how to protect against files whose permissions could compromise the system.

The following is a list of the information in this chapter.

- “Using UNIX Permissions to Protect Files” on page 119
- “Using Access Control Lists to Protect Files” on page 125
- “Preventing Executable Files From Compromising Security” on page 128
- “Protecting Files (Task Map)” on page 128
- “Protecting Files With UNIX Permissions (Task Map)” on page 129
- “Protecting Files With ACLs (Task Map)” on page 134
- “Protecting Against Programs With Security Risk (Task Map)” on page 140

Using UNIX Permissions to Protect Files

Files can be secured through UNIX file permissions and through ACLs. Files with sticky bits, and files that are executable, require special security measures.

Commands for Viewing and Securing Files

This table describes the commands for monitoring and securing files and directories.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ls</td>
<td>Lists the files in a directory and information about the files.</td>
<td>\ls(1)</td>
</tr>
<tr>
<td>chown</td>
<td>Changes the ownership of a file.</td>
<td>chown(1)</td>
</tr>
</tbody>
</table>
TABLE 6–1 Commands for Securing Files and Directories (Continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>chgrp</td>
<td>Changes the group ownership of a file.</td>
<td>chgrp(1)</td>
</tr>
<tr>
<td>chmod</td>
<td>Changes permissions on a file. You can use either symbolic mode, which uses letters and symbols, or absolute mode, which uses octal numbers, to change permissions on a file.</td>
<td>chmod(1)</td>
</tr>
</tbody>
</table>

File and Directory Ownership

Traditional UNIX file permissions can assign ownership to three classes of users:

- **user** – The file or directory owner, which is usually the user who created the file. The owner of a file can decide who has the right to read the file, to write to the file (make changes to it), or, if the file is a command, to execute the file.
- **group** – Members of a group of users.
- **others** – All other users who are not the file owner and are not members of the group.

The owner of the file can usually assign or modify file permissions. Additionally, users or roles with administrative capabilities, such as superuser or the Primary Administrator role, can change a file’s ownership. To override system policy, see Example 6–2.

A file can be one of seven types. Each type is displayed by a symbol:

- **(Minus symbol)** Text or program
- **b** Block special file
- **c** Character special file
- **d** Directory
- **l** Symbolic link
- **s** Socket
- **D** Door
- **P** Named pipe (FIFO)

UNIX File Permissions

The following table lists and describes the permissions that you can give to each class of user for a file or directory.
TABLE 6–2 File and Directory Permissions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Permission</th>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>Read</td>
<td>File</td>
<td>Designated users can open and read the contents of a file. Directory Designated users can list files in the directory.</td>
</tr>
<tr>
<td>w</td>
<td>Write</td>
<td>File</td>
<td>Designated users can modify the contents of the file or delete the file. Directory Designated users can add files or add links in the directory. They can also remove files or remove links in the directory.</td>
</tr>
<tr>
<td>x</td>
<td>Execute</td>
<td>File</td>
<td>Designated users can execute the file, if it is a program or shell script. They also can run the program with one of the exec (2) system calls. Directory Designated users can open files or execute files in the directory. They also can make the directory and the directories beneath it current.</td>
</tr>
<tr>
<td>-</td>
<td>Denied</td>
<td>File and Directory</td>
<td>Designated users cannot read, write, or execute the file.</td>
</tr>
</tbody>
</table>

These file permissions apply to regular files, and to special files such as devices, sockets, and named pipes (FIFOs).

For a symbolic link, the permissions that apply are the permissions of the file that the link points to.

You can protect the files in a directory and its subdirectories by setting restrictive file permissions on 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: `setuid`, `setgid`, and sticky bit. When these permissions are set, any user who runs that executable file assumes the ID of the owner (or group) of the executable file.

You must be extremely careful when you set special permissions, because special permissions constitute a security risk. For example, a user can gain superuser capabilities by executing a program that sets the user ID (UID) to 0, which is the UID of `root`. Also, all users can set special permissions for files that they own, which constitutes another security concern.

You should monitor your system for any unauthorized use of the `setuid` permission and the `setgid` permission to gain superuser capabilities. A suspicious permission grants ownership of an administrative program to a user rather than to `root` or `bin`. To search for and list all files that use this special permission, see “How to Find Files With Special File Permissions” on page 140.
setuid Permission

When setuid permission is set on an executable file, a process that runs this file is granted access on the basis of the owner of the file. The access is not based on the user who is running the executable file. This special permission allows a user to access files and directories that are normally available only to the owner.

For example, the setuid permission on the passwd command makes it possible for users to change passwords. A passwd command with setuid permission would resemble the following:
```
-r-sr-sr-x 3 root sys 28144 Jun 17 12:02 /usr/bin/passwd
```

This special permission presents a security risk. Some determined users can find a way to maintain the permissions that are granted to them by the setuid process even after the process has finished executing.

Note – The use of setuid permissions with the reserved UIDs (0–100) from a program might not set the effective UID correctly. Use a shell script, or avoid using the reserved UIDs with setuid permissions.

setgid Permission

The setgid permission is similar to the setuid permission. The process’s effective group ID (GID) is changed to the group that owns the file, and a user is granted access based on the permissions that are granted to that group. The /usr/bin/mail command has setgid permissions:
```
-r-x-s-s-x 1 root mail 67504 Jun 17 12:01 /usr/bin/mail
```

When the setgid permission is applied to a directory, files that were created in this directory belong to the group to which the directory belongs. The files do not belong to 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 that owns the directory, not to the group that the user belongs to.

You should monitor your system for any unauthorized use of the setgid permission to gain superuser capabilities. A suspicious permission grants group access to such a program to an unusual group rather than to root or bin. To search for and list all files that use this permission, see “How to Find Files With Special File Permissions” on page 140.

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 file owner, the directory owner, or by a privileged user. The root user and the Primary Administrator role are examples of privileged users. The sticky bit prevents a user from deleting other users’ files from public directories such as /tmp:
```
drwxrwxrwx@ 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. For instructions, see Example 6–5.

**Default umask Value**

When you create a file or directory, you create it with a default set of permissions. The system defaults are open. A text file has 666 permissions, which grants read and write permission to everyone. A directory and an executable file have 777 permissions, which grants read, write, and execute permission to everyone. Typically, users override the system defaults in their /etc/profile file, .cshrc file, or .login file.

The value assigned by the umask command is subtracted from the default. This process has the effect of denying permissions in the same way that the chmod command grants them. For example, the chmod 022 command grants write permission to group and others. The umask 022 command denies write permission to group and others.

The following table shows some typical umask settings and their effect on an executable file.

<table>
<thead>
<tr>
<th>Level of Security</th>
<th>umask Setting</th>
<th>Permissions Disallowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissive (744)</td>
<td>022</td>
<td>w for group and others</td>
</tr>
<tr>
<td>Moderate (740)</td>
<td>027</td>
<td>w for group, rwx for others</td>
</tr>
<tr>
<td>Moderate (741)</td>
<td>026</td>
<td>w for group, rw for others</td>
</tr>
<tr>
<td>Severe (700)</td>
<td>077</td>
<td>rwx for group and others</td>
</tr>
</tbody>
</table>

For more information on setting the umask value, see the umask(1) man page.

**File Permission Modes**

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. When you change permissions by using the absolute mode, you represent permissions for each triplet by an octal mode number. Absolute mode is the method most commonly used to set permissions.

- **Symbolic Mode** – Use combinations of letters and symbols to add permissions or remove permissions.
The following table 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 and write permissions for owner, and read-only permissions for group and other.

**TABLE 6-4 Setting File Permissions in Absolute Mode**

<table>
<thead>
<tr>
<th>Octal Value</th>
<th>File Permissions Set</th>
<th>Permissions Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>---</td>
<td>No permissions</td>
</tr>
<tr>
<td>1</td>
<td>--x</td>
<td>Execute permission only</td>
</tr>
<tr>
<td>2</td>
<td>-w-</td>
<td>Write permission only</td>
</tr>
<tr>
<td>3</td>
<td>-wx</td>
<td>Write and execute permissions</td>
</tr>
<tr>
<td>4</td>
<td>r--</td>
<td>Read permission only</td>
</tr>
<tr>
<td>5</td>
<td>r-x</td>
<td>Read and execute permissions</td>
</tr>
<tr>
<td>6</td>
<td>rw-</td>
<td>Read and write permissions</td>
</tr>
<tr>
<td>7</td>
<td>rwx</td>
<td>Read, write, and execute permissions</td>
</tr>
</tbody>
</table>

The following table 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 that are being assigned or changed.

**TABLE 6-5 Setting File Permissions in Symbolic Mode**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>who</td>
<td>User (owner)</td>
</tr>
<tr>
<td>g</td>
<td>who</td>
<td>Group</td>
</tr>
<tr>
<td>o</td>
<td>who</td>
<td>Others</td>
</tr>
<tr>
<td>a</td>
<td>who</td>
<td>All</td>
</tr>
<tr>
<td>=</td>
<td>operator</td>
<td>Assign</td>
</tr>
<tr>
<td>+</td>
<td>operator</td>
<td>Add</td>
</tr>
<tr>
<td>-</td>
<td>operator</td>
<td>Remove</td>
</tr>
<tr>
<td>r</td>
<td>permissions</td>
<td>Read</td>
</tr>
<tr>
<td>w</td>
<td>permissions</td>
<td>Write</td>
</tr>
<tr>
<td>x</td>
<td>permissions</td>
<td>Execute</td>
</tr>
</tbody>
</table>
TABLE 6–5 Setting File Permissions in Symbolic Mode  (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>permissions</td>
<td>Mandatory locking, setgid bit is on, group execution bit is off</td>
</tr>
<tr>
<td>s</td>
<td>permissions</td>
<td>setuid or setgid bit is on</td>
</tr>
<tr>
<td>t</td>
<td>permissions</td>
<td>Sticky bit is on, execution bit for others is on</td>
</tr>
</tbody>
</table>

The **who** operator permissions designations in the function column specify the symbols that change the permissions on the file or directory.

- **who**: Specifies whose permissions are to be changed.
- **operator**: Specifies the operation to be performed.
- **permissions**: Specifies what permissions are to be changed.

You can set special permissions on a file in absolute mode or symbolic mode. However, you must use symbolic mode to set or remove setuid permissions on a directory. In absolute mode, you set special permissions by adding a new octal value to the left of the permission triplet. The following table lists the octal values for setting special permissions on a file.

TABLE 6–6 Setting Special File Permissions in Absolute Mode

<table>
<thead>
<tr>
<th>Octal Value</th>
<th>Special File Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sticky bit</td>
</tr>
<tr>
<td>2</td>
<td>setgid</td>
</tr>
<tr>
<td>4</td>
<td>setuid</td>
</tr>
</tbody>
</table>

**Using Access Control Lists to Protect Files**

Traditional UNIX file protection provides read, write, and execute permissions for the three user classes: file owner, file group, and other. An access control list (ACL) provides better file security by enabling you to do the following:

- Define file permissions for the file owner, the group, other, specific users and groups
- Define default permissions for each of the preceding categories

For example, if you want everyone in a group to be able to read a file, you can simply grant group read permissions on that file. Now, assume that you want only one person in the group to be able to write to that file. Standard UNIX does not provide that level of file security. However, an ACL provides this level of file security.

ACL entries define an ACL on a file. The entries are set through the `setfacl` command. ACL entries consist of the following fields separated by colons:
entry-type: [uid]:gid:perms

**entry-type**
Is the type of ACL entry on which to set file permissions. For example, *entry-type* can be *user* (the owner of a file) or *mask* (the ACL mask). For a listing of ACL entries, see Table 6–7 and Table 6–8.

**uid**
Is the user name or user ID (UID).

**gid**
Is the group name or group ID (GID).

**perms**
Represents the permissions that are set on *entry-type*. *perms* can be indicated by the symbolic characters *rwx* or an octal number. These are the same numbers that are used with the *chmod* command.

In the following example, an ACL entry sets read and write permissions for the user *stacey*.

user:stacey:rw-

---

**Caution** – UFS file system attributes such as ACLs are supported in UFS file systems only. Thus, 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 following table lists the valid ACL entries that you might use when setting ACLs on files. The first three ACL entries provide the basic UNIX file protection.

<table>
<thead>
<tr>
<th>ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u[ser]:perms</td>
<td>File owner permissions.</td>
</tr>
<tr>
<td>g[roup]:perms</td>
<td>File group permissions.</td>
</tr>
<tr>
<td>o[ther]:perms</td>
<td>Permissions for users other than the file owner or members of the file group.</td>
</tr>
<tr>
<td>m[ask]:perms</td>
<td>The ACL mask. The mask entry indicates the maximum permissions that are 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 <em>mask: r--</em> mask entry indicates that users and groups cannot have more than read permissions, even though they might have write and execute permissions.</td>
</tr>
<tr>
<td>u[ser]:uid:perms</td>
<td>Permissions for a specific user. For <em>uid</em>, you can specify either a user name or a numeric UID.</td>
</tr>
</tbody>
</table>
TABLE 6–7 ACL Entries for Files  (Continued)

<table>
<thead>
<tr>
<th>ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g[roup]:gid:perms</td>
<td>Permissions for a specific group. For gid, you can specify either a group name or a numeric GID.</td>
</tr>
</tbody>
</table>

ACL Entries for Directories

In addition to the ACL entries that are described in Table 6–7, 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. Table 6–8 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 entries are required. They are the first four default ACL entries in the following table.

TABLE 6–8 Default ACL Entries for Directories

<table>
<thead>
<tr>
<th>Default ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[efault]:u[ser]:perms</td>
<td>Default file owner permissions.</td>
</tr>
<tr>
<td>d[efault]:g[roup]:perms</td>
<td>Default file group permissions.</td>
</tr>
<tr>
<td>d[efault]:o[ther]:perms</td>
<td>Default permissions for users other than the file owner or members of the file group.</td>
</tr>
<tr>
<td>d[efault]:m[ask]:perms</td>
<td>Default ACL mask.</td>
</tr>
<tr>
<td>d[efault]:u[ser]:uid:perms</td>
<td>Default permissions for a specific user. For uid, you can specify either a user name or a numeric UID.</td>
</tr>
<tr>
<td>d[efault]:g[roup]:gid:perms</td>
<td>Default permissions for a specific group. For gid, you can specify either a group name or a numeric GID.</td>
</tr>
</tbody>
</table>

Commands for Administering ACLs

The following commands administer ACLs on files or directories.

**setfacl command**  Sets, adds, modifies, and deletes ACL entries. For more information, see the `setfacl(1)` man page.

**getfacl command**  Displays ACL entries. For more information, see the `getfacl(1)` man page.
Preventing Executable Files From Compromising 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 are allowed, most programs can function correctly without using executable stacks.

The `noexec_user_stack` variable enables you to specify whether stack mappings are executable. The variable is available as of the Solaris 2.6 release. By default, the variable is set to zero, except on 64-bit applications, which provides ABI-compliant behavior. If the variable is set to a non-zero value, 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. This signal 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 the 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 `syslog` daemon when the `syslog_kern` facility is set to `notice` level. This logging is set by default in the `syslog.conf` file, which means that the message is sent to both the console and the `/var/adm/messages` file. For more information, see the `syslogd(1M)` and `syslog.conf(4)` man pages.

The `syslog` message is useful for observing potential security problems. The message also identifies valid programs that depend upon executable stacks that have been prevented from correct operation by setting this variable. If you do not want any messages logged, then set the `noexec_user_stack_log` variable to zero in the `/etc/system` file. Even though messages are not being logged, the SIGSEGV signal can continue to cause the executing program to terminate with a core dump.

You can use the `mprotect()` function if you want programs to explicitly mark their stack as executable. For more information, see the `mprotect(2)` man page.

Because of hardware limitations, the capability of catching and reporting executable stack problems is not available on most x86 based systems. Systems in the AMD64 product family can catch and report executable stack problems.

Protecting Files (Task Map)

The following task map points to sets of procedures for protecting files.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use UNIX permissions to protect files</td>
<td>Views UNIX permissions on files. Protects files with UNIX permissions.</td>
<td>&quot;Protecting Files With UNIX Permissions (Task Map)&quot; on page 129</td>
</tr>
</tbody>
</table>
Use ACLs to protect files

Add ACLs to protect files at a more granular level than UNIX permissions can.

"Protecting Files With ACLs (Task Map)" on page 134

Protect system from files that pose a security risk

Finds executable files that have suspicious ownership. Disables files that can damage the system.

"Protecting Against Programs With Security Risk (Task Map)" on page 140

### Protecting Files With UNIX Permissions (Task Map)

The following task map points to procedures that list file permissions, change file permissions, and protect files with special file permissions.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display file information</td>
<td>&quot;How to Display File Information&quot; on page 129</td>
<td></td>
</tr>
<tr>
<td>Change file ownership</td>
<td>&quot;How to Change the Owner of a File&quot; on page 130</td>
<td>&quot;How to Change Group Ownership of a File&quot; on page 131</td>
</tr>
<tr>
<td>Change file permissions</td>
<td>&quot;How to Change File Permissions in Symbolic Mode&quot; on page 132</td>
<td>&quot;How to Change File Permissions in Absolute Mode&quot; on page 132</td>
</tr>
<tr>
<td></td>
<td>&quot;How to Change Special File Permissions in Absolute Mode&quot; on page 133</td>
<td></td>
</tr>
</tbody>
</table>

### How to Display File Information

Display information about all the files in a directory by using the `ls` command.

- **Type the following command to display a long listing of all files in the current directory.**
  
  ```
  % ls -la
  ``

  - `-l` Displays the long format that includes user ownership, group ownership, and file permissions.
  - `-a` Displays all files, including hidden files that begin with a dot (`.`).

#### Example 6–1 Displaying File Information

In the following example, a partial list of the files in the `/sbin` directory is displayed.

```
% 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 ..
```
Each line displays information about a file in the following order:

- **Type of file** – For example, d. For list of file types, see “File and Directory Ownership” on page 120.
- **Permissions** – For example, r-xr-xr-x. For description, see “File and Directory Ownership” on page 120.
- **Number of hard links** – For example, 2.
- **Owner of the file** – For example, root.
- **Group of the file** – For example, bin.
- **Size of the file, in bytes** – For example, 7696.
- **Date the file was created or the last date that the file was changed** – For example, Aug 18 15:20.
- **Name of the file** – For example, mountall.

### ▼ How to Change the Owner of a File

The file owner, the Primary Administrator role, or superuser can change any file's ownership.

1. **Display the permissions on a file.**
   ```
   % ls -l example-file
   -rw-r--r-- 1 janedoe  staff  112640 May 24 10:49 example-file
   ```

2. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*. 
3 Change the owner of the file.
   # chown stacey example-file

4 Verify that the owner of the file has changed.
   # ls -l example-file
   -rw-r--r-- 1 stacey staff 112640 May 26 08:50 example-file

Example 6–2 Enabling Users to Change the Ownership of Files That Others Own

Security Consideration – You should have good reason to override system security policy by changing the setting of the `rstchown` variable to zero. Any user who accesses the system can change the ownership of any file on the system.

In this example, the value of the `rstchown` variable is set to zero in the `/etc/system` file. This setting enables the owner of a file to use the `chown` command to change the file’s ownership to another user. This setting also enables the owner to use the `chgrp` command to set the group ownership of a file to a group that the owner does not belong to. The change goes into effect when the system is rebooted.

```set rstchown = 0```

For more information, see the `chown(1)` and `chgrp(1)` man pages.

Also, be aware that NFS-mounted file systems have further restrictions on changing ownership and groups. For more information on restricting access to NFS-mounted systems, see Chapter 6, “Accessing Network File Systems (Reference),” in System Administration Guide: Network Services.

How to Change Group Ownership of a File

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Change the group ownership of a file.
   $ chgrp scifi example-file
   For information on setting up groups, see Chapter 4, “Managing User Accounts and Groups (Overview),” in System Administration Guide: Basic Administration.

3 Verify that the group ownership of the file has changed.
   $ ls -l example-file
   -rw-r--r-- 1 stacey scifi 112640 June 20 08:55 example-file
   Also see Example 6–2.
How to Change File Permissions in Symbolic Mode

1. If you are not the owner of the file or directory, become superuser or assume an equivalent role.
   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.
   ```
   % chmod who operator permissions filename
   ```
   - **who**: Specifies whose permissions are to be changed.
   - **operator**: Specifies the operation to be performed.
   - **permissions**: Specifies what permissions are to be changed. For the list of valid symbols, see Table 6–5.
   - **filename**: Specifies the file or directory.

3. Verify that the permissions of the file have changed.
   ```
   % ls -l filename
   ```

Example 6–3
Changing Permissions in Symbolic Mode

In the following example, read permission is taken away from others.

```
% chmod o-r example-file1
```

In the following example, read and execute permissions are added for user, group, and others.

```
$ chmod a+rx example-file2
```

In the following example, read, write, and execute permissions are assigned to group.

```
$ chmod g=rx example-file3
```

How to Change File Permissions in Absolute Mode

1. If you are not the owner of the file or directory, become superuser or assume an equivalent role.
   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.
   ```
   % chmod nnn filename
   ```
   - **nnn**: Specifies the octal values that represent the permissions for the file owner, file group, and others, in that order. For the list of valid octal values, see Table 6–4.
filename  

Specifies the file or directory.

**Note** - When you use the `chmod` command 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 permissions for other users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl(1)` man page.

3  **Verify that the permissions of the file have changed.**  

```
% ls -l filename
```

**Example 6–4 Changing Permissions in Absolute Mode**

In the following example, the permissions of a public directory are changed from 744 (read, write, execute; read-only; and read-only) to 755 (read, write, execute; read and execute; and read and 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
```

In the following example, the permissions of an executable shell script are changed from read and write to read, write, and 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 File Permissions in Absolute Mode**

1  **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**  

Only the current owner or a user with superuser capabilities can use the `chmod` command to change the special permissions on a file or directory.

2  **Change special permissions in absolute mode.**  

```
% chmod mmmm filename
```
nnnn  Specifies the octal values that change the permissions on the file or directory. The leftmost octal value sets the special permissions on the file. For the list of valid octal values for special permissions, see Table 6–6.

filename  Specifies the file or directory.

Note – When you use the chmod command 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 permissions for additional users and groups who have ACL entries on the file. Use the getfacl command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the getfacl(1) man page.

3  Verify that the permissions of the file have changed.
   % ls -l filename

Example 6–5  Setting Special File Permissions in Absolute Mode

In the following example, the setuid permission is set on the dbprog file.

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x 1 db staff 12095 May 6 09:29 dbprog
```

In the following example, the setgid permission is set on the dbprog2 file.

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x 1 db staff 24576 May 6 09:30 dbprog2
```

In the following example, the sticky bit permission is set on the public_dir directory.

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt 2 ignatz staff 512 May 15 15:27 public_dir
```

Protecting Files With ACLs (Task Map)

The following task map points to procedures that list the ACLs on a file, change the ACLs, and copy the ACLs to another file.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine if a file has an ACL</td>
<td>“How to Check if a File Has an ACL” on page 135</td>
</tr>
</tbody>
</table>
### How to Check if a File Has an ACL

Check if a file has an ACL.

```
% ls -l filename
```

where `filename` specifies the file or directory.

In the output, a plus sign (+) to the right of the mode field indicates that the file has an ACL.

**Note** – Unless you have added ACL entries that extend UNIX file permissions, a file is considered to have a “trivial” ACL and the plus sign (+) does not display.

### Example 6–6 Checking if a File Has an ACL

In the following example, the `ch1.sgm` file has an ACL. The ACL is indicated by the plus sign (+) to the right of the mode field.

```
% ls -l ch1.sgm
-rwxr-----+ 1 stacey techpubs 167 Nov 11 11:13 ch1.sgm
```

### How to Add ACL Entries to 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 ...
   
   -s            Sets an ACL on the file. If a file already has an ACL, it is replaced. This option requires at least the `user::`, `group::`, and `other::` entries.
   
   user::perms  Specifies the file owner permissions.
   
   group::perms Specifies the group ownership permissions.
   
   other::perms Specifies the permissions for users other than the file owner or members of the group.
   ```
mask:perms  Specifies the permissions for the ACL mask. The mask indicates the maximum permissions that are allowed for users (other than the owner) and for groups.

acl-entry-list  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 6–7 and Table 6–8 show the valid ACL entries.

filename ...  Specifies one or more files or directories on which to set the ACL. Multiple filenames are separated by spaces.

Caution – If an ACL already exists on the file, the -s option replaces the entire ACL with the new ACL.

For more information, see the setfacl(1) man page.

2 Verify that the ACL entries were set on the file.

% getfacl filename

For more information, see “How to Check if a File Has an ACL” on page 135.

Example 6–7  Setting an ACL on a File

In the following example, the file owner permissions are set to read and write, file group permissions are set to read only, and other permissions are set to none on the ch1.sgm file. In addition, the user anusha is given read and write permissions on the file. The ACL mask permissions are set to read and write, which means that no user or group can have execute permissions.

% setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:anusha:rw- ch1.sgm
% ls -l
total 124
-rw-r-----+ 1 stacey techpubs 34816 Nov 11 14:16 ch1.sgm
-rw-r--r-- 1 stacey techpubs 20167 Nov 11 14:16 ch2.sgm
-rw-r--r-- 1 stacey techpubs 8192 Nov 11 14:16 notes
% getfacl ch1.sgm
# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:rw-  #effective:rw-
group::r--     #effective:r--
mask:rw-
other:---

In the following example, the file owner permissions are set to read, write, and execute, file group permissions are set to read only, other permissions are set to none. In addition, the ACL mask
permissions are set to read on the ch2.sgm file. Finally, the user anusha is given read and write permissions. However, due to the ACL mask, the permissions for anusha are read only.

% setfacl -s u::7,g::4,o:0,m:4,u:anusha:7 ch2.sgm
% getfacl ch2.sgm
# file: ch2.sgm
# owner: stacey
# group: techpubs
user::rwx
user:anusha: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 6–8 Copying an ACL

In the following example, the ACL on ch2.sgm is copied to ch3.sgm.

% getfacl ch2.sgm | setfacl -f - ch3.sgm

▼ How to Change ACL Entries on a File

1 Modify ACL entries on a file by using the setfacl command.

% setfacl -m acl-entry-list filename ...

-m    Modifies the existing ACL entry.
acl-entry-list    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 6–7 and Table 6–8 show the valid ACL entries.
filename ...    Specifies one or more files or directories, separated by a space.

2 Verify that the ACL entries were modified on the file.

% getfacl filename
Example 6–9   Modifying ACL Entries on a File

In the following example, the permissions for the user anusha are modified to read and write.

```
% setfacl -m user:anusha:6 ch3.sgm
% getfacl ch3.sgm
# file: ch3.sgm
# owner: stacey
# group: techpubs
user::rw-
user::anusha:rw-   #effective:r--
group::r-          #effective:r--
mask:r--
other:r-
```

In the following example, the default permissions for the group staff are modified to read on the book directory. In addition, the default ACL mask permissions are modified to read and write.

```
% 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.

```
% setfacl -d acl-entry-list filename ...
```

- **d** Deletes the specified ACL entries.

```
acl-entry-list
```

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 6–7 and Table 6–8 show the valid ACL entries.

```
filename ...
```

Specifies one or more files or directories, separated by a space.

Alternatively, you can use the `setfacl -s` command to delete all the ACL entries on a file and replace them with the new ACL entries that are specified.

2   Verify that the ACL entries were deleted from the file.

```
% getfacl filename
```

Example 6–10   Deleting ACL Entries on a File

In the following example, the user anusha is deleted from the ch4.sgm file.

```
% setfacl -d user:anusha ch4.sgm
```
How to Display ACL Entries for a File

Display ACL entries for a file by using the `getfacl` command.

```bash
% getfacl [-a | -d] filename ...
```

- `-a` Displays the file name, file owner, file group, and ACL entries for the specified file or directory.
- `-d` Displays the file name, file owner, file group, and the default ACL entries, if they exist, for the specified directory.

`filename ...` Specifies one or more files or directories, separated by a space.

If you specify multiple file names on the command line, the ACL entries are displayed with a blank line between each entry.

Example 6–11 Displaying ACL Entries for a File

In the following example, all the ACL entries for the `ch1.sgm` file are displayed. The `#effective:` note beside the user and group entries indicates what the permissions are after being modified by the ACL mask.

```bash
% getfacl ch1.sgm

# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:r-   #effective:r--
group::rw-   #effective:rw-
mask:rw-
other:---
```

In the following example, the default ACL entries for the `book` directory are displayed.

```bash
% getfacl -d book

# file: book
# owner: stacey
# group: techpubs
user::rwx
user:anusha:r-x   #effective:r-x
group::rwx   #effective:rwx
mask:rwx
other:---
default:user::rw-
default:user:anusha:r--
default:group::rw-
```
Protecting Against Programs With Security Risk (Task Map)

The following task map points to procedures that find risky executables on the system, and that prevent programs from exploiting an executable stack.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find files with special permissions</td>
<td>Locates files with the setuid bit set, but that are not owned by the root user.</td>
<td>&quot;How to Find Files With Special File Permissions&quot; on page 140</td>
</tr>
<tr>
<td>Prevent executable stack from overflowing</td>
<td>Prevents programs from exploiting an executable stack.</td>
<td>&quot;How to Disable Programs From Using Executable Stacks&quot; on page 141</td>
</tr>
<tr>
<td>Prevent logging of executable stack messages</td>
<td>Turns off logging of executable stack messages.</td>
<td>Example 6–13</td>
</tr>
</tbody>
</table>

▼ How to Find Files With Special File Permissions

You should monitor your system for any unauthorized use of the setuid and setgid permissions on programs. The setuid and setgid permissions enable ordinary users to gain superuser capabilities. A suspicious executable file grants ownership to a user rather than to root or bin.

1 Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2 Find files with setuid permissions by using the find command.
   # find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/filename
   find directory Checks all mounted paths starting at the specified directory, which can be root (/), sys, bin, or mail.
   -user root Displays files owned only by root.
   -perm -4000 Displays files only with permissions set to 4000.
   -exec ls -ldb Displays the output of the find command in ls -ldb format.
   >/tmp/filename Is the file that contains the results of the find command.

3 Display the results in /tmp/filename.
   # more /tmp/filename
For background information on setuid permissions, see “setuid Permission” on page 122.

Example 6–12 Finding Files With setuid Permissions

The output from the following example shows that a user named rar has made a personal copy of /usr/bin/sh, and has set the permissions as setuid to root. As a result, the /usr/rar/bin/sh program runs with root permissions.

This output was saved for future reference by moving the file out of the /tmp directory.

```
# find / -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/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 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
# mv /var/tmp/ckprm /export/sysreports/ckprm
```

How to Disable Programs From Using Executable Stacks

For a description of the security risks of executable stacks, see “Preventing Executable Files From Compromising Security” on page 128.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Edit the /etc/system file, and add the following line:

```
set noexec_user_stack=1
```

3 Reboot the system.

```
# init 6
```
Example 6–13  Disabling the Logging of Executable Stack Messages

In this example, the logging of executable stack messages is disabled, and then the system is rebooted.

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# init 6
```
Using the 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.

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

- “Automated Security Enhancement Tool (ASET)” on page 143
- “Running ASET (Task Map)” on page 160
- “Troubleshooting ASET Problems” on page 164

For a more comprehensive tool than ASET, use the Solaris Security Toolkit. The Solaris Security Toolkit provides a framework for hardening and minimizing a Solaris system. The kit includes a profiling tool, a reporting tool, and an undo capability. The toolkit is free, and can be downloaded from the Sun web site, http://www.s.net.com/security/jass. The web site contains pointers to online documentation.


Automated Security Enhancement Tool (ASET)

The Solaris Operating System includes the Automated Security Enhancement Tool (ASET). ASET helps you to monitor and to 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 at which to run ASET. The security levels are low, medium, and high. At each higher level, ASET’s file-control functions increase to reduce file access and tighten your system security.

There are seven tasks that ASET runs. Each task performs 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 also 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 147.

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. The report notes detected security weaknesses and any changes that the task has made to the system files. When run at the highest security level, ASET attempts to modify all system security weaknesses. If ASET cannot correct a potential security problem, ASET reports the existence of the problem.

You can initiate an ASET session by using the /usr/aset/aset command interactively. Or, you can set up ASET to run periodically by putting an entry into the crontab file.

ASET tasks are disk-intensive. The tasks can interfere with regular activities. To minimize the impact on system performance, schedule ASET to run when system activity level is lowest. For example, run ASET 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 following table outlines these three levels of security.

<table>
<thead>
<tr>
<th>Security Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Ensures that attributes of system files are set to standard release values. ASET performs several checks, then reports potential security weaknesses. At this level, ASET takes no action, so ASET does not affect system services.</td>
</tr>
<tr>
<td>Medium</td>
<td>Provides adequate security control for most environments. ASET modifies some settings of system files and parameters. ASET restricts system access to reduce the risks from security attacks. ASET reports security weaknesses and any modifications that ASET has made to restrict access. At this level, ASET does not affect system services.</td>
</tr>
<tr>
<td>High</td>
<td>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. However, at this level, security considerations take precedence over other system behavior.</td>
</tr>
</tbody>
</table>
Note – ASET does not change the permissions of a file to make the file less secure, unless you
downgrade the security level. You could also intentionally revert the system to the settings that
existed prior to running ASET.

ASET Task List

This section discusses what ASET does. You should understand each ASET task. By understanding
the objectives of ASET, the operations that ASET performs, and the system components that ASET
affects, you can interpret and use the reports effectively.

ASET report files contain messages that describe as specifically as possible any problems that were
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 novice administrator, you can refer to other
Solaris system administration documentation. You can read related manual pages to prepare yourself
for ASET administration.

The taskstat utility identifies the tasks that have been completed. The utility also identifies the tasks
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 Tuning

This task sets the permissions on system files to the security level that you designate. This task is run
when the system is installed. If you decide later to alter the previously established levels, then run this
task again. At low security, permissions are set to values that are appropriate for an open
information-sharing environment. At medium security, permissions are tightened to produce
adequate security for most environments. At high security, permissions 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. For an example of the files that ASET consults when ASET sets permissions, see
“Tune File Examples” on page 158.

System Files Checks

This task examines system files and compares each file with a description of that file in a master file.
The master file is created the first time ASET runs this task. The master file contains the system file
settings that are 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 the list, 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 that ASET finds 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 and Group Checks**

This task checks the consistency and integrity of user accounts and groups. The task uses the definitions in the passwd and group files. This task checks the local, and NIS or NIS+ password files. Password file problems for NIS+ 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 server or an NIS+ server

Discrepancies are reported in the usrgroup.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 the following:

- /etc/default/login
- /etc/hosts.equiv
- /etc/inetd.conf
- /etc/aliases
- /var/adm/utmpx
- /.rhosts
- /etc/vfstab
- /etc/dfs/dfstab
- /etc/ftp/ftpusers

ASET performs various checks and various modifications on these files. ASET reports problems in the sysconf.rpt file.

**Environment Variables Check**

This task checks how the PATH and UMASK environment variables are set for root, and for other users. The task checks 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 the parameter 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. This task 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 54. The firewall system separates two networks. In this situation, each network approaches the other network as untrusted. The firewall setup task disables the forwarding of Internet Protocol (IP) packets. The firewall also 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. You eliminate the task by editing the `asetenv` file.

Any changes that are made are reported in the `firewall.rpt` file.

**ASET Execution Log**

ASET generates an execution log whether ASET 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, see the `aset(1M)` man page.

**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 execution log first shows the system and time that ASET was run. Then, the execution log lists each task as the task was started.

ASET invokes a background process for each of these tasks, which are described in “ASET Task List” on page 145. The task is listed in the execution log when the task starts. This listing does not indicate that the task completed. To check the status of the background tasks, use the taskstat command.

**ASET Reports**

All report files that are generated from ASET tasks are stored 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 convention enables you to keep an orderly trail of records that document the system status as the status varies between ASET executions. You can monitor and compare these reports to determine the soundness of your system’s security.

The following figure shows an example of the `reports` directory structure.
This example shows two report subdirectories.

- 0124_01:00
- 0125_01:00

The subdirectory names indicate the date and time that the reports were generated. Each report subdirectory name has the following format:

\textit{monthdate\_hour:minute}

\textit{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 that are generated from one execution of ASET.

The \textit{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 the report. The following table lists tasks and their reports.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>System files permissions tuning (tune)</td>
<td>tune.rpt</td>
</tr>
<tr>
<td>System files checks (cklist)</td>
<td>cklist.rpt</td>
</tr>
<tr>
<td>User and group checks (usrgrp)</td>
<td>usrgrp.rpt</td>
</tr>
<tr>
<td>System configuration files check (sysconf)</td>
<td>sysconf.rpt</td>
</tr>
<tr>
<td>Environment variables check (env)</td>
<td>env.rpt</td>
</tr>
<tr>
<td>eeprom check (eeprom)</td>
<td>eeprom.rpt</td>
</tr>
<tr>
<td>Firewall setup (firewall)</td>
<td>firewall.rpt</td>
</tr>
</tbody>
</table>

Within each report file, messages are bracketed by a beginning and an ending banner line. Sometimes, a task ends prematurely. For example, a task can end prematurely when a component of ASET is accidentally removed or damaged. In such cases, the report file usually contains a message near the end that indicates the reason for the premature termination.

The following is a sample report file, usgrp.rpt.

```plaintext
*** 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 you initially run or reconfigure ASET, you should examine the report files closely. Reconfiguration includes modifying the ase t env file or the master files in the masters subdirectory, or changing the security level at which ASET operates.

The reports record any errors that were introduced when you reconfigured ASET. By watching the reports closely, you can react to, and solve, problems as the problems 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 begins to stabilize. When the reports contain little 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. For more detail, see the `asetmasters(4)` man page.

Tune Files

The `tune.low`, `tune.med`, and `tune.high` master files define the available ASET security levels. The files 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 that share the same user ID (UID). 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 UIDs if these entries are specified in the `uid_aliases` file.

Avoid having multiple user accounts share the same UID. 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. You could also create a role. The sharing of UIDs should be your last resort, used only when other methods cannot accomplish your objectives.

You can use the `UID_ALIASES` environment variable to specify an alternate aliases file. The default file is `/usr/aset/masters/uid_aliases`.

The Checklist Files

The master files that are used by the system files checks are generated when you first execute ASET. The master files are also generated when you run ASET after changing the security level.

The following environment variables define the files that are checked by this task:

- `CKLISTPATH_LOW`
- `CKLISTPATH_MED`
- `CKLISTPATH_HIGH`
**ASET Environment File (asetenv)**

The environment file, asetenv, contains a list of environment variables that affect ASET tasks. Some of these variables can be changed to modify ASET operation. For details about the asetenv file, see asetenv(4).

**Configuring ASET**

This section discusses how ASET is configured. This section also discusses the environment in which ASET operates.

ASET requires minimum administration and minimum configuration. In most cases, you can run ASET 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 you change the default values, you should understand how ASET works, and how ASET affects the components of your system.

ASET relies on four configuration files to control the 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 environment variables 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. These settings should not be modified.

You can edit the entries in the user-configurable section to do the following:

- Choose which tasks to run
- Specify the directories for the system files checks task
- Schedule ASET execution
- Specify a UID aliases file
- Extend checks to NIS+ tables

**Choosing Which Tasks to Run: TASKS**

Each task that 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 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 you do not require firewall protection.
You can set up ASET to run at the high security level without the firewall feature. To do so, edit the TASKS list of environment variables in the aset.env file. By default, the TASKS list contains all of the ASET tasks. To delete a task, remove the task-related environment variable from the file. In this case, you would delete the firewall environment variable from the list. The next time ASET runs, the excluded task is not performed.

In the following example, the TASKS list with all of the ASET tasks is displayed.

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

**Specifying Directories for System Files Checks Task:** `CKLISTPATH`

The system files check checks the attributes of files in selected system directories. You define which directories to check by using the following environment variables.

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 that is defined by an environment variable at a lower security level should be a subset of the directory list that is defined at the next higher level. For example, all directories that are specified for `CKLISTPATH_LOW` should be included in `CKLISTPATH_MED`. Similarly, all the directories that are specified for `CKLISTPATH_MED` should be included in `CKLISTPATH_HIGH`.

Checks that are performed on these directories are not recursive. ASET only checks those directories that are explicitly listed in the environment variable. ASET does not check their subdirectories.

You can edit these environment 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.

**Scheduling ASET Execution:** `PERIODIC_SCHEDULE`

You can start ASET interactively, or you can use the -p option to request that the ASET tasks run at a scheduled time. 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 the tasks. For detailed instructions about setting up ASET to run periodically, see “How to Run ASET Periodically” on page 161.

The format of `PERIODIC_SCHEDULE` follows the format of crontab entries. For complete information, see `crontab(1)`.

**Specifying an Aliases File:** `UID_ALIASES`

The `UID_ALIASES` variable specifies an aliases file that lists shared UIDs. The default file is `/usr/aset/masters/uid_aliases`.

---

**Automated Security Enhancement Tool (ASET)**

Chapter 7 • Using the Automated Security Enhancement Tool (Tasks) 153
Extending 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 YPCHECK. The default value is false, which disables NIS+ table checking.

To understand how this environment variable works, consider its effect on the passwd file. When set to false, ASET checks the local passwd file. When set to true, the task also checks the NIS+ passwd table for the domain of the system.

Note – Although ASET automatically repairs the local files, ASET only reports potential problems in the NIS+ tables. ASET does not change the tables.

Modifying the Tune Files

ASET uses the three master tune files, tune.low, tune.med, and tune.high, to ease or tighten access to critical system files. These master files are located in the /usr/aset/masters directory. You can modify the files to suit your environment. For examples, see “Tune File Examples” on page 158.

The tune.low file sets permissions to values that are appropriate for default system settings. The tune.med file further restricts these permissions. The tune.med file also includes entries that are not present in tune.low. The tune.high file restricts permissions even further.

Note – Modify settings in the tune files by adding or deleting file entries. You cannot effectively set a permission to a less restrictive value than the current setting. 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, ASET saves and archives the original system files. The aset.restore utility reinstates these files. This utility also deschedules ASET, if ASET is currently scheduled for periodic execution. The aset.restore command is located in /usr/aset, the ASET operating directory.

Changes that are made to system files are lost when you run the aset.restore command.

You should use the aset.restore command in the following instances:

- When you want to remove ASET changes and to restore the original system.
  If you want to deactivate ASET permanently, you can remove ASET from crontab scheduling if the aset command had previously been added to root’s crontab. For instructions on how to use crontab to remove automatic execution, see "How to Stop Running ASET Periodically" on page 162.
- After a brief period of experimenting with ASET, to restore the original system state.
When some major system feature is not working properly, and you suspect that ASET is causing the problem.

**Network Operation With 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. Therefore, you are responsible 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 your clients. To facilitate ASET management across several client systems, you can make configuration changes that are applied globally to all clients. By globally applying changes, you eliminate the need to log in to each system to repeat the configuration changes.

When you are deciding how to set up ASET on your networked systems, you should consider who you want to control security. You might want users to control some security on their own systems. You might want to centralize responsibility for security control.

**Providing a Global Configuration for Each Security Level**

A situation 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. You might want to set up another configuration for medium level clients, and yet another configuration 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. You create one configuration 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 by using links.

**Collecting ASET Reports**

Not only can you centralize the ASET components on a server, but you can also set up a central directory on a server to collect all ASET reports. The server can be accessed by clients with or without superuser privileges. For instructions on setting up a collection mechanism, see “How to Collect ASET Reports on a Server” on page 162.

By setting up the collection of reports on a server, you can review reports for all clients from one location. You can use this method whether or not a client has superuser privileges. 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 following is a list of the ASET environment variables and the values that the variables specify.
### ASETDIR Environment Variable

ASETDIR specifies an ASET working directory.

From the C shell, type:

```bash
% setenv ASETDIR pathname
```

From the Bourne shell or the Korn shell, type:

```bash
$ ASETDIR=pathname
$ export ASETDIR
```

Set `pathname` to the full path name of the ASET working directory.

### ASETSECLEVEL Environment Variable

The ASETSECLEVEL variable specifies a security level at which ASET tasks are executed.

From the C shell, type:

```bash
% setenv ASETSECLEVEL level
```

From the Bourne shell or the Korn shell, type:

```bash
$ ASETSECLEVEL=level
$ export ASETSECLEVEL
```

In these commands, `level` can be set to one of the following:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW</td>
<td>Is the directory list for low security</td>
</tr>
<tr>
<td>MED</td>
<td>Is the directory list for medium security</td>
</tr>
<tr>
<td>HIGH</td>
<td>Is the directory list for high security</td>
</tr>
</tbody>
</table>

The environment variables that are listed in the following sections are found in the `/usr/aset/asetenv` file. The ASETDIR and ASETSECLEVEL variables are optional. The variables can be set only through the shell by using the `/usr/aset/aset` command. The other environment variables can be set by editing the file.
PERIODIC_SCHEDULE Environment 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, with each field separated by a space:

"minutes hours day-of-month month day-of-week"

- **minutes hours** Specifies start time in number of minutes (0-59) after the hour and the hour (0-23).
- **day-of-month** Specifies the day of the month when ASET should be run, with values from 1-31.
- **month** Specifies the month of the year when ASET should be run, with values from 1-12.
- **day-of-week** Specifies the day of the week when ASET should be run, with values from 0-6. Sunday is day 0.

The following rules apply when creating a periodic schedule for ASET:
- 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 the value as a range. A range is a pair of numbers that are joined by a hyphen. A range states that the ASET tasks should be executed for every time that is included in the range.
- You can specify an asterisk (*) as the value of any field. An asterisk inclusively specifies all possible values of the field.

The default entry for the PERIODIC_SCHEDULE variable causes ASET to execute at 12:00 midnight every day:

PERIODIC_SCHEDULE="0 0 * * *"

TASKS Environment Variable
The TASKS variable lists the tasks that ASET performs. The default is to list all seven tasks:

```
TASKS="env sysconfig usrgrp tune cktlist eeprom firewall"
```

UID_ALIASES Environment 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, where pathname is the full path name of the aliases file.
The default is as follows:

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

**YPCHECK Environment Variable**

The YPCHECK variable extends the task of checking system tables to include NIS or NIS+ tables. The YPCHECK variable is a Boolean variable, which can be set to either true or false.

The default is false, which confines the checking to local system tables:

```
YPCHECK=false
```

**CKLISTPATH level Environment Variables**

The three checklist path variables list the directories to be checked by the system files checks task. The following definitions of the variables are set by default. The definitions 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/ucblib
```

The values for the checklist path environment variables are similar to the values of the shell path variables. Like the shell path variables, the checklist path environment variables are lists of directory names. The directory names are 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 ASET files, including the tune files and the aliases file.

**Tune File Examples**

ASET maintains three tune files. Each entry in a tune file occupies one line. The fields in an entry are in the following order:

```
pathname mode owner group type
```

- **pathname**: The full path name of the file
- **mode**: A five-digit number that represents the permission setting
- **owner**: The owner of the file
- **group**: The group owner of the file
- **type**: The type of file
The following rules apply when you edit the tune files:

- You can use regular shell wildcard characters, such as an asterisk (*) and a question mark (?), in the path name for multiple references. For more information, see sh(1).
- `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 remains unchanged, because 00777 is always less restrictive than whatever the current setting is.

This process is how ASET handles mode setting. The process is different if the security level is being downgraded, or if you are removing ASET. When you decrease the security level from the level in 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 `symbolic link`, `directory`, or `file`. A symbolic link is a symbolic link.
- Higher security level tune files reset file permissions to be at least as restrictive as file permissions at lower levels. Also, at higher security levels, additional files are added to the list.
- A file can match more than one tune file entry. For example, `/etc/passwd` matches the `/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 the `/etc/passwd` file is 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` designations or `group` designations, the last entry takes precedence. In the following example, the owner of `/usr/sbin/chroot` is set to `root`.

```
/usr/sbin/chroot 00555 bin bin file
/usr/sbin/chroot 00555 root bin file
```

### Aliases File Examples

The aliases file contains a list of aliases that share the same user ID.

Each entry is in this form:

```
uid=alias1=alias2=alias3=...
```

- `uid` Shared UID.
- `aliasn` User accounts that share a UID.
For example, the following entry lists the UID 0. The UID is being shared by the sysadm and root accounts:

0=root=sysadm

Running ASET (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run ASET from the command line</td>
<td>Protects the system at the ASET level that you specify. Views the execution log to see the changes.</td>
<td>“How to Run ASET Interactively” on page 160</td>
</tr>
<tr>
<td>Run ASET in batch mode at regular intervals</td>
<td>Sets up a cron job to ensure that ASET protects the system.</td>
<td>“How to Run ASET Periodically” on page 161</td>
</tr>
<tr>
<td>Stop running ASET in batch mode</td>
<td>Removes the ASET cron job.</td>
<td>“How to Stop Running ASET Periodically” on page 162</td>
</tr>
<tr>
<td>Store ASET reports on a server</td>
<td>Collects ASET reports from clients for monitoring in a central location.</td>
<td>“How to Collect ASET Reports on a Server” on page 162</td>
</tr>
</tbody>
</table>

To set the variables in ASET, see “ASET Environment Variables” on page 155. To configure ASET, see “Configuring ASET” on page 152.

How to Run ASET Interactively

1. **Become superuser or assume an equivalent role.**
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.

2. **Run ASET interactively by using the `aset` command.**
   ```
   # /usr/aset/aset -l level -d pathname
   ```
   - `level` Specifies the level of security. Valid values are low, medium, or high. The default setting is low. For detailed information about security levels, see “ASET Security Levels” on page 144.
   - `pathname` Specifies the working directory for ASET. The default is `/usr/aset`.

3. **Verify that 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 7–1 Running ASET Interactively

In the following example, ASET is run at low security with the default working directory.

```bash
# /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
  usgrp
  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 or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.
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 every day at midnight.

If you want to set up a different time, edit the PERIODIC_SCHEDULE variable in the /usr/aset/asetenv file. For detailed information about setting the PERIODIC_SCHEDULE variable, see "PERIODIC_SCHEDULE Environment Variable" on page 157.

3 Add an entry to the crontab file by using the set command.

```
# /usr/aset/aset -p
```

The -p option 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 is scheduled to run.

```
# crontab -l root
```

▼ How to Stop Running ASET Periodically

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

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 that the ASET entry is deleted.

```
# crontab -l root
```

▼ How to Collect ASET Reports on a Server

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.
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.
      
      This step creates a client_rpt subdirectory for a client. Repeat this step for each client whose reports you need to collect.
      
      mars# cd rptdir
      mars# mkdir client_rpt
      
      In the following example, the directory all_reports, and the subdirectories pluto_rpt and neptune_rpt are created.
      
      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 and write options.
   
   For example, the following entries in the dfstab file are shared with read and 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 that are listed in vfstab are automatically mounted.
   
   mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes hard
Troubleshooting ASET Problems

This section describes the error messages that are 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 if these values are incorrect, enter the correct values.

ASET working directory undefined.

To specify, set ASETDIR environment variable or use command line option -d.

ASET startup unsuccessful.

Cause: The ASET working directory is not defined, or the directory is defined incorrectly. The working directory is the operating directory.

Solution: Use the ASETDIR environment variable or the -d command-line option to correct the error, and restart ASET.

ASET working directory $ASETDIR missing.

ASET startup unsuccessful.

Cause: The ASET working directory is not defined, or the directory is defined incorrectly. The working directory is the operating directory. This problem might be because the ASETDIR variable refers to a nonexistent directory. Or the -d command-line option might refer to a nonexistent directory.

Solution: Ensure that the correct directory, that is, the directory that contains the ASET directory hierarchy, is referred to correctly.

Cannot expand $ASETDIR to full pathname.

Cause: ASET cannot expand the directory name that is given by the ASETDIR variable or the -d command-line option to a full path name.
**Solution:** Ensure that the directory name is correct. Ensure that the directory refers to an existing directory to which the user has access.

*aset: invalid/undefined security level.*

To specify, set `ASETSECELEVEL` environment variable or use command line option `-l`, with argument=`low/med/high`.

**Cause:** The security level is not defined, or the level is defined incorrectly. Only the values `low`, `med`, or `high` are acceptable.

**Solution:** Use the `ASETSECELEVEL` 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. For the details about this file, see the `asetenv(4)` man page.

*filename doesn’t exist or is not readable.*

**Cause:** The file that is referred to by `filename` either does not exist or is not readable. This problem can occur when you are using the `-u` option. The option permits you to specify a file that contains a list of users whom you want to check.

**Solution:** Ensure that the argument to the `-u` option exists and that the argument is readable.

*ASET task list TASKLIST undefined.*

**Cause:** The ASET task list, which should be defined in the `asetenv` file, is not defined. This message can mean that your `asetenv` file is bad.

**Solution:** Examine your `asetenv` file. Ensure that the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure that the file is intact. For the content of a valid `asetenv` file, see the `asetenv(4)` man page.

*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 message can mean that your `asetenv` file is bad.
Solution: Examine your asetenv file. Ensure that the task list is defined in the User Configurable section. Also check other parts of the file to ensure that the file is intact. For the content of a valid asetenv file, see the asetenv(4) man page.

Schedule undefined for periodic invocation.
No tasks executed or scheduled. Check asetenv file.

Cause: ASET scheduling is requested by using the -p option, but the environment variable PERIODIC_SCHEDULE is undefined in the asetenv file.

Solution: Check the User Configurable section of the asetenv file to ensure that the variable is defined. Ensure that the variable is in proper format.

Warning! Duplicate ASET execution scheduled.
Check crontab file.

Cause: ASET is scheduled to run more than once. In other words, ASET scheduling is requested while a schedule is already in effect. This message does not necessarily indicate an error if more than one schedule is indeed desired. In this instance, the messages servers only as a warning. If you want more than one schedule, you should use the proper scheduling format with the crontab command. For more information, see the crontab(1) man page.

Solution: Verify, through the crontab command, that the correct schedule is in effect. Ensure that no unnecessary crontab entries for ASET are in place.
This section covers role-based access control (RBAC) and process rights management. RBAC components include roles, rights profiles, and authorizations. Process rights management is implemented through privileges. Privileges work with RBAC to provide a more secure administration alternative than administration of a system by a superuser.
Solaris role-based access control (RBAC) and privileges provide a more secure alternative to superuser. This chapter provides overview information about RBAC and about privileges.

The following is a list of the overview information in this chapter.

- “Role-Based Access Control (Overview)” on page 169
- “Privileges (Overview)” on page 178

### Role-Based Access Control (Overview)

Role-based access control (RBAC) is a security feature for controlling user access to tasks that would normally be restricted to superuser. By applying security attributes to processes and to users, RBAC can divide up superuser capabilities among several administrators. Process rights management is implemented through privileges. User rights management is implemented through RBAC.

- For a discussion of process rights management, see “Privileges (Overview)” on page 178.
- For information on RBAC tasks, see Chapter 9.
- For reference information, see Chapter 10.

### RBAC: An Alternative to the Superuser Model

In conventional UNIX systems, the root user, also referred to as superuser, is all-powerful. Programs that run as root, or setuid programs, are all-powerful. The root user has 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 confidential records, and shut down the entire network. A setuid program that is hijacked can do anything on the system.

Role-based access control (RBAC) provides a more secure alternative to the all-or-nothing superuser model. With RBAC, you can enforce security policy at a more fine-grained level. RBAC uses the security principle of least privilege. Least privilege means that a user has precisely the amount of privilege that is necessary to perform a job. Ordinary users have enough privilege to use their
applications, check the status of their jobs, print files, create new files, and so on. Capabilities beyond ordinary user capabilities are grouped into rights profiles. Users who are expected to do jobs that require some of the capabilities of superuser assume a role that includes the appropriate rights profile.

RBAC collects superuser capabilities into rights profiles. These rights profiles are assigned to special user accounts that are called roles. A user can then assume a role to do a job that requires some of superuser’s capabilities. Predefined rights profiles are supplied with Solaris software. You create the roles and assign the profiles.

Rights profiles can provide broad capabilities. For example, the Primary Administrator rights profile is equivalent to superuser. Rights profiles can also be narrowly defined. For example, the Cron Management rights profile manages at and cron jobs. When you create roles, you can decide to create roles with broad capabilities, or roles with narrow capabilities, or both.

In the RBAC model, superuser creates one or more roles. The roles are based on rights profiles. Superuser then assigns the roles to users who are trusted to perform the tasks of the role. Users log in with their user name. After login, users assume roles that can run restricted administrative commands and graphical user interface (GUI) tools.

The flexibility in setting up roles enables a variety of security policies. Although no roles are shipped with the Solaris Operating System (Solaris OS), three recommended roles can easily be configured. The roles are based on rights profiles of the same name:

- **Primary Administrator** – A powerful role that is equivalent to the root user, or superuser.
- **System Administrator** – A less powerful role for administration that is not related to security. This role can manage file systems, mail, and software installation. However, this role cannot set passwords.
- **Operator** – A junior administrator role for operations such as backups and printer management.

These three roles do not have to be implemented. Roles are a function of an organization’s security needs. Roles can be set up for special-purpose administrators in areas such as security, networking, or firewall administration. Another strategy is to create a single powerful administrator role along with an advanced user role. The advanced user role would be for users who are permitted to fix portions of their own systems.

The superuser model and the RBAC model can co-exist. The following table summarizes the gradations from superuser to restricted ordinary user that are possible in the RBAC model. The table includes the administrative actions that can be tracked in both models. For a summary of the effect of privileges alone on a system, see Table 8–2.

<table>
<thead>
<tr>
<th>User Capabilities on a System</th>
<th>Superuser Model</th>
<th>RBAC Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can become superuser with full superuser capability</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
TABLE 8–1  Superuser Model Versus RBAC With Privileges Model (Continued)

<table>
<thead>
<tr>
<th>User Capabilities on a System</th>
<th>Superuser Model</th>
<th>RBAC Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can log in as a user with full user capabilities</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can become superuser with limited capabilities</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can log in as a user, and have superuser capabilities, sporadically</td>
<td>Yes, with setuid programs only</td>
<td>Yes, with setuid programs and with RBAC</td>
</tr>
<tr>
<td>Can log in as a user with administrative capabilities, but without full superuser capability</td>
<td>No</td>
<td>Yes, with RBAC and with directly-assigned privileges and authorizations</td>
</tr>
<tr>
<td>Can log in as a user with fewer capabilities than an ordinary user</td>
<td>No</td>
<td>Yes, with RBAC and with removed privileges</td>
</tr>
<tr>
<td>Can track superuser actions</td>
<td>Yes, by auditing the su command</td>
<td>Yes, by auditing profile shell commands</td>
</tr>
</tbody>
</table>

Solaris RBAC Elements and Basic Concepts

The RBAC model in the Solaris OS introduces the following elements:

- **Authorization** – A permission that enables a user or role to perform a class of actions that could affect security. For example, security policy at installation gives ordinary users the `solaris.device.cdrw` authorization. This authorization enables users to read and write to a CD-ROM device. For a list of authorizations, see the `/etc/security/auth_attr` file.

- **Privilege** – A discrete right that can be granted to a command, a user, a role, or a system. Privileges enable a process to succeed. For example, the `proc_exec` privilege allows a process to call `execve()`). Ordinary users have basic privileges. To see your basic privileges, run the `ppriv -vl` basic command.

- **Security attributes** – An attribute that enables a process to perform an operation. In a typical UNIX environment, a security attribute enables a process to perform an operation that is otherwise forbidden to ordinary users. For example, `setuid` and `setgid` programs have security attributes. In the RBAC model, operations that ordinary users perform might require security attributes. In addition to `setuid` and `setgid` programs, authorizations and privileges are also security attributes in the RBAC model. For example, a user with the `solaris.device.allocate` authorization can allocate a device for exclusive use. A process with the `sys_time` privilege can manipulate system time.
Privileged application – An application or command that can override system controls by checking for security attributes. In a typical UNIX environment and in the RBAC model, programs that use setuid and setgid are privileged applications. In the RBAC model, programs that require privileges or authorizations to succeed are also privileged applications. For more information, see “Privileged Applications and RBAC” on page 174.

Rights profile – A collection of administrative capabilities that can be assigned to a role or to a user. A rights profile can consist of authorizations, of commands with security attributes, and of other rights profiles. Rights profiles offer a convenient way to group security attributes.

Role – A special identity for running privileged applications. The special identity can be assumed by assigned users only. In a system that is run by roles, superuser is unnecessary. Superuser capabilities are distributed to different roles. For example, in a two-role system, security tasks would be handled by a security role. The second role would handle system administration tasks that are not security-related. Roles can be more fine-grained. For example, a system could include separate administrative roles for handling the cryptographic framework, printers, system time, file systems, and auditing.

The following figure shows how the RBAC elements work together.

In RBAC, roles are assigned to users. When a user assumes a role, the capabilities of the role are available. Roles get their capabilities from rights profiles. Rights profiles can contain authorizations, privileged commands, and other supplementary rights profiles. Privileged commands are commands that execute with security attributes.

The following figure uses the Operator role, the Operator rights profile, and the Printer Management rights profile to demonstrate RBAC relationships.
The Operator role is used to maintain printers and to perform media backup. The role is assigned to the user jdoe. jdoe can assume the role by switching to the role, and then supplying the role password.

The Operator rights profile has been assigned to the Operator role. The Operator rights profile contains two supplementary profiles, Printer Management and Media Backup. The supplementary profiles reflect the role’s primary tasks.

The Printer Management rights profile is for managing printers, print daemons, and spoolers. Three authorizations are included in the Printer Management rights profile:
solaris.admin.printer.read, solaris.admin.printer.modify, and solaris.admin.printer.delete. These authorizations enable roles and users to manipulate...
information in the printer queue. The Printer Management rights profile also includes a number of commands with security attributes, such as `/usr/sbin/lpshut` with `euid=lp` and `/usr/ucb/lpq` with `euid=0`.

**RBAC Authorizations**

An **authorization** is a discrete right that can be granted to a role or to a user. Authorizations enforce policy at the user application level. Authorizations can be assigned directly to a role or to a user. Typically, authorizations are included in a rights profile. The rights profile is then included in a role, and the role is assigned to a user. For an example, see Figure 8–2.

RBAC-compliant applications can check a user’s authorizations prior to granting access to the application or specific operations within the application. This check replaces the check in conventional UNIX applications for `UID=0`. For more information on authorizations, see the following sections:

- “Authorization Naming and Delegation” on page 218
- “`auth_attr Database`” on page 221
- “Commands That Require Authorizations” on page 226

**Authorizations and Privileges**

Privileges enforce security policy in the kernel. The difference between authorizations and privileges concerns the level at which the security policy is enforced. Without the proper privilege, a process can be prevented from performing privileged operations by the kernel. Without the proper authorizations, a user might be prevented from using a privileged application or from performing security-sensitive operations within a privileged application. For a fuller discussion of privileges, see “Privileges (Overview)” on page 178.

**Privileged Applications and RBAC**

Applications and commands that can override system controls are considered privileged applications. Security attributes such as `UID=0`, privileges, and authorizations make an application privileged.

**Applications That Check UIDs and GIDs**

Privileged applications that check for root (`UID=0`) or some other special UID or GID have long existed in the UNIX environment. The rights profile mechanism enables you to isolate commands that require a specific ID. Instead of changing the ID on a command that anyone can access, you can place the command with execution security attributes in a rights profile. A user or role with that rights profile can then run the program without having to become superuser.
IDs can be specified as real or effective. Assigning effective IDs is preferred over assigning real IDs. Effective IDs are equivalent to the setuid feature in the file permission bits. Effective IDs also identify the UID for auditing. However, because some shell scripts and programs require a real UID of root, real UIDs can be set as well. For example, the pkgadd command requires a real rather than an effective UID. If an effective ID is not sufficient to run a command, you need to change the ID to a real ID. For the procedure, see “How to Create or Change a Rights Profile” on page 206.

**Applications That Check for Privileges**

Privileged applications can check for the use of privileges. The RBAC rights profile mechanism enables you to specify the privileges for specific commands. Instead of requiring superuser capabilities to use an application or command, you can isolate the command with execution security attributes in a rights profile. A user or role with that rights profile can then run the command with just the privileges that the command requires to succeed.

Commands that check for privileges include the following:

- Kerberos commands, such as kadmin, kprop, and kdb5_util
- Network commands, such as ifconfig, routeadm, and snoop
- File and file system commands, such as chmod, chgrp, and mount
- Commands that control processes, such as kill, pcre, and rcapadm

To add commands with privileges to a rights profile, see “How to Create or Change a Rights Profile” on page 206. To determine what commands check for privileges in a particular profile, see "Determining Your Assigned Privileges" on page 237.

**Applications That Check Authorizations**

The Solaris OS additionally provides commands that check authorizations. By definition, the root user has all authorizations. Therefore, the root user can run any application. Applications that check for authorizations include the following:

- The entire Solaris Management Console suite of tools
- Audit administration commands, such as auditconfig and auditreduce
- Printer administration commands, such as lpdadm and lpfilter
- The batch job-related commands, such as at, atq, batch, and crontab
- Device-oriented commands, such as allocate, deallocate, list_devices, and cdrw.

To test a script or program for authorizations, see Example 9–19. To write a program that requires authorizations, see "About Authorizations" in Solaris Security for Developers Guide.

**RBAC Rights Profiles**

A rights profile is a collection of system overrides that can be assigned to a role or user. A rights profile can include authorizations, commands with assigned security attributes, and other rights profiles. Rights profile information is split between the prof_attr and exec_attr databases. The rights
profile names and authorizations are in the prof_attr database. The rights profile name and the commands with assigned security attributes are in the exec_attr database. For more information on rights profiles, see the following sections:

- “Contents of Rights Profiles” on page 213
- “prof_attr Database” on page 222
- “exec_attr Database” on page 223

**RBAC 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. Roles have a home directory, a group assignment, a password, and so on. Rights profiles and authorizations give the role administrative capabilities. Roles cannot inherit capabilities from other roles or other users. Discrete roles parcel out superuser capabilities, and thus enable more secure administrative practices.

When a user assumes a role, the role’s attributes replace all user attributes. Role information is stored in the passwd, shadow, and user_attr databases. Role information can be added to the audit_user database. For detailed information on setting up roles, see the following sections:

- “How to Plan Your RBAC Implementation” on page 189
- “How to Create a Role From the Command Line” on page 193
- “How to Change the Properties of a Role” on page 204

A role can be assigned to more than one user. 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 the su command and supplying the role name and password. Users can also assume a role in the Solaris Management Console tool.

A role cannot log in directly. A user logs in, and then assumes a role. Having assumed a role, the user cannot assume another role without first exiting their current role. Having exited the role, the user can then assume another role.

You can prevent anonymous root login by changing the root user into a role, as shown in “How to Make root User Into a Role” on page 198. If the profile shell command, pfexec, is being audited, the audit trail contains the login user’s real UID, the roles that the user has assumed, and the actions that the role performed. To audit the system or a particular user for role operations, see "How to Audit Roles” on page 197.

No predefined roles are shipped with Solaris software.

- To configure the Primary Administrator role, see “Using the Solaris Management Tools With RBAC (Task Map)” in *System Administration Guide: Basic Administration*.
- To configure other roles, see “How to Create and Assign a Role By Using the GUI” on page 191.
- To create roles on the command line, see “Managing RBAC (Task Map)” on page 203.
Profile Shell in RBAC

Roles can run privileged applications from the Solaris Management Console launcher or from a profile shell. A profile shell is a special shell that recognizes the security attributes that are included in a rights profile. Profile shells are launched when the user runs the su command to assume a role. The profile shells are pfsh, pfcsh, and pfksh. The shells correspond to Bourne shell (sh), C shell (csh), and Korn shell (ksh), respectively.

Users who have been directly assigned a rights profile must invoke a profile shell to run the commands with security attributes. For usability and security considerations, see “Security Considerations When Directly Assigning Security Attributes” on page 177.

All commands that are executed in a profile shell can be audited. For more information, see “How to Audit Roles” on page 197.

Name Service Scope and RBAC

Name service scope is an important concept for understanding RBAC. The scope of a role might be limited to an individual host. Alternatively, the scope might include all hosts that are served by a name service such as NIS, NIS+, or LDAP. The name service scope for a system is specified in the file /etc/nsswitch.conf. A lookup stops at the first match. For example, if a rights profile exists in two name service scopes, only the entries in the first name service scope are used. If files is the first match, then the scope of the role is limited to the local host.

Security Considerations When Directly Assigning Security Attributes

Typically, a user obtains administrative capabilities through a role. Authorizations and privileged commands are grouped into a rights profile. The rights profile is included in a role, and the role is assigned to a user.

Direct assignment of rights profiles and security attributes is also possible:

- Rights profiles, privileges, and authorizations can be assigned directly to users.
- Privileges and authorizations can be assigned directly to roles.

However, direct assignment is not a secure practice. Users and roles with a directly assigned privilege could override security policy wherever this privilege is required by the kernel. When a privilege is a security attribute of a command in a rights profile, that privilege is available only for that command by someone who has that rights profile. The privilege is not available for other commands that the user or role might run.

Since authorizations act at the user level, direct assignment of authorizations can be less dangerous than direct assignment of privileges. However, authorizations can enable a user to perform highly secure tasks, such as delegate device administration.
A rights profile that is assigned directly to a user presents usability problems more than security problems. The commands with security attributes in the rights profile can only succeed in a profile shell. The user must open a profile shell, then type the commands. A role that is assigned a rights profile gets a profile shell automatically. Therefore, the commands succeed in the role’s shell.

Rights profiles provide an extensible, clean way to group security characteristics for particular administrative tasks.

**Privileges (Overview)**

Process rights management enables processes to be restricted at the command, user, role, or system level. The Solaris OS implements process rights management through privileges. Privileges decrease the security risk that is associated with one user or one process having full superuser capabilities on a system. Privileges and RBAC provide a compelling alternative model to the traditional superuser model.

- For information on RBAC, see "Role-Based Access Control (Overview)" on page 169.
- For information on how to administer privileges, see Chapter 11.
- For reference information on privileges, see Chapter 12.

**Privileges Protect Kernel Processes**

A privilege is a discrete right that a process requires to perform an operation. The right is enforced in the kernel. A program that operates within the bounds of the Solaris basic set of privileges operates within the bounds of the system security policy. setuid programs are examples of programs that operate outside the bounds of the system security policy. By using privileges, programs eliminate the need for calls to setuid.

Privileges discretely enumerate the kinds of operations that are possible on a system. Programs can be run with the exact privileges that enable the program to succeed. For example, a program that sets the date and writes the date to an administrative file might require the file_dac_write and sys_time privileges. This capability eliminates the need to run any program as root.

Historically, systems have not followed the privilege model. Rather, systems used the superuser model. In the superuser model, processes run as root or as a user. User processes were limited to acting on the user’s directories and files. root processes could create directories and files anywhere on the system. A process that required creation of a directory outside the user’s directory would run with a UID=0, that is, as root. Security policy relied on DAC, discretionary access control, to protect system files. Device nodes were protected by DAC. For example, devices owned by group sys could be opened only by members of group sys.

However, setuid programs, file permissions, and administrative accounts are vulnerable to misuse. The actions that a setuid process is permitted are more numerous than the process requires to complete its operation. A setuid program can be compromised by an intruder who then runs as the all-powerful root user. Similarly, any user with access to the root password can compromise the entire system.
In contrast, a system that enforces policy with privileges allows a gradation between user capabilities and root capabilities. A user can be granted privileges to perform activities that are beyond the capabilities of ordinary users, and root can be limited to fewer privileges than root currently possesses. With RBAC, a command that runs with privileges can be isolated in a rights profile and assigned to one user or role. Table 8–1 summarizes the gradation between user capabilities and root capabilities that the RBAC plus privileges model provides.

The privilege model provides greater security than the superuser model. Privileges that have been removed from a process cannot be exploited. Process privileges prevent a program or administrative account from gaining access to all capabilities. Process privileges can provide an additional safeguard for sensitive files, where DAC protections alone can be exploited to gain access.

Privileges, then, can restrict programs and processes to just the capabilities that the program requires. This capability is called the principle of least privilege. On a system that implements least privilege, an intruder who captures a process has access to only those privileges that the process has. The rest of the system cannot be compromised.

Privilege Descriptions

Privileges are logically grouped on the basis of the area of the privilege.

- **FILE privileges** – Privileges that begin with the string `file` operate on file system objects. For example, the `file_dac_write` privilege overrides discretionary access control when writing to files.

- **IPC privileges** – Privileges that begin with the string `ipc` override IPC object access controls. For example, the `ipc_dac_read` privilege enables a process to read remote shared memory that is protected by DAC.

- **NET privileges** – Privileges that begin with the string `net` give access to specific network functionality. For example, the `net_rawaccess` privilege enables a device to connect to the network.

- **PROC privileges** – Privileges that begin with the string `proc` allow processes to modify restricted properties of the process itself. PROC privileges include privileges that have a very limited effect. For example, the `proc_clock_highres` privilege enables a process to use high resolution timers.

- **SYS privileges** – Privileges that begin with the string `sys` give processes unrestricted access to various system properties. For example, the `sys_linkdir` privilege enables a process to make and break hard links to directories.

Some privileges have a limited effect on the system, and some have a broad effect. The definition of the `proc_taskid` privilege indicates its limited effect:

```
proc_taskid
   Allows a process to assign a new task ID to the calling process.
```

The definition of the `file_setid` privilege indicates its broad effect:

```
net_rawaccess
   Allow a process to have direct access to the network layer.
```
The `privileges(5)` man page provides descriptions of every privilege. The command `ppriv -lv` prints a description of every privilege to standard out.

**Administrative Differences on a System With Privileges**

A system that has privileges has several visible differences from a system that does not have privileges. The following table lists some of the differences.

<table>
<thead>
<tr>
<th>Feature</th>
<th>No Privileges</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daemons</td>
<td>Daemons run as root.</td>
<td>Daemons run as the user daemon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, the following daemons have been assigned appropriate privileges and run as daemon: lockd, mountd, nfud, and rpcbind.</td>
</tr>
<tr>
<td>Log File</td>
<td>Log files are owned by root.</td>
<td>Log files are now owned by daemon, who created the log file. The root user does not own the file.</td>
</tr>
<tr>
<td>Ownership</td>
<td></td>
<td>Log files are now owned by daemon, who created the log file. The root user does not own the file.</td>
</tr>
<tr>
<td>Error Messages</td>
<td>Error messages refer to superuser.</td>
<td>Error messages reflect the use of privileges.</td>
</tr>
<tr>
<td></td>
<td>For example, <code>chroot: not superuser</code>.</td>
<td>For example, the equivalent error message for <code>chroot</code> failure is <code>chroot: exec failed</code>.</td>
</tr>
<tr>
<td>setuid Programs</td>
<td>Programs use <code>setuid</code> to complete tasks that ordinary users are not allowed to perform.</td>
<td>Many <code>setuid</code> programs have been changed to run with privileges.</td>
</tr>
<tr>
<td>File Permissions</td>
<td>Device permissions are controlled by DAC. For example, members of the group <code>sys</code> can open <code>/dev/ip</code>.</td>
<td>File permissions (DAC) do not predict who can open a device. Devices are protected with DAC and device policy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, the <code>/dev/ip</code> file has 666 permissions, but the device can only be opened by a process with the appropriate privileges. Raw sockets are still protected by DAC.</td>
</tr>
<tr>
<td>Audit Events</td>
<td>Auditing the use of the <code>su</code> command covers many administrative functions.</td>
<td>Auditing the use of privileges covers most administrative functions. The <code>pm</code> and <code>as</code> audit classes include audit events that configure device policy and audit events that set privileges.</td>
</tr>
<tr>
<td>Processes</td>
<td>Processes are protected by who owns the process.</td>
<td>Processes are protected by privileges. Process privileges and process flags are visible as a new entry in the <code>/proc/&lt;pid&gt;</code> directory, <code>priv</code>.</td>
</tr>
</tbody>
</table>
### How Privileges Are Implemented

Every process has four sets of privileges that determine whether a process can use a particular privilege. The kernel automatically calculates the effective set of privileges. You can modify the initial inheritable set of privileges. A program that is coded to use privileges can reduce the program’s permitted set of privileges. You can shrink the limit set of privileges.

- **Effective privilege set, or E** – Is the set of privileges that is currently in effect. A process can add privileges that are in the permitted set to the effective set. A process can also remove privileges from E.

- **Permitted privilege set, or P** – Is the set of privileges that is available for use. Privileges can be available to a program from inheritance or through assignment. An execution profile is one way to assign privileges to a program. The `setuid` command assigns all privileges that root has to a program. Privileges can be removed from the permitted set, but privileges cannot be added to the set. Privileges that are removed from P are automatically removed from E.

A privilege-aware program removes the privileges that a program never uses from the program’s permitted set. In this way, unnecessary privileges cannot be exploited by the program or a malicious process. For more information on privilege-aware programs, see Chapter 2, “Developing Privileged Applications,” in Solaris Security for Developers Guide.

- **Inheritable privilege set, or I** – Is the set of privileges that a process can inherit across a call to `exec`. After the call to `exec`, the permitted and the effective sets are equal, except in the special case of a `setuid` program.

For a `setuid` program, after the call to `exec`, the inheritable set is first restricted by the limit set. Then, the set of privileges that were inherited (I), minus any privileges that were in the limit set (L), are assigned to P and E for that process.

- **Limit privilege set, or L** – Is the outside limit of what privileges are available to a process and its children. By default, the limit set is all privileges. Processes can shrink the limit set but can never extend the limit set. L is used to restrict I. Consequently, L restricts P and E at the time of `exec`.

If a user has been assigned a profile that includes a program that has been assigned privileges, the user can usually run that program. On an unmodified system, the program’s assigned privileges are within the user’s limit set. The privileges that have been assigned to the program become part of the user’s permitted set. To run the program that has been assigned privileges, the user must run the program from a profile shell.

---

**TABLE 8–2** Visible Differences Between a System With Privileges and a System Without Privileges

<table>
<thead>
<tr>
<th>Feature</th>
<th>No Privileges</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugging</td>
<td>No reference to privileges in core dumps.</td>
<td>The ELF note section of core dumps includes information about process privileges and flags in the <code>NT_PRPRIV</code> and <code>NT_PRPRIVINFO</code> notes. The <code>ppriv</code> utility and other utilities show the proper number of properly sized sets. The utilities correctly map the bits in the bit sets to privilege names.</td>
</tr>
</tbody>
</table>
The kernel recognizes a basic privilege set. On an unmodified system, each user’s initial inheritable set equals the basic set at login. You can modify the user’s initial inheritable set. You cannot modify the basic set.

On an unmodified system, a user’s privilege sets at login would appear similar to the following:

- **E (Effective):** basic
- **I (Inheritable):** basic
- **P (Permitted):** basic
- **L (Limit):** all

Therefore, at login, all users have the basic set in their inheritable set, their permitted set, and their effective set. A user’s limit set contains all privileges. To put more privileges in the user’s effective set, you must assign a rights profile to the user. The rights profile would include commands to which you have added privileges. You can also assign privileges directly to the user or role, though such privilege assignment can be risky. For a discussion of the risks, see “Security Considerations When Directly Assigning Security Attributes” on page 177.

**How Processes Get Privileges**

Processes can inherit privileges. Or, processes can be assigned privileges. A process inherits privileges from its parent process. At login, the user’s initial inheritable set of privileges determines what privileges are available to the user’s processes. All child processes of the user’s initial login inherit that set.

You can also directly assign privileges to programs, users, and roles. When a program requires privileges, you assign the privileges to the program’s executable in a rights profile. Users or roles that are permitted to run the program are assigned the profile that includes the program. At login or when a profile shell is entered, the program runs with privilege when the program’s executable is typed in the profile shell. For example, a role that includes the Object Access Management profile is able to run the chmod command with the file_chown privilege.

When a role or user runs a program that has been directly assigned an additional privilege, the assigned privilege is added to the role or user’s inheritable set. Child processes of the program that was assigned privilege inherit the privileges of the parent. If the child process requires more privileges than the parent process, the child process must be directly assigned those privileges.

Programs that are coded to use privileges are called privilege-aware programs. A privilege-aware program turns on the use of privilege and turns off the use of privilege during program execution. To succeed in a production environment, the program must be assigned the privileges that the program turns on and off.

For examples of privilege-aware code, see Chapter 2, “Developing Privileged Applications,” in Solaris Security for Developers Guide. To assign privileges to a program that requires privileges, see “How to Add Privileges to a Command” on page 233.
Assigning Privileges

You, in your capacity as system administrator, are responsible for assigning privileges. Typically, you assign the privilege to a command in a rights profile. The rights profile is then assigned to a role or to a user. The Solaris Management Console provides the graphical user interface (GUI) to assign privileges. Privileges can also be assigned by using commands such as smuser and smrole. For more information on how to use the GUI to assign privileges, see Chapter 9.

Privileges can also be assigned directly to a user. If you trust a subset of users to use a privilege responsibly throughout their sessions, you can assign the privilege directly. Good candidates for direct assignment are privileges that have a limited effect, such as proc_clock_highres. Poor candidates for direct assignment are privileges that have far-reaching effects, such as file_dac_write.

Privileges can also be denied to a user or to a system. Care must be taken when removing privileges from the initial inheritable set or the limit set of a user or a system.

Expanding a User or Role’s Privileges

Users and roles have an inheritable set of privileges, and a limit set of privileges. The limit set cannot be expanded, since the limit set is initially all privileges. The initial inheritable set can be expanded for users, roles, and systems. A privilege that is not in the inheritable set can also be assigned to a process.

The assignment of privileges per process is the most precise way to add privileges. You can expand the number of privileged operations that a user can perform by enabling the user to assume a role. The role would be assigned profiles that include commands with added privileges. When the user assumes the role, the user gets the role’s profile shell. By typing in the role’s shell, the commands in the role’s profiles execute with the added privileges.

You can also assign a profile to the user rather than to a role that the user assumes. The profile would include commands with added privileges. When the user opens a profile shell, such as pfksh, the user can execute the commands in the user’s profile with privilege. In a regular shell, the commands do not execute with privilege. The privileged process can only execute in a privileged shell.

To expand the initial inheritable set of privileges for users, roles, or systems is a riskier way to assign privileges. All privileges in the inheritable set are in the permitted and effective sets. All commands that the user or role types in a shell can use the directly assigned privileges. Directly assigned privileges enable a user or role to easily perform operations that can be outside the bounds of their administrative responsibilities.

When you add to the initial inheritable set of privileges on a system, all users who log on to the system have a larger set of basic privileges. Such direct assignment enables all users of the system to easily perform operations that are probably outside the bounds of ordinary users.
Restricting a User or Role’s Privileges

By removing privileges, you can prevent users and roles from performing particular tasks. You can remove privileges from the initial inheritable set, and from the limit set. You should carefully test removal of privileges before you distribute an initial inheritable set or a limit set that is smaller than the default set. By removing privileges from the initial inheritable set, you might prevent users from logging in. When privileges are removed from the limit set, a legacy setuid program might fail because the program requires a privilege that was removed.

Assigning Privileges to a Script

Scripts are executables, like commands. Therefore, in a rights profile, you can add privileges to a script just as you can add privileges to a command. The script runs with the added privileges when a user or role who has been assigned the profile executes the script in a profile shell. If the script contains commands that require privileges, the commands with added privileges should also be in the profile.

Privilege-aware programs can restrict privileges per process. Your job with a privilege-aware program is to assign the executable just the privileges that the program needs. You then test the program to see that the program succeeds in performing its tasks. You also check that the program does not abuse its use of privileges.

Privileges and Devices

The privilege model uses privileges to protect system interfaces that are protected by file permissions alone in the superuser model. In a system with privileges, file permissions are too weak to protect the interfaces. A privilege such as proc_owner could override file permissions and then give full access to all of the system.

Therefore, ownership of the device directory is not sufficient to open a device. For example, members of the group sys are no longer automatically allowed to open the /dev/1p device. The file permissions on /dev/1p are 0666, but the net_rawaccess privilege is required to open the device.

Device policy is controlled by privileges. The getdevpolicy command displays the device policy for every device. The device configuration command, devfsadm, installs the device policy. The devfsadm command binds privilege sets with open for reading or writing of devices. For more information, see the getdevpolicy(1M) and devfsadm(1M) man pages.

Device policy allows you more flexibility in granting permission to open devices. You can require different privileges or more privileges than the default device policy. The privilege requirements can be modified for the device policy and for the driver proper. You can modify the privileges when installing, adding, or updating a device driver.

The add_drv and update_drv commands can modify device policy entries and driver-specific privileges. You must be running a process with the full set of privileges to change the device policy. For more information, see the add_drv(1M) and update_drv(1M) man pages.
Privileges and Debugging

The Solaris OS provides tools to debug privilege failure. The `ppriv` command and the `truss` command provide debugging output. For examples, see the `ppriv(1)` man page. For a procedure, see "How to Determine Which Privileges a Program Requires" on page 232.
Using Role-Based Access Control (Tasks)

This chapter covers tasks for distributing the capabilities of superuser by using discrete roles. The mechanisms that roles can use include rights profiles, authorizations, and privileges. The following is a list of the task maps in this chapter.

- “Using RBAC (Task Map)” on page 187
- “Configuring RBAC (Task Map)” on page 188
- “Using Roles (Task Map)” on page 200
- “Managing RBAC (Task Map)” on page 203

For an overview of RBAC, see “Role-Based Access Control (Overview)” on page 169. For reference information, see Chapter 10. To use privileges with RBAC or without RBAC, see Chapter 11.

Using RBAC (Task Map)

To use RBAC requires planning, configuring RBAC, and knowing how to assume a role. Once roles become familiar, you might further customize RBAC to handle new operations. The following task map points to these major tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan and configure RBAC</td>
<td>Configure RBAC at your site.</td>
<td>“Configuring RBAC (Task Map)” on page 188</td>
</tr>
<tr>
<td>Use roles</td>
<td>Assume roles from the command line and in the Solaris Management Console GUI.</td>
<td>“Using Roles (Task Map)” on page 200</td>
</tr>
<tr>
<td>Customize RBAC</td>
<td>Customize RBAC for your site.</td>
<td>“Managing RBAC (Task Map)” on page 203</td>
</tr>
</tbody>
</table>
## Configuring RBAC (Task Map)

To use RBAC effectively requires planning. Use the following task map to plan and initially implement RBAC at your site.

<table>
<thead>
<tr>
<th>Task Description For Instructions</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan for RBAC Involves examining your site’s security needs, and deciding how to use RBAC at your site.</td>
<td>&quot;How to Plan Your RBAC Implementation&quot; on page 189</td>
</tr>
<tr>
<td>Learn to use the Solaris Management Console Involves becoming familiar with the Solaris Management Console.</td>
<td>Chapter 2, &quot;Working With the Solaris Management Console (Tasks),&quot; in System Administration Guide: Basic Administration</td>
</tr>
<tr>
<td>Configure the first user and role Uses the RBAC configuration tools in the Solaris Management Console to create a user and a role, and to assign the role to the user.</td>
<td>&quot;Using the Solaris Management Tools With RBAC (Task Map)&quot; in System Administration Guide: Basic Administration</td>
</tr>
<tr>
<td>(Optional) Create other users who can assume roles Ensures that users who can assume an administrative role exist.</td>
<td>&quot;Using the Solaris Management Tools With RBAC (Task Map)&quot; in System Administration Guide: Basic Administration</td>
</tr>
<tr>
<td>(Recommended) Create other roles and assign them to users Uses the RBAC tools to create roles for particular administrative areas, and to assign the roles to users.</td>
<td>&quot;How to Create and Assign a Role By Using the GUI&quot; on page 191</td>
</tr>
<tr>
<td>Example 9–5 Uses the command line to create roles, and to assign the roles to users</td>
<td>&quot;How to Create a Role From the Command Line&quot; on page 193</td>
</tr>
<tr>
<td>(Recommended) Audit role actions</td>
<td>&quot;How to Audit Roles&quot; on page 197</td>
</tr>
<tr>
<td>(Optional) Make root user a role Prevents anonymous root login, which is a security hole.</td>
<td>&quot;How to Make root User Into a Role&quot; on page 198</td>
</tr>
</tbody>
</table>

### Configuring RBAC

RBAC can be configured with the following utilities:

- **Solaris Management Console GUI** – The preferred method for performing RBAC-related tasks is through the GUI. The console tools for managing the RBAC elements are contained in the Users Tool collection.

- **Solaris Management Console commands** – With the Solaris Management Console command-line interfaces, such as `smrole`, you can operate on any name service. The Solaris Management Console commands require authentication to connect to the server. As a result, these commands are not practical for use in scripts.
Local commands – With the user* and role* set of command-line interfaces, such as useradd, you can operate on local files only. The commands that operate on local files must be run by superuser or by a role with the appropriate privileges.

How to Plan Your RBAC Implementation

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 your organization.

1 Learn the basic RBAC concepts.
   Read “Role-Based Access Control (Overview)” on page 169. Using RBAC to administer a system is very different from using conventional UNIX administrative practices. You should be familiar with the RBAC concepts before you start your implementation. For greater detail, see Chapter 10.

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 the security-relevant tasks through RBAC can be a part of the strategy. Although you can install the recommended roles and their configurations as is, you might need to customize your RBAC configuration to adhere to 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 configuration, you log in as yourself. Then, you type root as the user when you select a Solaris Management Console tool.
   - **Single Role Only** – This method adds one role. The one role is assigned the Primary Administrator rights profile. This method is similar to the superuser model, in that the role has superuser capabilities. However, this method enables you to track the user who has assumed the role.
   - **Recommended Roles** – This method creates three roles that are based on the following rights profiles: Primary Administrator, System Administrator, and Operator. The roles are suitable for organizations with administrators at different levels of responsibility.
   - **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.
   - **Root User as a Role** – This method prevents any user from logging in as root. Instead, users must log in as ordinary users prior to assuming the root role. For details, see “How to Make root User Into a Role” on page 198.
Decide which recommended roles are appropriate for your organization.

Review the capabilities of the recommended roles and default rights profiles. Default rights profiles enable administrators to configure a recommended role by using a single profile. Three default rights profiles are available for configuring the recommended roles:

- **Primary Administrator rights profile** – For configuring a role that can perform all administrative tasks, can grant rights to others, and can edit rights that are associated with administrative roles. A user in this role can assign this role to other users, and can grant rights to other users.

- **System Administrator rights profile** – For configuring a role that can perform most administrative tasks that are not related to security. For example, the System Administrator can add new user accounts, but cannot set passwords or grant rights to other users.

- **Operator rights profile** – For configuring a role that can perform simple administrative tasks, such as media backup and printer maintenance.

To further examine rights profiles, read one of the following:

- In the `/etc/security` directory, read the contents of the `prof_attr` database and the `exec_attr` database.
- In the Solaris Management Console, use the Rights tool to display the contents of a rights profile.
- In this book, refer to “Contents of Rights Profiles” on page 213 for summaries of some typical rights profiles.

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. You can customize roles and rights profiles to handle the security requirements of your organization.

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.

Decide which users should be assigned to the available roles.

According to the principle of least privilege, you should assign users to roles that are appropriate to their level of trust. When you prevent users from access to tasks that the users do not need to perform, you reduce potential problems.
How to Create and Assign a Role By Using the GUI

To create a new role, you can be superuser, or you can use the Primary Administrator role. In this procedure, the creator of the new role has assumed the role of Primary Administrator.

Before You Begin
- You have already created users who can assume a role at your site. If the users are not yet created, create them by following the instructions in “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.
- You have been assigned the Primary Administrator role by following the procedures in “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.

1 Start the Solaris Management Console.
   
   # /usr/sbin/smc &
   
   For login instructions, see “How to Assume a Role in the Solaris Management Console” on page 202.

2 Click the Administrative Roles icon.

3 Select Add Administrative Role from the Action menu.

4 Create a new role by filling in the fields in the series of dialog boxes.
   
   For possible roles, see Example 9–1 to Example 9–4.

   Tip – All tools in the Solaris Management Console display information in the bottom section of the page or at the left side of a wizard panel. Choose Help at any time to find additional information about performing tasks in this interface.

5 Assign the role to a user.

   Tip – After filling in the properties of the role, the last dialog box prompts you for a user for the role.

6 In a terminal window, restart the name service cache daemon.
   
   # svcadm restart system/name-service-cache
   
   For more information, see the svcadm(1M) and nscd(1M) man pages.

Example 9–1 Creating a Role for the System Administrator Rights Profile

In this example, the new role can do system administration tasks that are not connected to security. The role is created by performing the preceding procedure with the following parameters:

- Role name: sysadmin
- Role full name: System Administrator
Role description: Performs non-security admin tasks
Rights profile: System Administrator

This rights profile is at the top of the list of profiles that are included in the role.

Example 9–2 Creating a Role for the Operator Rights Profile

The Operator rights profile can manage printers and back up the system to offline media. You might want to assign the role to one user on each shift. To do so, you would select the role mailing list option in the Step 1: Enter a Role Name dialog box. The role is created by performing the preceding procedure with the following parameters:

- Role name: operadm
- Role full name: Operator
- Role description: Backup operator
- Rights profile: Operator

This rights profile must be at the top of the list of profiles that are included in the role.

Example 9–3 Creating a Role for a Security-Related Rights Profile

By default, the only rights profile that contains security-related commands and rights is the Primary Administrator profile. If you want to create a role that is not as powerful as Primary Administrator, but can handle some security-related tasks, you must create the role.

In the following example, the role protects devices. The role is created by performing the preceding procedure with the following parameters:

- Role name: devicesec
- Role full name: Device Security
- Role description: Configures Devices
- Rights profile: Device Security

In the following example, the role secures systems and hosts on the network. The role is created by performing the preceding procedure with the following parameters:

- Role name: netsec
- Role full name: Network Security
- Role description: Handles IPSec, IKE, and SSH
- Rights profile: Network Security

Example 9–4 Creating a Role for a Rights Profile With Limited Scope

A number of rights profiles are of limited scope. In this example, the sole task of the role is to manage DHCP. The role is created by performing the preceding procedure with the following parameters:

- Role name: dhcpmgmt
- Role full name: DHCP Management
Role description: Manages Dynamic Host Config Protocol
Rights profile: DHCP Management

Example 9–5  Modifying a User’s Role Assignment

In this example, a role is added to an existing user. The user’s role assignment is modified by clicking the User Accounts icon in the Users tool in the Solaris Management Console, double-clicking the user, and following the online help to add a role to the user’s capabilities.

Troubleshooting
Check the following if the role does not have the capabilities that it should:

- Are the role’s rights profiles listed in the GUI from most to least powerful?
  For example, if the All rights profile is at the top of the list, then no commands are run with security attributes. A profile that contains commands with security attributes must precede the All rights profile in the list.

- Do the commands in the role’s rights profiles have the appropriate security attributes?
  For example, when the policy is suser, some commands require uid=0 rather than euid=0.

- Is the rights profile defined in the appropriate name service scope? Is the role operating in the name service scope where the rights profile is defined?

- Has the name service cache, svc:/system/name-service-cache, been restarted?
  The nscd daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the name service with current data.

How to Create a Role From the Command Line

The Solaris Management Console GUI is the preferred method for managing RBAC. To use the GUI, see “How to Create and Assign a Role By Using the GUI” on page 191. You can also use the command-line interfaces, as described in this procedure.

Note – Do not attempt to administer RBAC with the command line and the graphical user interface at the same time. Conflicting changes could be made to the configuration, and the behavior would be unpredictable. You can use both tools to administer RBAC, but you cannot use both concurrently.

Before You Begin
To create a role, you must either assume a role that includes the Primary Administrator rights profile, or switch to the user root.

1  Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.
2 Choose one of the following commands to create a role on the command line.

- **For roles in the local name service scope, use the `roleadd` command.**

  **Note** – The `roleadd` command is more limited than the Solaris Management Console GUI or command-line interfaces. After running the `roleadd` command, you must run the `usermod` command to assign the role to a user. And, the user then must set the password for the role, as shown in “How to Assign a Role to a Local User” on page 196.

  ```
  # roleadd -c comment \\
  -g group -m homedir -u UID -s shell \\
  -P profile rolename
  -c comment Is a comment that describes rolename.
  -g group Is the group assignment for rolename.
  -m homedir Is the path to the home directory for rolename.
  -u UID Is the UID for rolename.
  -s shell Is the login shell for rolename. This shell must be a profile shell.
  -P profile Is one or more rights profiles for rolename.
  rolename Is the name of the new local role.
  ```

- **Use the `smrole add` command.**

  This command creates a role in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

  ```
  $ /usr/sadm/bin/smrole -D domain-name \\
  -r admin-role -l <Type admin-role password> \\
  add -- -n rolename -a rolename -d directory\ 
  -F full-description -p profile
  -D domain-name Is the name of the domain that you want to manage.
  -r admin-role Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
  -l Is the prompt for the password of `admin-role`.
  - Is the required separator between authentication options and subcommand options.
  -n rolename Is the name of the new role.
  -c comment Is the comment that describes the capabilities of the role.
  ```
Example 9–6  Creating a Custom Operator Role by Using the smrole Command

The smrole command specifies a new role and its attributes in a name service. In the following example, the Primary Administrator creates a new version of the Operator role. The role includes the standard Operator rights profile as well as the Media Restore rights profile. Note that the command prompts you for a password for the new role.

`% su primaryadm`
Password:  <Type primaryadm password>

$ /usr/sadm/bin/smrole add -H myHost -- -c "Backup and Restore Operator" \
   -n operadm2 -a janedoe -d /export/home/operadm \
   -F "Backup/Restore Operator" -p "Operator" -p "Media Restore"

Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::  <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm 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 ::  <Type operadm2 password>

$ svcadm restart system/name-service-cache

The smrole command with the list subcommand is used to display the new role:

`$ /usr/sadm/bin/smrole list --`

Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::  <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
How to Assign a Role to a Local User

This procedure assigns a local role to a local user, restarts the name cache daemon, and then shows how the user can assume the role.

To assign a role to a user in a distributed name service, see "How to Create a Role From the Command Line" on page 193 and "How to Change the Properties of a Role" on page 204.

Before You Begin

You have added a local role, as described in "How to Create a Role From the Command Line" on page 193. You must have assumed the role of Primary Administrator or have switched to superuser.

1 Assign the role to a local user.

If you added a local role with the roleadd command, this step is required. This step is optional when you use the smrole command and the Solaris Management Console to create a role.

```
# usermod -u UID -R rolename
-u UID Is the UID of the user.
-R rolename Is the role that is being assigned to the user.
```

2 To put the changes into effect, restart the name service cache daemon.

```
# svcadm restart system/name-service-cache
```

If you added a role with a Solaris Management Console interface, go to “Using Roles (Task Map)” on page 200. Otherwise, continue with the next step.

3 (Optional) To unlock the role account, the user must create a password.

If you added a local role with the roleadd command, this step is required.

```
% su rolename
Password: <Type rolename password>
Confirm Password: <Retype rolename password>
```

$
Creating and Assigning a Local Role From the Command Line

In this example, a role is created to administer the Solaris Cryptographic Framework. The Crypto Management rights profile contains the cryptoadm command for administering hardware and software cryptographic services on a local system.

```
# roleadd -c "Cryptographic Services manager" \
  -g 14 -m /export/home/cryptoadm -u 104 -s pfksh \
  -P "Crypto Management" cryptomgt
# usermod -u 1111 -R cryptomgt
# svcadm restart system/name-service-cache
% su cryptomgt
Password:  <Type cryptomgt password>
Confirm Password:  <Retype cryptomgt password>
$ /usr/ucb/whoami
cryptomgt
$
```

For information about the Solaris Cryptographic Framework, see Chapter 13. To administer the framework, see “Administering the Cryptographic Framework (Task Map)” on page 263.

How to Audit Roles

The actions that a role performs can be audited. Included in the audit record is the login name of the user who assumed the role, the role name, and the action that the role performed. The 6180:AUE_prof_cmd:profile command:ua, as audit event collects the information. By preselecting the as class or the ua class, you can audit role actions.

1 Plan for auditing and edit the audit configuration files.
   For more information, see “Solaris Auditing (Task Map)” on page 523.

2 Include the ua class or the as class in the flags line of the audit_control file.
   # audit_control file
   dir: /var/audit
   flags: lo, as
   minfree: 20
   naflags: lo

   The ua class and the as class include other audit events. To see the audit events that are included in a class, read the audit_event file. You can also use the bsmrecord command, as shown in Example 29–22.

3 Finish configuring the auditing service, then enable auditing.
   For more information, see “Configuring and Enabling the Auditing Service” on page 533.
How to Make root User Into a Role

This procedure shows how to change root from a login user to a role. When you complete this procedure, you can no longer log in to the system as root, except in single-user mode. You can su to root if the root role has been assigned to you.

By changing the root user into a role, you prevent anonymous root login. Because a user must log in and then assume the root role, the user’s login ID is provided to the auditing service and is in the su log file.

Before You Begin

If you change the root user into a role without assigning the role to a valid user or without a currently existing role that is equivalent to the root user, no one can become superuser.

- For safety, at least one local user should be assigned the root role.
- You cannot perform this procedure when you are logged in as root. You must log in as yourself, then su to root.

1 As an ordinary user, log in to the target host.

2 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.

3 Create a local user who can assume the root role.

$ useradd -c "comment" -d homedir username

- c comment Is the comment that describes the user.
- d homedir Is the home directory of the user. This directory should be on the local system.

username Is the name of the new local user.

# useradd -c "Local administrative user" -d /export/homel admuser

4 Give the user a password.

# passwd -r files admuser

New Password: <Type password>
Re-enter new Password: <Retype password>

passwd: password successfully changed for admuser

#

5 Make sure that you are not logged in as root.

# who

jdoe console May 24 13:51 (:0)
jdoe pts/5 May 24 13:51 (:0.0)
6 Change root user into a role.
   
   # usermod -K type=role root

7 Verify that root is a role.
   
   The root entry in the user_attr file should appear similar to the following:
   
   root:::type=role;auths=solaris.*,solaris.grant;profiles=Web Console Management,All;lock_after_retries=no

8 Assign the root role to the local administrative user.
   
   # usermod -R root admuser

9 Configure the name service to return in case of failure.
   
   a. Open a new terminal window and assume the root role.
      
      % whoami
      jdoe
      % su admuser
      Enter password: <Type admuser password>
      % roles
      root
      % su root
      Enter password: <Type root password>
      #

   b. Edit the nsswitch.conf file.
      
      For example, the following entries in the nsswitch.conf file would enable the name service to return.
      
      passwd: files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
      group: files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]

10 Assign the root role to selected user accounts in the name service.
    
    For the procedure, see “How to Change the RBAC Properties of a User” on page 208.
Using Roles (Task Map)

The following task map points to procedures for using your role after roles have been assigned.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Solaris Management Console</td>
<td>Authenticate yourself as a role to perform administrative tasks in the Solaris Management Console.</td>
<td>“How to Assume a Role in the Solaris Management Console” on page 202</td>
</tr>
<tr>
<td>Assume a role in a terminal window</td>
<td>Perform command-line administrative tasks in a profile shell.</td>
<td>“How to Assume a Role in a Terminal Window” on page 200</td>
</tr>
</tbody>
</table>

Using Roles

After you have set up roles with default Solaris rights profiles, and assigned the roles to users, the roles can be used. A role can be assumed on the command line. In the Solaris Management Console, a role can also be used for administering the system locally and over the network.

▼ How to Assume a Role in a Terminal Window

Before You Begin

The role must already be assigned to you. The name service must be updated with that information.

1 In a terminal window, determine which roles you can assume.
   
   % roles
   
   Comma-separated list of role names is displayed

2 Use the su command to assume a role.
   
   % su rolename
   
   Password: <Type rolename password>

   $ 
   
   The su command with a role name changes the shell to a profile shell for the role. A profile shell recognizes security attributes (authorizations, privileges, and set ID bits).

3 Verify that you are now in a role.
   
   $ /usr/ucb/whoami rolename
   
   You can now perform role tasks in this terminal window.

4 (Optional) View the capabilities of your role.
   
   For the procedure, see “How to Determine the Privileged Commands That a Role Can Run” on page 240.
Example 9-8  Assuming the Primary Administrator Role

In the following example, the user assumes the role of Primary Administrator. In the default configuration, this role is equivalent to superuser. The role then checks to see which privileges are available to any command that is typed in the profile shell for the role.

```bash
% roles
sysadmin,oper,primaryadm
% su primaryadm
Password: <Type primaryadm password>
$ /usr/ucb/whoami Prompt has changed to role prompt
primaryadm
$ ppriv $$
1200: pfksh
flags = <none>
    E (Effective): all
    I (Inheritable): basic
    P (Permitted): all
    L (Limit): all
```
For information about privileges, see “Privileges (Overview)” on page 178.

Example 9-9  Assuming the root Role

In the following example, the user assumes the root role. The role was created in “How to Make root User Into a Role” on page 198

```
% roles
root
% su root
Password: <Type root password>
#/usr/ucb/whoami Prompt has changed to role prompt
root
$ ppriv $$
1200: pfksh
flags = <none>
    E: all
    I: basic
    P: all
    L: all
```
For information about privileges, see “Privileges (Overview)” on page 178.

Example 9-10  Assuming the System Administrator Role

In the following example, the user assumes the role of System Administrator. In contrast to the Primary Administrator role, the System Administrator has the basic set of privileges in its effective set.
% roles
sysadmin,oper,primaryadm
% su sysadmin
Password: <Type sysadmin password>
$ /usr/ucb/whoami
Prompt has changed to role prompt
sysadmin
$ ppriv $$
1200: pfksh
flags = <none>
  E: basic
  I: basic
  P: basic
  L: all

For information about privileges, see "Privileges (Overview)" on page 178. For a short description of the capabilities of the role, see "System Administrator Rights Profile" on page 214.

▼ How to Assume a Role in the Solaris Management Console

To change information in the Solaris Management Console GUI requires administrative capabilities. A role gives you administrative capabilities. If you want to view information, you must have the solaris.admin.usermgr.read authorization. The Basic Solaris User rights profile includes this authorization.

Before You Begin
An administrative role that can change the properties of users or roles must have already been assigned to you. For example, the Primary Administrator role can change the properties of users or roles.

1 Start the Solaris Management Console.
% /usr/sbin/smc &

For detailed instructions, see "Using the Solaris Management Tools With RBAC (Task Map)" in System Administration Guide: Basic Administration.

2 Select the toolbox for your task.
Navigate to the toolbox that contains the tool or collection in the appropriate name service 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 that you want to use.
Navigate to the tool or collection and click the icon. The tools for managing the RBAC elements are in the Users tool, as shown in the following figure.
4 Type your user name and password in the Login: User Name dialog box.

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.

Managing RBAC (Task Map)

The following task map points to procedures for customizing role-based access control (RBAC) after RBAC has been initially implemented.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify the properties of a role</td>
<td>Modifies the capabilities (privileges, privileged commands, profiles, or authorizations) of a role.</td>
<td>&quot;How to Change the Properties of a Role&quot; on page 204</td>
</tr>
<tr>
<td>Create or change rights profiles</td>
<td>Creates a rights profile. Or modifies the authorizations, privileged commands, or supplementary rights profiles in a rights profile.</td>
<td>&quot;How to Create or Change a Rights Profile&quot; on page 206</td>
</tr>
</tbody>
</table>
### Managing RBAC

The Solaris Management Console GUI is the preferred method for managing RBAC.

---

**Note** – Do not attempt to administer RBAC with the command line and the graphical user interface at the same time. Conflicting changes could be made to the configuration, and the behavior would be unpredictable. Both tools can administer RBAC, but you cannot use both tools concurrently.

### How to Change the Properties of a Role

**Before You Begin**

You must have assumed the role of Primary Administrator or have switched to superuser to change the properties of a role. Role properties include password, rights profiles, and authorizations.

- Use one of the following methods to change the properties of a role.
  - **Use the Users tool in the Solaris Management Console.**
    
    To start the console, see “How to Assume a Role in the Solaris Management Console” on page 202. Follow the instructions in the left-hand pane to modify a role in Administrative Roles. For more extensive information, see the online help.
  
  - **Use the rolemod command.**
    
    This command modifies the attributes of a role that is defined in the local name service.

    ```sh
    rolemod -c comment -P profile-list rolename
    ```
    
    - `-c comment` Is the new comment that describes the capabilities of the role.
    - `-P profile-list` Is the list of the profiles that are included in the role. This list replaces the current list of profiles.
    - `rolename` Is the name of an existing, local role that you want to modify.
Use the `smrole` command with the `modify` subcommand.

This command modifies the attributes of a role in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

```
$ /usr/sadm/bin/smrole -D domain-name \
  -r admin-role -l <Type admin-role password> \ 
  modify -- -n rolename -r username -u username
```

- `-D domain-name` is the name of the domain that you want to manage.
- `-r admin-role` is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
- `-l` is the prompt for the password of `admin-role`.
- `--` is the required separator between authentication options and subcommand options.
- `-n rolename` is the name of the new role.
- `-r username` is the name of the user who can no longer assume `rolename`.
- `-u username` is the name of the user who can now assume `rolename`.

For more command options, see the `smrole(1M)` man page.

**Example 9–11** Changing a Local Role's Properties With the `rolemod` Command

In this example, the `operadm` role is modified to include the Media Restore rights profile.

```
$ rolemod -c "Handles printers, backup, AND restore" \
  -P "Printer Management,Media Backup,Media Restore,All" operadm
```

**Example 9–12** Changing a Local Role's Properties With the `smrole modify` Command

In the following example, the `operadm` role is modified to add the Media Restore rights profile.

```
$ /usr/sadm/bin/smrole -r primaryadm -l <Type primaryadm password> \ 
  modify -- -n operadm -c "Handles printers, backup, AND restore" \ 
  -p "Media Restore"
```

**Example 9–13** Changing a Role in a Domain With the `smrole modify` Command

In the following example, the `clockmgr` role is changed. The NIS user whose ID is 108 can no longer assume the role. The NIS user whose ID is 110 can assume the role `clockmgr`.
How to Create or Change a Rights Profile

A rights profile is a property of a role. You should create or change a rights profile when the prof_attr database does not contain a rights profile that fulfills your needs. To learn more about rights profiles, see "RBAC Rights Profiles" on page 175.

Before You Begin

To create or change a rights profile, you must have assumed the role of Primary Administrator or have switched to superuser.

Use one of the following methods to change the properties of a role.

- Use the Users tool in the Solaris Management Console.
  
  To start the console, see "How to Assume a Role in the Solaris Management Console" on page 202. Follow the instructions in the left-hand pane to create or change a rights profile in Rights. For more extensive information, see the online help.

- Use the smprofile command.
  
  This command enables you to add, modify, list, or delete a rights profile. The command works on files, and in a distributed name service, such as NIS, NIS+, or LDAP. The smprofile command runs as a client of the Solaris Management Console server.

```bash
$ /usr/sadm/bin/smprofile -D domain-name \ 
- r admin-role - l <Type admin-role password> \ 
add | modify -- - n profile-name \ 
- d description - m help-file - p supplementary-profile
```

- **-D domain-name** Is the name of the domain that you want to manage.

- **-r admin-role** Is the name of the administrative role that can modify the role. The administrative role must have the solaris.role.assign authorization. If you are modifying a role that you have assumed, the role must have the solaris.role.delegate authorization.

- **-l** Is the prompt for the password of admin-role.

- **--** Is the required separator between authentication options and subcommand options.

- **-n profile-name** Is the name of the new profile.

- **-d description** Is a short description of the profile.
-m help-file  
Is the name of the HTML help file that you have created and placed in the /usr/lib/help/profiles/locale/C directory.

- p supplementary-profile  
Is the name of an existing rights profile that is included in this rights profile. You can specify multiple - p supplementary-profile options.

For more command options, see the smprofile(1M) man page.

Example 9–14  Modifying a Rights Profile From the Command Line

In the following example, the Network Management rights profile is made a supplementary profile of the Network Security rights profile. The role that contains the Network Security profile can now configure the network and hosts, as well has run security-relevant commands.

```
$ /usr/sadm/bin/smprofile -D nisplus:/example.host/example.domain -r primaryadm -l <Type primaryadm password> 
modify -- -n "Network Security" -d "Manage network and host configuration and security" -m RtNetConfSec.html -p "Network Management"
```

The administrator created a new help file, RtNetConfSec.html, and placed it in the /usr/lib/help/profiles/locale/C directory, before running this command.

Example 9–15  Creating a New Rights Profile With the Rights Tool

The following table shows sample data for a hypothetical rights profile that is called “Build Administrator”. This rights profile includes the commands in the subdirectory /usr/local/swctrl/bin. These commands have an effective UID of 0. The Build Administrator rights profile would be useful for administrators who manage the builds and versioning for software development.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Field</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Name</td>
<td>Build Administrator</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>For managing software builds and versioning.</td>
</tr>
<tr>
<td></td>
<td>Help File Name</td>
<td>BuildAdmin.html</td>
</tr>
<tr>
<td>Commands</td>
<td>Add Directory</td>
<td>Click Add Directory, type /usr/local/swctrl/bin in the dialog box, and click OK.</td>
</tr>
<tr>
<td></td>
<td>Commands Denied / Commands Permitted</td>
<td>Move /usr/local/swctrl/bin to the Commands Permitted column.</td>
</tr>
<tr>
<td></td>
<td>Set Security Attributes</td>
<td>Select /usr/local/swctrl/bin, click Set Security Attributes, and set Effective UID = root.</td>
</tr>
</tbody>
</table>
**Managing RBAC**

<table>
<thead>
<tr>
<th>Tab</th>
<th>Field</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorizations</td>
<td>Authorizations Excluded / Authorizations Included</td>
<td>No authorizations.</td>
</tr>
<tr>
<td>Supplementary Rights</td>
<td>Rights Excluded / Rights Included</td>
<td>No supplementary rights profiles.</td>
</tr>
</tbody>
</table>

**Troubleshooting**

Check the following if the rights profile does not provide the role with the capabilities that you expect:

- Are the rights profiles for the role listed in the GUI from most to least powerful?
  
  For example, if the All rights profile is at the top of the list, then no commands are run with security attributes. A profile that contains commands with security attributes must precede the All rights profile in the list.

- Is a command listed more than once in the role’s rights profiles? If so, does the first instance of the command have all the security attributes that are required?
  
  For example, a command can require privileges for particular options to the command. For the options that require privileges to succeed, the first instance of the command in the highest rights profile in the list must have the assigned privileges.

- Do the commands in the role’s rights profiles have the appropriate security attributes?
  
  For example, when the policy is suser, some commands require uid=0 rather than euid=0 to succeed.

- Has the name service cache, svc:/system/name-service-cache, been restarted?
  
  The nscd daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the name service with current data.

**How to Change the RBAC Properties of a User**

User properties include password, rights profiles, and authorizations. The most secure method of giving a user administrative capabilities is to assign a role to the user. For a discussion, see “Security Considerations When Directly Assigning Security Attributes” on page 177.

**Before You Begin**

You must have assumed the role of Primary Administrator or have switched to superuser to change the properties of a user.

- Use one of the following methods to change the RBAC properties of a user.

  - **Use the Users tool in the Solaris Management Console.**
    
    To start the console, see “How to Assume a Role in the Solaris Management Console” on page 202. Follow the instructions in the left-hand pane to modify a user in User Accounts. For more extensive information, see the online help.
Tip – It is not good practice to assign authorizations, privileges, or rights profiles directly to users. The preferred approach is to assign a role to users. Users then assume a role to perform privileged operations.

- **Use the `usermod` command.**

  This command modifies the attributes of a user that is defined in the local name service.

  ```
  $ usermod -R rolename username
  
  -R rolename    Is the name of an existing local role.
  
  username       Is the name of an existing, local user that you want to modify.
  
  For more command options, see the `usermod(1M)` man page.
  ```

- **Use the `smuser` command with the `modify` subcommand.**

  This command modifies the attributes of a user in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

  ```
  $ /usr/sadm/bin/smuser -D domain-name \
  -r admin-role -l <Type admin-role password> \
  modify -- -n username -a rolename
  
  -D domain-name   Is the name of the domain that you want to manage.
  
  -r admin-role   Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
  
  -l               Is the prompt for the password of `admin-role`.
  
  -   Is the required separator between authentication options and subcommand options.
  
  -n username      Is the name of the user who is being assigned `rolename`.
  
  -a rolename      Is the name of the role that you are assigning to `username`. You can specify multiple `-a rolename` options.
  
  For more command options, see the `smuser(1M)` man page.

**Example 9–16** Modifying a Local User’s RBAC Properties From the Command Line

In this example, the user `jdoe` can now assume the role of System Administrator.

```
$ usermod -R sysadmin jdoe
```
Example 9–17  Modifying a User’s RBAC Properties With the `smuser` Command

In this example, the user `jdoe` is assigned two roles, System Administrator and Operator. Because the user and the roles are defined locally, the `-D` option is not necessary.

```bash
$ /usr/sadm/bin/smuser -r primaryadm -l <Type primaryadm password> \
modify -- -n jdoe -a sysadmin -a operadm
```

In the following example, the user is defined in the NIS name service. Therefore, the `-D` option is required. Two roles are defined in the name service. One role, `root`, is defined locally.

```bash
$ /usr/sadm/bin/smuser -D nis:/examplehost/example.domain \
   -r primaryadm -l <Type primaryadm password> \
modify -- -n jdoe -a sysadmin -a operadm -a root
```

How to Add RBAC Properties to Legacy Applications

A legacy application is a command or set of commands. The security attributes are set for each command in a rights profile. The rights profile is then included in a role. A user who assumes the role can run the legacy application with the security attributes.

To add legacy applications to the Solaris Management Console, see “Adding Tools to the Solaris Management Console” in System Administration Guide: Basic Administration.

Before You Begin

You must have assumed the role of Primary Administrator or have switched to superuser to change the security attributes of a command in a rights profile.

1  Use the Users tool in the Solaris Management Console.
   To start the console, see “How to Assume a Role in the Solaris Management Console” on page 202. Follow the instructions in the left-hand pane to modify a rights profile in Rights. For more extensive information, see the online help.

2  Add security attributes to the commands that implement the legacy application.
   You add security attributes to a legacy application in the same way that you would for any command. You must add the command with security attributes to a rights profile. For a legacy command, give the command `euid=0` or `uid=0` security attributes. For details of the procedure, see “How to Create or Change a Rights Profile” on page 206.

3  After adding the legacy application to a rights profile, include the rights profile in a role’s list of profiles.
   To add a rights profile to a role, see “How to Change the Properties of a Role” on page 204.
Adding Security Attributes to Commands in a Script

If a command in a script needs to have the setuid bit or setgid bit set to succeed, the script executable and the command must have the security attributes added in a rights profile. Then, the rights profile is included in a role, and the role is assigned to a user. When the user assumes the role and executes the script, the command runs with the security attributes.

To add security attributes to a command or shell script, see “How to Create or Change a Rights Profile” on page 206.

Checking 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 command. For detailed information about this command, see the auths(1) man page.

For example, the following line tests if the user has the authorization that is supplied 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 would need to check for the following strings:

- solaris.admin.usermgr.write
- solaris.admin.usermgr.*
- solaris.admin.*
- solaris.*

If you are writing a program, use the function getauthattr() to test for the authorization.
Role-Based Access Control (Reference)

This chapter provides reference material about RBAC. The following is a list of the reference information in this chapter:

- “Contents of Rights Profiles” on page 213
- “Authorization Naming and Delegation” on page 218
- “Databases That Support RBAC” on page 219
- “RBAC Commands” on page 225

For information on using RBAC, see Chapter 9. For overview information, see “Role-Based Access Control (Overview)” on page 169.

Contents of Rights Profiles

This section describes some typical rights profiles. Rights profiles can include authorizations, commands with security attributes, and supplementary rights profiles. The rights profiles are listed from most to least powerful. For suggestions on how to distribute rights profiles to roles at your site, see “How to Plan Your RBAC Implementation” on page 189.

- **Primary Administrator rights profile** – Provides the capabilities of superuser in one profile.
- **System Administrator rights profile** – Provides a profile that can do most tasks that are not connected with security. This profile includes several other profiles to create a powerful role.
- **Operator rights profile** – Provides limited capabilities to manage files and offline media. This profile includes supplementary rights profiles to create a simple role.
- **Printer Management rights profile** – Provides a limited number of commands and authorizations to handle printing. This profile is one of several profiles that cover a single area of administration.
- **Basic Solaris User rights profile** – Enables users to use the system within the bounds of security policy. This profile is listed by default in the policy.conf file.
- **All rights profile** – For roles, provides access to commands that do not have security attributes.
Each rights profile has an associated help file. The help files are in HTML and are customizable. The files reside in the /usr/lib/help/auths/locale/C directory.

**Primary Administrator Rights Profile**

The Primary Administrator rights profile is assigned to the most powerful role on the system. The role that includes the Primary Administrator rights profile has 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.

You can customize the help file `RtPriAdmin.html` for your site, 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, the profile 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. Those other rights profiles would then be assigned to roles.

**TABLE 10–1 Contents of Primary Administrator Rights Profile**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To perform all administrative tasks</td>
<td><strong>Commands:</strong> <code>::uid=0;gid=0</code></td>
</tr>
<tr>
<td></td>
<td><strong>Authorizations:</strong> <code>solaris.*</code>, <code>solaris.grant</code></td>
</tr>
<tr>
<td></td>
<td><strong>Help File:</strong> <code>RtPriAdmin.html</code></td>
</tr>
</tbody>
</table>

**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 capabilities of the Primary Administrator, no wildcards are used. Instead, this profile is a set of discrete, supplementary administrative rights profiles that do not deal with security. The commands with security attributes from one of the supplementary rights profiles are shown.

Note that the All rights profile is assigned at the end of the list of supplementary rights profiles.
**TABLE 10–2 Contents of System Administrator Rights Profile**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To perform most nonsecurity administrative tasks</td>
<td><strong>Supplementary rights profiles:</strong> 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</td>
</tr>
<tr>
<td></td>
<td><strong>Help File:</strong> RtSysAdmin.html</td>
</tr>
<tr>
<td>Commands from one of the supplementary profiles</td>
<td><strong>Object Access Management rights profile:</strong> solaris policy:</td>
</tr>
<tr>
<td></td>
<td>/usr/bin/chgrp:privs=file_chown, /usr/bin/chmod:privs=file_chown,</td>
</tr>
<tr>
<td></td>
<td>/usr/bin/chown:privs=file_chown, /usr/bin/setfacl:privs=file_chown</td>
</tr>
<tr>
<td></td>
<td>suser policy: /usr/bin/chgrp:euid=0, /usr/bin/chmod:euid=0,</td>
</tr>
<tr>
<td></td>
<td>/usr/bin/chown:euid=0, /usr/bin/getfacl:euid=0,</td>
</tr>
<tr>
<td></td>
<td>/usr/bin/setfacl:euid=0</td>
</tr>
</tbody>
</table>

**Operator Rights Profile**

The Operator rights profile is a less powerful profile that provides the ability to do backups and printer maintenance. The ability to restore files has more security consequences. Therefore, in this profile, the default is to not include the ability to restore files.

**TABLE 10–3 Contents of Operator Rights Profile**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To perform simple administrative tasks</td>
<td><strong>Supplementary rights profiles:</strong> Printer Management, Media Backup, All</td>
</tr>
<tr>
<td></td>
<td><strong>Help File:</strong> RtOperator.html</td>
</tr>
</tbody>
</table>

**Printer Management Rights Profile**

Printer Management is a typical rights profile that is intended for a specific task area. This profile includes authorizations and commands. The following table shows a partial list of commands.
Basic Solaris User Rights Profile

By default, the Basic Solaris User rights profile is assigned automatically to all users through the policy.conf file. This profile provides basic authorizations that are useful in normal operations. Note that the convenience that is offered by the Basic Solaris User rights profile must be balanced against site security requirements. Sites that need stricter security might prefer to remove this profile from the policy.conf file.

TABLE 10–5 Contents of Basic Solaris User Rights Profile

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Authorizations</th>
<th>Supplementary rights profiles</th>
<th>Help File</th>
</tr>
</thead>
<tbody>
<tr>
<td>To automatically assign rights to all users</td>
<td>solaris.profmgr.read, solaris.jobs.users, solaris.mail.mailq, solaris.admin.usermgr.read, solaris.admin.logsvc.read, solaris.admin.fsmgr.read, solaris.admin.serialmgr.read, solaris.admin.diskmgr.read, solaris.admin.procmgr.user, solaris.compsys.read, solaris.admin.printer.read, solaris.admin.prodreg.read, solaris.admin.dcmgr.read, solaris.project.read, solaris.admin.patchmgr.read, solaris.network.hosts.read, solaris.compsys.read, solaris.admin.volmgr.read</td>
<td>All</td>
<td>Rtdefault.html</td>
</tr>
</tbody>
</table>
All Rights Profile

The All rights profile uses the wildcard to include all commands. This profile provides a role with access to all commands that are not explicitly assigned in other rights profiles. Without the All rights profile or other rights profiles that use wildcards, a role has access to explicitly assigned commands only. Such a limited set of commands is not very practical.

The All rights profile, if used, should be the final rights profile that is assigned. This last position ensures that explicit security attribute assignments in other rights profiles are not inadvertently overridden.

<table>
<thead>
<tr>
<th>Table 10-6 Contents of All Rights Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
</tr>
<tr>
<td>------------------------------------------</td>
</tr>
<tr>
<td>To execute any command as the user or role</td>
</tr>
<tr>
<td>Help File: RtAll.html</td>
</tr>
</tbody>
</table>

Order of Rights Profiles

The commands in rights profiles are interpreted in order. The first occurrence of a command is the only version of the command that is used for that role or user. Different rights profiles can include the same command. Therefore, the order of rights profiles in a list of profiles is important. The rights profile with the most capabilities should be listed first.

Rights profiles are listed in the Solaris Management Console GUI and in the prof_attr file. In the Solaris Management Console GUI, the rights profile with the most capabilities should be the top profile in a list of assigned rights profiles. In the prof_attr file, the rights profile with the most capabilities should be the first in a list of supplementary profiles. This placement ensures that a command with security attributes is listed before that same command without security attributes.

Viewing the Contents of Rights Profiles

The Solaris Management Console Rights tool provides one way of inspecting the contents of the rights profiles.

The prof_attr and exec_attr files offer a more fragmented view. The prof_attr file contains the name of every rights profile that is defined on the system. The file also includes the authorizations and the supplementary rights profiles for each profile. The exec_attr file contains the names of rights profiles and their commands with security attributes.
Authorization Naming and Delegation

An RBAC authorization is a discrete right that can be granted to a role or a user. Authorizations are checked by RBAC-compliant applications before a user gets access to the application or specific operations within the application. This check replaces the tests in conventional UNIX applications for UID=0.

Authorization Naming Conventions

An authorization has a name that is used internally and in files. For example, `solaris.admin.usermgr.pswd` is the name of an authorization. An authorization has a short description, which appears in the graphical user interfaces (GUIs). For example, Change Passwords is the description of the `solaris.admin.usermgr.pswd` authorization.

By convention, authorization names consist of the reverse order of the Internet name of the supplier, the subject area, any subareas, and the function. The parts of the authorization name are separated by dots. An example would be `com.xyzcorp.device.access`. Exceptions to this convention are the authorizations from Sun Microsystems, Inc., which use the prefix `solaris` instead of an Internet name. The naming convention enables administrators to apply authorizations in a hierarchical fashion. A wildcard (`*`) can 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 user configuration files. The System Administrator role naturally has the `solaris.admin.usermgr.read` and the `solaris.admin.usermgr.write` authorizations for making changes to user files. However, 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.

Delegation Authority in Authorizations

An authorization that ends with the suffix `grant` enables a user or a 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.*` authorizations can delegate any of the authorizations with the `solaris.admin.usermgr` prefix to other users.
Databases That Support RBAC

The following four databases store the data for the RBAC elements:

- **Extended user attributes database** (`user_attr`) – Associates users and roles with authorizations and rights
- **Rights profile attributes database** (`prof_attr`) – Defines rights profiles, lists the profiles' assigned authorizations, and identifies the associated help file
- **Authorization attributes database** (`auth_attr`) – Defines authorizations and their attributes, and identifies the associated help file
- **Execution attributes database** (`exec_attr`) – Identifies the commands with security attributes that are assigned to specific rights profiles

The `policy.conf` database contains authorizations, privileges, and rights profiles that are applied to all users. For more information, see “`policy.conf` File” on page 224.

RBAC Database Relationships

Each RBAC database uses a `key=value` syntax for storing attributes. This method accommodates future expansion of the databases. The method also enables a system to continue to operate if the system encounters a keyword that is unknown to its policy. The `key=value` contents link the files. The following linked entries from the four databases illustrate how the RBAC databases work together.

**EXAMPLE 10-1** Showing RBAC Database Connections

In the following example, the user `jdoe` gets the capabilities of the File System Management profile through being assigned the role `filemgr`.

1. The user `jdoe` is assigned the role `filemgr` in the `jdoe` user entry in the `user_attr` database.

   ```
   # user_attr - user definition
   jdoe:::type=normal;roles=filemgr
   ```

2. The role `filemgr` is assigned the rights profile `FileSystem Management` in the role's entry in the `user_attr` database.

   ```
   # user_attr - role definition
   filemgr:::profiles=File System Management;type=role
   ```

   The user and the role are uniquely defined in the `passwd` and `shadow` files on the local system, or in equivalent databases in a distributed name service.

3. The File System Management rights profile is defined in the `prof_attr` database. This database also assigns three sets of authorizations to the File System Management entry.

   ```
   # prof_attr - rights profile definitions and assigned authorizations
   File System Management:::Manage, mount, share file systems:
   help=RtFileSysMngmnt.html;
   auths=solaris.admin.fsmgr.*,solaris.admin.diskmgr.*,solaris.admin.volmgr.*
   ```
4. The authorizations are defined in the auth_attr database.

```plaintext
# auth_attr - authorization definitions
solaris.admin.fsmgr:::Mounts and Shares::help=AuthFsmgrHeader.html
solaris.admin.fsmgr.read:::View Mounts and Shares::help=AuthFsmgrRead.html
solaris.admin.fsmgr.write:::Mount and Share Files::help=AuthFsmgrWrite.html
```

5. The File System Management rights profile is assigned commands with security attributes in the exec_attr database.

```plaintext
# exec_attr - rights profile names with secured commands
File System Management:suser:cmd:::/usr/sbin/mount:uid=0
File System Management:suser:cmd:::/usr/sbin/dfshares:euid=0
...
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
...
```

### RBAC Databases and the Name Service

The name service scope of the RBAC databases can apply to the local host only. The scope can also include all hosts that are served by a name service such as NIS, NIS+, or LDAP. Which name service has precedence is set for each of the databases in the /etc/nsswitch.conf file.

- **auth_attr entry** – Sets the name service precedence for the auth_attr database.
- **passwd entry** – Sets the name service precedence for the user_attr database.
- **prof_attr entry** – Sets the name service precedence for the prof_attr database. Also sets the name service precedence for the exec_attr database.

For example, if a command with security attributes is assigned to a rights profile that exists in two name service scopes, only the entry in the first name service scope is used.

### user_attr Database

The user_attr database contains user and role information that supplements the passwd and shadow databases. The user_attr database 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:

```plaintext
user:qualifier:res1:res2:attr
```

The fields have the following meanings:

- **user**
  - The name of the user or role as specified in the passwd database.
- **qualifier:res1:res2**
  - These fields are reserved for future use.
attr
An optional list of semicolon-separated (;) key-value pairs that describes the security attributes to be applied when the user runs commands. The four valid keys are type, auths, profiles, and roles.

- The type keyword can be set to normal, if this account is for a normal user. The type is role if this account is for a role.
- The auths keyword specifies a comma-separated list of authorization names that are chosen from names that are defined in the auth_attr database. Authorization names can include the asterisk (*) character as a wildcard. For example, solaris.device.* means all of the Solaris device authorizations.
- The profiles keyword specifies an ordered, comma-separated list of rights profile names from the prof_attr database. The order of rights profiles works similarly to UNIX search paths. The first profile in the list that contains the command to be executed defines which (if any) security attributes are to be applied to the command.
- The roles keyword can be assigned to the user through a comma-separated list of role names. Note that roles are defined in the same user_attr database. Roles are indicated by setting the type value to role. Roles cannot be assigned to other roles.

The following example demonstrates how the Operator role is defined in a typical user_attr database. The example shows how the role is assigned to user jdoe. Roles and users are differentiated by the type keyword.

% grep operator /etc/user_attr
jdoe:::::type=normal;roles=operator
operator:::::profiles=Operator;type=role

auth_attr Database

All authorizations are stored in the auth_attr database. Authorizations can be assigned to users, to roles, or to rights profiles. The preferred method is to place authorizations in a rights profile, to include the profile in a role’s list of profiles, and then to assign the role to a user.

The fields in the auth_attr database are separated by colons, as follows:
authname:res1:res2:short_desc:long_desc:attr

The fields have the following meanings:

authname A unique character string that is used to identify the authorization in the format prefix.[suffix]. Authorizations for the Solaris OS use solaris 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, com.xyz.company). The suffix indicates what is being authorized, which is typically the functional area and operation.

When the authname consists of a prefix and functional area and ends with a period, the authname serves as a heading to be used by applications in their GUIs. A
Two-part authname is not an actual authorization. The authname of `solaris.printmgr` is an example of a heading.

When authname ends with the word "grant," the authname serves as a grant authorization. A grant authorization enables the user to delegate to other users authorizations with the same prefix and functional area. The authname of `solaris.printmgr.grant` is an example of a grant authorization.

`solaris.printmgr.grant` gives the user the right to delegate to other users such authorizations as `solaris.printmgr.admin` and `solaris.printmgr.nobanner`.

**Reserved for future use.**

**A short name for the authorization. This short name is suitable for display in user interfaces, such as in a scrolling list in a GUI.**

**A long description. This field identifies the purpose of the authorization, the applications in which the authorization is used, and the type of user who might use the authorization. The long description can be displayed in the help text of an application.**

**An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys can be specified.**

The keyword `help` identifies a help file in HTML. Help files can be accessed from the `index.html` file in the `/usr/lib/help/auths/locale/C` directory.

The following example shows an auth_attr database with some typical values:

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

Note that `solaris.admin.printer.` is defined as a heading, because the authorization name ends in a dot (`.`). Headings are used by the GUIs to organize families of authorizations.

### prof_attr Database

The prof_attr database stores the name, description, help file location, and authorizations that are assigned to rights profiles. The commands and security attributes that are assigned to rights profiles are stored in the exec_attr database. For more information, see "exec_attr Database" on page 223.

The fields in the prof_attr database are separated by colons, as follows:

```
profname:res1:res2:desc:attr
```

The fields have the following meanings:
profname  The name of the rights profile. Rights profile names are case-sensitive. This name is also used by the user_attr database to indicate the profiles that are assigned to roles and users.

res1:res2  Reserved for future use.

desc  A long description. This field should explain the purpose of the rights profile, including what type of user would be interested in using the profile. The long description should be suitable for display in the help text of an application.

attr  An optional list of key-value pairs that are separated by semicolons (;) that describes the security attributes to apply to the object on execution. Zero or more keys can be specified. The two valid keys are help and auths.

   The keyword help identifies a help file in HTML. Help files can be accessed from the index.html file in the /usr/lib/help/auths/locale/C directory.

   The keyword auths specifies a comma-separated list of authorization names that are chosen from those names that are defined in the auth_attr database. Authorization names can be specified with the asterisk (*) character as a wildcard.

The following example shows two typical prof_attr database entries. Note that the Printer Management rights profile is a supplementary rights profile of the Operator rights profile. The example is wrapped for display purposes.

% grep 'Printer Management' /etc/security/prof_attr
Printer Management::: Name of rights profile
Manage printers, daemons, spooling: Description
help=RtPrntAdmin.html; Help file
auths=solaris.admin.printer.read, Authorizations
solaris.admin.printer.modify,solaris.admin.printer.delete ...

Operator::: Name of rights profile
Can perform simple administrative tasks: Description
profiles=Printer Management, Supplementary rights profiles
Media Backup,All;
help=RtOperator.html Help file

exec_attr Database

The exec_attr database defines commands that require security attributes to succeed. The commands are part of a rights profile. A command with its security attributes can be run by roles to whom the profile is assigned.

The fields in the exec_attr database are separated by colons, as follows:

name:policy:type:res1:res2:id:attr
The fields have the following meanings.

**profname**  
The name of the rights profile. Rights profile names are case-sensitive. The name refers to a profile in the `prof_attr` database.

**policy**  
The security policy that is associated with this entry. Currently, `suser` and `solaris` are the valid entries. The `solaris` policy recognizes privileges. The `suser` policy does not.

**type**  
The type of entity that is specified. Currently, the only valid entity type is `cmd` (command).

**res1:res2**  
Reserved for future use.

**id**  
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 `id` to the script.

**attr**  
An optional list of semicolon (;) separated key-value pairs that describes the security attributes to apply to the entity on execution. Zero or more keys can be specified. The list of valid keywords depends on the policy that is enforced.

For the `suser` policy, the four valid keys are `euid`, `uid`, `egid`, and `gid`.

- The `euid` and `uid` keywords contain a single user name or a numeric user ID (UID). Commands that are designated with `euid` run with the supplied UID, which is similar to setting the `setuid` bit on an executable file. Commands that are designated with `uid` run with both the real UID and the effective UID.

- The `egid` and `gid` keywords contain a single group name or numeric group ID (GID). Commands that are designated with `egid` run with the supplied GID, which is similar to setting the `setgid` bit on an executable file. Commands that are designated with `gid` run with both the real GID and the effective GID.

For the `solaris` policy, the valid keyword is `privs`. The value consists of a list of privileges that are separated by commas.

The following example shows some typical values from an `exec_attr` database:

```
% grep 'File System Management' /etc/security/exec_attr
File System Management:suser:cmd:::/usr/sbin/ff:euid=0
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
```

**policy.conf File**

The `policy.conf` file provides a way of granting specific rights profiles, specific authorizations, and specific privileges to all users. The relevant entries in the file consist of `key=value` pairs:

- `AUTHS_GRANTED=authorizations` – Refers to one or more authorizations.
- PROFS_GRANTED=rights profiles – Refers to one or more rights profiles.
- PRIV_DEFAULT=privileges – Refers to one or more privileges.
- PRIV_LIMIT=privileges – Refers to all privileges.

The following example shows some typical values from a policy.conf database:

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw

# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User

# grep PRIV /etc/security/policy
#PRIV_DEFAULT=basic
#PRIV_LIMIT=all
```

For more information about privileges, see "Privileges (Overview)" on page 178.

## RBAC Commands

This section lists commands that are used to administer RBAC. Also provided is a table of commands whose access can be controlled by authorizations.

### Commands That Manage RBAC

While you can edit the local RBAC databases manually, such editing is strongly discouraged. The following commands are available for managing access to tasks with RBAC.

**TABLE 10–7 RBAC Administration Commands**

<table>
<thead>
<tr>
<th>Man Page for Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auths(1)</td>
<td>Displays authorizations for a user.</td>
</tr>
<tr>
<td>makedbm(1M)</td>
<td>Makes a dbm file.</td>
</tr>
<tr>
<td>nscl(1M)</td>
<td>Name service cache daemon, useful for caching the user_attr, prof_attr, and exec_attr databases. Use the svcadm command to restart the daemon.</td>
</tr>
<tr>
<td>pam_roles(5)</td>
<td>Role account management module for PAM. Checks for the authorization to assume role.</td>
</tr>
<tr>
<td>pfexec(1)</td>
<td>Used by profile shells to execute commands with security attributes that are specified in the exec_attr database.</td>
</tr>
</tbody>
</table>
TABLE 10–7 RBAC Administration Commands  (Continued)

<table>
<thead>
<tr>
<th>Man Page for Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy.conf(4)</td>
<td>Configuration file for system security policy. Lists granted authorizations, granted privileges, and other security information.</td>
</tr>
<tr>
<td>profiles(1)</td>
<td>Displays rights profiles for a specified user.</td>
</tr>
<tr>
<td>roles(1)</td>
<td>Displays roles that a specified user can assume.</td>
</tr>
<tr>
<td>roleadd(1M)</td>
<td>Adds a role to a local system.</td>
</tr>
<tr>
<td>roledel(1M)</td>
<td>Deletes a role from a local system.</td>
</tr>
<tr>
<td>rolemod(1M)</td>
<td>Modifies a role’s properties on a local system.</td>
</tr>
<tr>
<td>smattrpop(1M)</td>
<td>Merges the source security attribute database into the target database. For use in situations where local databases need to be merged into a name service. Also for use in upgrades where conversion scripts are not supplied.</td>
</tr>
<tr>
<td>smexec(1M)</td>
<td>Manages entries in the exec_attr database. Requires authentication.</td>
</tr>
<tr>
<td>smmultiuser(1M)</td>
<td>Manages bulk operations on user accounts. Requires authentication.</td>
</tr>
<tr>
<td>smprofile(1M)</td>
<td>Manages rights profiles in the prof_attr and exec_attr databases. Requires authentication.</td>
</tr>
<tr>
<td>smrole(1M)</td>
<td>Manages roles and users in role accounts. Requires authentication.</td>
</tr>
<tr>
<td>smuser(1M)</td>
<td>Manages user entries. Requires authentication.</td>
</tr>
<tr>
<td>useradd(1M)</td>
<td>Adds a user account to the system. The -P option assigns a role to a user’s account.</td>
</tr>
<tr>
<td>userdel(1M)</td>
<td>Deletes a user’s login from the system.</td>
</tr>
<tr>
<td>usermod(1M)</td>
<td>Modifies a user’s account properties on the system.</td>
</tr>
</tbody>
</table>

Commands That Require Authorizations

The following table provides examples of how authorizations are used to limit command options on a Solaris system. For more discussion of authorizations, see “Authorization Naming and Delegation” on page 218.

TABLE 10–8 Commands and Associated Authorizations

<table>
<thead>
<tr>
<th>Man Page for Command</th>
<th>Authorization Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>at(1)</td>
<td>solaris.jobs.user required for all options (when neither at.allow nor at.deny files exist)</td>
</tr>
<tr>
<td>atq(1)</td>
<td>solaris.jobs.admin required for all options</td>
</tr>
<tr>
<td>Man Page for Command</td>
<td>Authorization Requirements</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>cdrw(1)</td>
<td><code>solaris.device.cdrw</code> required for all options, and is granted by default in the <code>policy.conf</code> file</td>
</tr>
</tbody>
</table>
| 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)          | `solaris.device.allocate` (or other authorization as specified in `device_allocate` file) required to allocate a device  
`solaris.device.revoke` (or other authorization as specified in `device_allocate` file) required to allocate a device to another user (-F option) |
| deallocate(1)        | `solaris.device.allocate` (or other authorization as specified in `device_allocate` file) required to deallocate another user’s device  
`solaris.device.revoke` (or other authorization as specified in `device_allocate`) required to force deallocation of the specified device (-F option) or all devices (-I option) |
| list_devices(1)      | `solaris.device.revoke` required to list another user’s devices (-U option) |
| sendmail(1M)         | `solaris.mail` required to access mail subsystem functions; `solaris.mail.mailq` required to view mail queue |
This chapter provides step-by-step instructions for managing privileges and using privileges on your system. The following is a list of the information in this chapter.

- “Managing and Using Privileges (Task Map)” on page 229
- “Managing Privileges (Task Map)” on page 230
- "Determining Your Privileges (Task Map)” on page 237

For an overview of privileges, see “Privileges (Overview)” on page 178. For reference information, see Chapter 12.

### Managing and Using Privileges (Task Map)

The following task map points to task maps for managing privileges and for using privileges.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use privileges at your site</td>
<td>Involves assigning, removing, adding, and debugging the use of privileges.</td>
<td>“Managing Privileges (Task Map)” on page 230</td>
</tr>
<tr>
<td>Use privileges when you run a command</td>
<td>Involves using the privileges that have been assigned to you.</td>
<td>&quot;Determining Your Privileges (Task Map)” on page 237</td>
</tr>
</tbody>
</table>
Managing Privileges (Task Map)

The following task map points to procedures for viewing privileges, assigning privileges, and running a script that contains privileged commands.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine what privileges are in a process</td>
<td>Lists the effective, inheritable, permitted, and limit privilege sets for a process.</td>
<td>&quot;How to Determine the Privileges on a Process&quot; on page 230</td>
</tr>
<tr>
<td>Determine what privileges are missing from a process</td>
<td>Lists the privileges that a failed process requires to succeed.</td>
<td>&quot;How to Determine Which Privileges a Program Requires&quot; on page 232</td>
</tr>
<tr>
<td>Add privileges to a command</td>
<td>Adds privileges to a command in a rights profile. Users or roles can be assigned the rights profile. The users can then run the command with the assigned privileges in a profile shell.</td>
<td>&quot;How to Add Privileges to a Command&quot; on page 233</td>
</tr>
<tr>
<td>Assign privileges to a user</td>
<td>Expands a user’s or role’s inheritable set of privileges. Use this procedure with caution.</td>
<td>&quot;How to Assign Privileges to a User or Role&quot; on page 234</td>
</tr>
<tr>
<td>Restrict a user’s privileges</td>
<td>Limits the user’s basic set of privileges. Use this procedure with caution.</td>
<td>&quot;How to Limit a User’s or Role’s Privileges&quot; on page 235</td>
</tr>
<tr>
<td>Run a privileged shell script</td>
<td>Adds privilege to a shell script and to the commands in the shell script. Then, runs the script in a profile shell.</td>
<td>&quot;How to Run a Shell Script With Privileged Commands&quot; on page 236</td>
</tr>
</tbody>
</table>

Managing Privileges

The most secure way to manage privileges for users and roles is to confine use of privilege to commands in a rights profile. The rights profile is then included in a role. The role is assigned to a user. When the user assumes the assigned role, the privileged commands are available to be run in a profile shell. The following procedures show how to assign privileges, remove privileges, and debug privilege use.

How to Determine the Privileges on a Process

This procedure shows how to determine which privileges are available to your processes. The listing does not include privileges that have been assigned to particular commands.

- List the privileges that are available to your shell’s process.

  % ppriv pid
  $ ppriv -v pid

  *pid* is the process number. Use a double dollar sign ($$) to pass the process number of the parent shell to the command.
Determining the Privileges in Your Current Shell

In the following example, the privileges in the parent process of the user's shell process are listed. In the second example, the full names of the privileges are listed. The single letters in the output refer to the following privilege sets:

- **E**: Is the effective privilege set.
- **I**: Is the inheritable privilege set.
- **P**: Is the permitted privilege set.
- **L**: Is the limit privilege set.

```sh
% ppriv $$
1200: -csh
flags = <none>
E: basic
I: basic
P: basic
L: all

% ppriv -v $$
1200: -csh
flags = <none>
E: file_link_any,proc_exec,proc_fork,proc_info,proc_session
I: file_link_any,proc_exec,proc_fork,proc_info,proc_session
P: file_link_any,proc_exec,proc_fork,proc_info,proc_session
L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

Determining the Privileges of a Role That You Can Assume

Roles use an administrative shell, or profile shell. You must assume a role and use the role's shell to list the privileges that have been directly assigned to the role. In the following example, the role `sysadmin` has no directly assigned privileges.

```sh
% su sysadmin
Password: <Typesysadminpassword>
$ /usr/ucb/whoami
sysadmin
$ ppriv -v $$
1400: pfksh
flags = <none>
E: file_link_any,proc_exec,proc_fork,proc_info,proc_session
I: file_link_any,proc_exec,proc_fork,proc_info,proc_session
```
How to Determine Which Privileges a Program Requires

This procedure determines which privileges a command or process requires to succeed.

Before You Begin

The command or process must have failed for this procedure to work.

1 Type the command that is failing as an argument to the ppriv debugging command.
   % ppriv -eD touch /etc/acct/yearly
   touch[11365]: missing privilege "file_dac_write"
   (euid = 130, syscall = 224) needed at ufs_direnter_cm+0x27c
   touch: /etc/acct/yearly cannot create

2 Determine which system call is failing by finding the syscall number in the /etc/name_to_sysnum file.
   % grep 224 /etc/name_to_sysnum
   creat64 224

Example 11–3 Using the truss Command to Examine Privilege Use

The truss command can debug privilege use in a regular shell. For example, the following command
   debugs the failing touch process:
   % truss -t creat touch /etc/acct/yearly
   creat64("/etc/acct/yearly", 0666)
   Err#13 EACCES [file_dac_write]
   touch: /etc/acct/yearly cannot create

   The extended /proc interfaces report the missing privilege after the error code in truss output.

Example 11–4 Using the ppriv Command to Examine Privilege Use in a Profile Shell

The ppriv command can debug privilege use in a profile shell. If you assign a rights profile to a user,
and the rights profile includes commands with privileges, the commands must be typed in a profile
shell. When the privileged commands are typed in a regular shell, the commands do not execute with
privilege.

In this example, the jdoe user can assume the role objadmin. The objadmin role includes the Object
Access Management rights profile. This rights profile allows the objadmin role to change
permissions on files that objadmin does not own.

In the following excerpt, jdoe fails to change the permissions on the useful script file:
When jdoe assumes the objadmin role, the permissions on the file are changed:

```bash
jdoe% su objadmin
Password: <Type objadmin password>
$ ls -l useful.script
-rw-r--r-- 1 aloe staff 2303 Mar 11 05:29 useful.script
$ chown objadmin useful.script
$ ls -l useful.script
-rw-r--r-- 1 objadmin staff 2303 Mar 11 05:29 useful.script
$ chgrp admin useful.script
$ ls -l objadmin.script
-rw-r--r-- 1 objadmin admin 2303 Mar 11 05:31 useful.script
```

### Example 11-5 Changing a File Owned by the root User

This example illustrates the protections against privilege escalation. For a discussion, see “Prevention of Privilege Escalation” on page 245. The file is owned by the root user. The less powerful role, objadmin role needs all privileges to change the file’s ownership, so the operation fails.

```bash
jdoe% su objadmin
Password: <Type objadmin password>
$ cd /etc; ls -l system
-rw-r--r-- 1 root sys 1883 Mar 20 14:04 system
$ chown objadmin system
chown: system: Not owner
$ ppriv -ED chown objadmin system
chown[11481]: missing privilege "ALL"
    (euid = 101, syscall = 16) needed at ufs_setattr+0x258
chown: system: Not owner
```

### How to Add Privileges to a Command

You add privileges to a command when you are adding the command to a rights profile. The privileges enable the role that includes the rights profile to run the administrative command, while not gaining any other superuser capabilities.

#### Before You Begin

The command or program must be privilege-aware. For a fuller discussion, see “How Processes Get Privileges” on page 182.
1 Become superuser or assume an equivalent role.
Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.

2 Open the Solaris Management Console GUI.
For instructions, see “How to Assume a Role in the Solaris Management Console” on page 202.

3 Use the Rights tool to update an appropriate profile.
Select the command to include. For each included command, add the privileges that the command requires.

Caution – When you include commands in a rights profile and add privileges to the commands, the commands execute with those privileges when the commands are run in a profile shell.

The order of profiles is important. The profile shell executes a command or action with the security attributes that are specified in the earliest profile in the account’s list of profiles. For example, if the chgrp command is in the Object Access Management rights profile with privileges, and Object Access Management is the first profile in which the chgrp command is found, then the chgrp command executes with the privileges specified in the Object Access Management profile.

How to Assign Privileges to a User or Role
You might trust some users with a particular privilege all the time. Very specific privileges that affect a small part of the system are good candidates for assigning to a user. For a discussion of the implications of directly assigned privileges, see “Security Considerations When Directly Assigning Security Attributes” on page 177.

The following procedure enables user jdoe to use high resolution timers.

1 Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Add the privilege that affects high resolution times to the user’s initial inheritable set of privileges.
$ usermod -K defaultpriv=basic,proc_clock_highres jdoe

3 Read the resulting user_attr entry.
$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
Creating a Role With Privileges to Configure System Time

In this example, a role is created whose only task is to handle time on the system.

```
$ /usr/sadm/bin/smrole -D nisplus:/examplehost/example.domain \
   -r primaryadm -l "<Type primaryadm password>" \
   add -- -n clockmgr \
   -c "Role that sets system time" \
   -F "Clock Manager" \
   -s /bin/pfksh \
   -u 108 \
   -P "<Type clockmgr password>" \
   -K defaultpriv=basic,proc_priocntl,sys_cpu_config,\ 
   proc_clock_highres,sys_time
```

The -K line is wrapped for display purposes.

If the role was created locally, the user_attr entry for the role would appear similar to the following:

```
clockmgr:::Role that sets system time:  
type=role;defaultpriv=basic,proc_priocntl,sys_cpu_config,\ 
proc_clock_highres,sys_time
```

How to Limit a User’s or Role’s Privileges

You can limit the privileges that are available to a user or role by reducing the basic set, or by reducing the limit set. You should have good reason to limit the user’s privileges in this way, because such limitations can have unintended side effects.

Caution – You should thoroughly test any user’s capabilities where the basic set or the limit set has been modified for a user.

- When the basic set is less than the default, users can be prevented from using the system.
- When the limit set is less than all privileges, processes that need to run with an effective UID=0 might fail.

1 Determine the privileges in a user’s basic set and limit set.
For the procedure, see “How to Determine the Privileges on a Process” on page 230.

2 (Optional) Remove one of the privileges from the basic set.
```
$ usermod -K defaultpriv=basic,!priv-name username
```
By removing the proc_session privilege, you prevent the user from examining any processes outside the user’s current session. By removing the file_link_any privilege, you prevent the user from making hard links to files that are not owned by the user.
Caution – Do not remove the `proc_fork` or the `proc_exec` privilege. Without these privileges, the user would not be able to use the system. In fact, these two privileges are only reasonably removed from daemons that should not `fork()` or `exec()` other processes.

3 (Optional) Remove one of the privileges from the limit set.
   $ usermod -K limitpriv=all,!priv-name username

4 Test the capabilities of `username`.
   Log in as `username` and try to perform the tasks that `username` must perform on the system.

Example 11–7 Removing Privileges From a User’s Limit Set

In the following example, all sessions that originate from `jdoe`’s initial login are prevented from using the `sys_linkdir` privilege. That is, the user cannot make hard links to directories, nor can the user unlink directories, even after the user runs the `su` command.

   $ usermod -K limitpriv=all,!sys_linkdir jdoe
   $ grep jdoe /etc/user_attr
   jdoe::::type=normal;defaultpriv=basic;limitpriv=all,!sys_linkdir

Example 11–8 Removing Privileges From a User’s Basic Set

In the following example, all sessions that originate from `jdoe`’s initial login are prevented from using the `proc_session` privilege. That is, the user cannot examine any processes outside the user’s session, even after the user runs the `su` command.

   $ usermod -K defaultpriv=basic,!proc_session jdoe
   $ grep jdoe /etc/user_attr
   jdoe::::type=normal;defaultpriv=basic,!proc_session;limitpriv=all

⚠️ How to Run a Shell Script With Privileged Commands

Note – When you create a shell script that runs commands with inherited privileges, the appropriate rights profile must contain the commands with privileges assigned to them.

1 Start the script with `/bin/pfsh`, or any other profile shell, on the first line.
   #!/bin/pfsh
   # Copyright (c) 2003 by Sun Microsystems, Inc.

2 Determine the privileges that the commands in the script need.
   % ppriv -eD script-full-path
3 **Open the Solaris Management Console GUI.**

   For instructions, see “How to Assume a Role in the Solaris Management Console” on page 202. Choose a role, such as Primary Administrator, that can create a rights profile.

4 **Use the Rights tool to create or update an appropriate profile.**

   Select the script, and include in the rights profile each of the commands in the shell script that need privileges to run. For each included command, add the privileges that the command requires.

   **Caution** – The order of rights profiles is important. The profile shell executes the earliest instance of a command in the list of profiles. For example, if the chgrp command is in the Object Access Management rights profile, and Object Access Management is the first profile in which the chgrp command is found, then the chgrp command executes with the privileges that are specified in the Object Access Management profile.

5 **Add the rights profile to a role and assign the role to a user.**

   To execute the profile, the user assumes the role and runs the script in the role’s profile shell.

---

**Determining Your Privileges (Task Map)**

The following task map points to procedures for using the privileges that have been assigned to you.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View your privileges as a user in any shell</td>
<td>Shows the privileges that have been directly assigned to you. All of your processes run with these privileges.</td>
<td>“How to Determine the Privileges That You Have Been Directly Assigned” on page 238</td>
</tr>
<tr>
<td>Determine which commands you can run with privilege</td>
<td>When privileges are assigned to executables in a rights profile, the executable must be typed in a profile shell.</td>
<td>“How to Determine the Privileged Commands That You Can Run” on page 239</td>
</tr>
<tr>
<td>Determine which commands a role can run with privileges</td>
<td>Assumes the role to determine which commands the role can run with privileges.</td>
<td>“How to Determine the Privileged Commands That a Role Can Run” on page 240</td>
</tr>
</tbody>
</table>

**Determining Your Assigned Privileges**

When a user is directly assigned privileges, the privileges are in effect in every shell. When a user is not directly assigned privileges, then the user must open a profile shell. For example, when commands with assigned privileges are in a rights profile that is in the user’s list of rights profiles, then the user must execute the command in a profile shell.
How to Determine the Privileges That You Have Been Directly Assigned

The following procedure shows how to determine if you have been directly assigned privileges.

Caution – Inappropriate use of directly assigned privileges can result in unintentional breaches of security. For a discussion, see "Security Considerations When Directly Assigning Security Attributes" on page 177.

1. List the privileges that your processes can use.
   See “How to Determine the Privileges on a Process” on page 230 for the procedure.

2. Invoke actions and run commands in any shell.
   The privileges that are listed in the effective set are in effect throughout your session. If you have been directly assigned privileges in addition to the basic set, the privileges are listed in the effective set.

Determining Your Directly-Assigned Privileges

If you have been directly assigned privileges, then your basic set contains more than the default basic set. In this example, the user always has access to the proc_clock_highres privilege.

```
% /usr/ucb/whoami
jdoe
% ppriv -v $$
1800: pfksh
flags = <none>
E: file_link.any,...,proc_clock_highres,proc_session
I: file_link.any,...,proc_clock_highres,proc_session
P: file_link.any,...,proc_clock_highres,proc_session
L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
% ppriv -vl proc_clock_highres
Allows a process to use high resolution timers.
```

Determining a Role's Directly-Assigned Privileges

Roles use an administrative shell, or profile shell. Users who assume a role can use the role's shell to list the privileges that have been directly assigned to the role. In the following example, the role realtime has been directly assigned privileges to handle date and time programs.

```
% su realtime
Password: <Type realtime password>
$ /usr/ucb/whoami
realtime
$ ppriv -v $$
```
How to Determine the Privileged Commands That You Can Run

When a user is not directly assigned privileges, then the user gets access to privileged commands through a rights profile. Commands in a rights profile must be executed in a profile shell.

Before You Begin

The user or role who authenticates to the Solaris Management Console must have the `solaris.admin.usermgr.read` authorization. The Basic Solaris User rights profile includes this authorization.

1 Determine the rights profiles that you have been assigned.

$ /usr/sadm/bin/smuser list -- -n username -l
Authenticating as user: admin
... Please enter a string value for: password ::
...
User name: username
User ID (UID): 130
Primary group: staff
Secondary groups:
  Comment: object mgt jobs
Login Shell: /bin/sh
Home dir server: system
Home directory: /export/home/username
AutoHome setup: True
Mail server: system
Rights: Object Access Management
Assigned Roles:

2 Locate the line that begins with “Rights:”.

The “Rights” line lists the names of the rights profiles that have been directly assigned to you.

3 Find the names of the rights profiles in the `exec_attr` database.

$ cd /etc/security
$ grep "Object Access Management" exec_attr
Object Access Management:solaris:cmd:::/usr/bin/chgrp:privil=file_chown
Object Access Management:solaris:cmd:::/usr/bin/chown:privil=file_chown
Object Access Management:suser:cmd:::/usr/bin/chgrp:euid=0
4 Type the commands that require privileges in a profile shell.

When the commands are typed in a regular shell, the commands do not run with privilege, and do not succeed.

% pfsh
$

Example 11–11 Running Privileged Commands in a Profile Shell

In the following example, the user jdoe cannot change the group permissions on a file from his regular shell. However, jdoe can change the permissions when typing the command in a profile shell.

% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 1 nodeo eng 262 Apr 2 10:52 useful.script
chgrp staff useful.script
chgrp: useful.script: Not owner
% pfsh
$ /usr/ucb/whoami
jdoe
$ chgrp staff useful.script
$ chown jdoe useful.script
$ ls -l useful.script
-rwxr-xr-- 1 jdoe staff 262 Apr 2 10:53 useful.script

How to Determine the Privileged Commands That a Role Can Run

A role gets access to privileged commands through a rights profile that contains commands with assigned privileges. The most secure way to provide a user with access to privileged commands is to assign a role to them. After assuming the role, the user can execute all the privileged commands that are included in the rights profiles for that role.

Before You Begin

The user or role who authenticates to the Solaris Management Console must have the solaris.admin.usermgr.read authorization. The Basic Solaris User rights profile includes this authorization.

1 Determine the roles that you can assume.

$ /usr/sadm/bin/smuser list -- -n username -l
Authenticate as user: primadmin
...
User name: username
User ID (UID): 110
Primary group: staff
Secondary groups:
Comment: Has admin roles
Login Shell: /bin/sh
...
Rights:
Assigned Roles: primadmin, admin

2 Locate the line that begins with “Assigned Roles:”. The “Assigned Roles” line lists the roles that you can assume.

3 Determine the rights profiles that are included in one of your roles.
   $ /usr/sadm/bin/smuser list -- -n admin -l
   Authenticating as user: primadmin
   ...
   User name: admin
   User ID (UID): 101
   Primary group: sysadmin
   Secondary groups:
   Comment: system administrator
   Login Shell: /bin/pfksh
   ...
   Rights: System Administrator
   Assigned Roles:

4 Locate the names of the rights profiles for the role in the “Rights:” line.

5 Find the rights profiles in the prof_attr database.
   Because the System Administrator profile is a collection of profiles, you need to list the profiles in the System Administrator profile.
   $ cd /etc/security
   $ grep "System Administrator" prof_attr
For each rights profile, find the rights profiles in the exec_attr database.

For example, the Network Management profile is a supplementary profile of the System Administrator profile. The Network Management profile includes a number of privileged commands.

```shell
$ cd /etc/security
$ grep "Network Management" exec_attr
Network Management:solaris:cmd:::/usr/sbin/ifconfig:privs=sys_net_config
Network Management:solaris:cmd:::/usr/sbin/route:privs=sys_net_config
...```

The commands and their assigned privileges are the final two fields of solaris policy entries. You can run these commands in the profile shell of your role.

Example 11–12 Running the Privileged Commands in Your Role

When a user assumes a role, the shell becomes a profile shell. Therefore, the commands are executed with the privileges that were assigned to the commands. In the following example, the admin role can change the permissions on the useful.script file.

```bash
% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 1 elsee eng 262 Apr 2 10:52 useful.script
chgrp admin useful.script
chgrp: useful.script: Not owner
% su admin
Password: <Type admin password>
$ /usr/ucb/whoami
admin
$ chgrp admin useful.script
$ chown admin useful.script
$ ls -l useful.script
-rwxr-xr-- 1 admin admin 262 Apr 2 10:53 useful.script
```
Privileges (Reference)

The following is a list of the reference information in this chapter:

- “Administrative Commands for Handling Privileges” on page 243
- “Files With Privilege Information” on page 244
- “Privileges and Auditing” on page 245
- “Prevention of Privilege Escalation” on page 245
- “Legacy Applications and the Privilege Model” on page 246

To use privileges, see Chapter 11. For overview information, see “Privileges (Overview)” on page 178.

Administrative Commands for Handling Privileges

The following table lists the commands that are available to handle privileges.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Command</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examine process privileges</td>
<td><code>ppriv -v pid</code></td>
<td><code>ppriv(1)</code></td>
</tr>
<tr>
<td>Set process privileges</td>
<td><code>ppriv -s spec</code></td>
<td></td>
</tr>
<tr>
<td>List the privileges on the system</td>
<td><code>ppriv -l</code></td>
<td></td>
</tr>
<tr>
<td>List a privilege and its description</td>
<td><code>ppriv -lv priv</code></td>
<td></td>
</tr>
<tr>
<td>Debug privilege failure</td>
<td><code>ppriv -eD failed-operation</code></td>
<td></td>
</tr>
<tr>
<td>Assign privileges to a new local user</td>
<td><code>useradd</code></td>
<td><code>useradd(1M)</code></td>
</tr>
<tr>
<td>Add privileges to an existing local user</td>
<td><code>usermod</code></td>
<td><code>usermod(1M)</code></td>
</tr>
<tr>
<td>Assign privileges to a user in a name service</td>
<td><code>smuser</code></td>
<td><code>smuser(1M)</code></td>
</tr>
</tbody>
</table>
### TABLE 12–1 Commands for Handling Privilege (Continued)

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Command</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign privileges to a new local role</td>
<td>roleadd</td>
<td>roleadd(1M)</td>
</tr>
<tr>
<td>Add privileges to an existing local role</td>
<td>rolemod</td>
<td>rolemod(1M)</td>
</tr>
<tr>
<td>Assign privileges to a role in a name service</td>
<td>smrole</td>
<td>smrole(1M)</td>
</tr>
<tr>
<td>View device policy</td>
<td>getdevpolicy</td>
<td>getdevpolicy(1M)</td>
</tr>
<tr>
<td>Set device policy</td>
<td>devfsadm</td>
<td>devfsadm(1M)</td>
</tr>
<tr>
<td>Update device policy on open devices</td>
<td>update_drv -p policy driver</td>
<td>update_drv(1M)</td>
</tr>
<tr>
<td>Add device policy to a device</td>
<td>add_drv -p policy driver</td>
<td>add_drv(1M)</td>
</tr>
</tbody>
</table>

The Solaris Management Console GUI is the preferred tool for assigning privileges to commands, users, and roles. For more information, see “How to Assume a Role in the Solaris Management Console” on page 202.

### Files With Privilege Information

The following files contain information about privileges.

### TABLE 12-2 Files That Contain Privilege Information

<table>
<thead>
<tr>
<th>File and Man Page</th>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/security/policy.conf</td>
<td>PRIV_DEFAULT</td>
<td>Inheritable set of privileges for the system</td>
</tr>
<tr>
<td></td>
<td>PRIV_LIMIT</td>
<td>Limit set of privileges for the system</td>
</tr>
<tr>
<td>/etc/user_attr</td>
<td>defaultpriv keyword in user or role's entry</td>
<td>Inheritable set of privileges for a user or role</td>
</tr>
<tr>
<td></td>
<td>limitpriv keyword in user or role's entry</td>
<td>Limit set of privileges for a user or role</td>
</tr>
<tr>
<td>/etc/security/exec_attr</td>
<td>prims keyword in the profile's entry for the command</td>
<td>List of privileges that are assigned to a command in a rights profile</td>
</tr>
<tr>
<td></td>
<td>Policy for the command must be solaris</td>
<td></td>
</tr>
</tbody>
</table>
Privileges and Auditing

Privilege use can be audited. When a process uses a privilege, the use of privilege is recorded in the audit trail. The privileges are recorded in their textual representation. The following audit events record use of privilege:

- **AUE_SETPPRIV audit event** – The event generates an audit record when a privilege set is changed. The AUE_SETPPRIV audit event is in the pm class.
- **AUE_MODALLOCPRIV audit event** – The audit event generates an audit record when a privilege is added from outside the kernel. The AUE_MODALLOCPRIV audit event is in the ad class.
- **AUE_MODEVPLCY audit event** – The audit event generates an audit record when the device policy is changed. The AUE_MODEVPLCY audit event is in the ad class.
- **AUE_prof_cmd audit event** – The audit event generates an audit record when a command is executed in a profile shell. The AUE_prof_cmd audit event is in the as and ua audit classes.

The successful use of privileges that are in the basic set is not audited. The attempt to use a basic privilege that has been removed from a user’s basic set is audited.

Prevention of Privilege Escalation

The Solaris kernel prevents privilege escalation. Privilege escalation is when a privilege enables a process to do more than the process should be able to do. To prevent a process from gaining more privileges than the process should have, certain system modifications require the full set of privileges. For example, a file or process that is owned by root (UID=0) can only be changed by a process with the full set of privileges. The root user does not require privileges to change a file that root owns. However, a non-root user must have all privileges in order to change a file that is owned by root.

Similarly, operations that provide access to devices require all privileges in the effective set.

The `file_chown_self` and `proc_owner` privileges are subject to privilege escalation. The `file_chown_self` privilege allows a process to give away its files. The `proc_owner` privilege allows a process to inspect processes that the process does not own.

---

### TABLE 12-2 Files That Contain Privilege Information (Continued)

<table>
<thead>
<tr>
<th>File and Man Page</th>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>syslog.conf</td>
<td>System log file for debug messages</td>
<td>Privilege debugging log</td>
</tr>
<tr>
<td>syslog.conf(4)</td>
<td>Path set in priv.debug entry</td>
<td></td>
</tr>
</tbody>
</table>

**Note** – Do not edit the `exec_attr` and `user_attr` databases directly. To administer privileges, use the Solaris Management Console, or commands such as `smuser`. For more information, see the `snc(1M)` and the `smuser(1M)` man pages. For procedures, see "Managing Privileges (Task Map)" on page 230.
The `file_chown_self` privilege is limited by the `rstchown` system variable. When the `rstchown` variable is set to zero, the `file_chown_self` privilege is removed from the initial inheritable set of the system and of all users. For more information on the `rstchown` system variable, see the `chown(1)` man page.

The `file_chown_self` privilege is most safely assigned to a particular command, placed in a profile, and assigned to a role for use in a profile shell.

The `proc_owner` privilege is not sufficient to switch a process UID to 0. To switch a process from any UID to UID=0 requires all privileges. Because the `proc_owner` privilege gives unrestricted read access to all files on the system, the privilege is most safely assigned to a particular command, placed in a profile, and assigned to a role for use in a profile shell.

---

**Caution** – A user’s account can be modified to include the `file_chown_self` privilege or the `proc_owner` privilege in the user’s initial inheritable set. You should have overriding security reasons for placing such powerful privileges in the inheritable set of privileges for any user, role, or system.

For details of how privilege escalation is prevented for devices, see “Privileges and Devices” on page 184.

---

**Legacy Applications and the Privilege Model**

To accommodate legacy applications, the implementation of privileges works with both the superuser and the privilege models. The kernel automatically tracks the `PRIV_AWARE` flag, which indicates that a program has been designed to work with privileges. Consider a child process that is not aware of privileges. Any privileges that were inherited from the parent process are available in the child’s permitted and effective sets. If the child process sets a UID to 0, the child process might not have full superuser capabilities. The process’s effective and permitted sets are restricted to those privileges in the child’s limit set. Thus, the limit set of a privilege-aware process restricts the root privileges of child processes that are not aware of privileges.
Solaris Cryptographic Services

This section describes the centralized cryptographic services that the Solaris OS provides.
This chapter describes the Solaris™ Cryptographic Framework. The following is a list of the information in this chapter.

- “What’s New in the Solaris Cryptographic Framework?” on page 249
- “Solaris Cryptographic Framework” on page 250
- ”Terminology in the Solaris Cryptographic Framework” on page 250
- “Scope of the Solaris Cryptographic Framework” on page 252
- “Administrative Commands in the Solaris Cryptographic Framework” on page 252
- “User-Level Commands in the Solaris Cryptographic Framework” on page 253
- “Plugins to the Solaris Cryptographic Framework” on page 253
- “Cryptographic Services and Zones” on page 254

To administer and use the Solaris Cryptographic Framework, see Chapter 14.

### What’s New in the Solaris Cryptographic Framework?

**Solaris 10 1/06**: The framework library, `/libpkcs11.so`, contains a new component, the metaslot. The metaslot serves as a single virtual slot with the combined capabilities of all tokens and slots that have been installed in the framework. Effectively, the metaslot enables an application to transparently connect with any available cryptographic service through a single slot.

- For more information, see the definitions of slot, metaslot, and token in “Terminology in the Solaris Cryptographic Framework” on page 250.
- To administer the metaslot, see the `cryptoadm(1M)` man page.
- For a complete listing of new Solaris features and a description of Solaris releases, see Solaris 10 What’s New.
The Solaris Cryptographic Framework provides a common store of algorithms and PKCS #11 libraries to handle cryptographic requirements. The PKCS #11 libraries are implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki).

At the kernel level, the framework currently handles cryptographic requirements for Kerberos and IPsec. User-level consumers include \libasl and IKE.

Export law in the United States requires that the use of open cryptographic interfaces be restricted. The Solaris Cryptographic Framework satisfies the current law by requiring that kernel cryptographic providers and PKCS #11 cryptographic providers be signed. For further discussion, see “Binary Signatures for Third-Party Software” on page 253.

The framework enables providers of cryptographic services to have their services used by many consumers in the Solaris Operating System. Another name for providers is plugins. The framework allows three types of plugins:

- **User-level plugins** – Shared objects that provide services by using PKCS #11 libraries, such as pkcs11_softtoken.so.1.

- **Kernel-level plugins** – Kernel modules that provide implementations of cryptographic algorithms in software, such as AES.
  
  Many of the algorithms in the framework are optimized for x86 with the SSE2 instruction set and for SPARC hardware.

- **Hardware plugins** – Device drivers and their associated hardware accelerators. A hardware accelerator offloads expensive cryptographic functions from the operating system. The Sun Crypto Accelerator 1000 board is one example.

The framework implements a standard interface, the PKCS #11, v2.11 library, for user-level providers. The library can be used by third-party applications to reach providers. Third parties can also add signed libraries, signed kernel algorithm modules, and signed device drivers to the framework. These plugins are added when the pkgadd utility installs the third-party software. For a diagram of the major components of the framework, see Chapter 8, “Introduction to the Solaris Cryptographic Framework,” in Solaris Security for Developers Guide.

**Terminology in the Solaris Cryptographic Framework**

The following list of definitions and examples is useful when working with the cryptographic framework.

- **Algorithms** – Cryptographic algorithms. These are established, recursive computational procedures that encrypt or hash input. Encryption algorithms can be symmetric or asymmetric. Symmetric algorithms use the same key for encryption and decryption. Asymmetric algorithms, which are used in public-key cryptography, require two keys. Hashing functions are also algorithms.
Examples of algorithms include:
- Symmetric algorithms, such as AES and ARCFOUR
- Asymmetric algorithms, such as Diffie-Hellman and RSA
- Hashing functions, such as MD5

**Consumers** – Are users of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations.

Examples of consumers include:
- Applications, such as IKE
- End users, such as an ordinary user who runs the `encrypt` command
- Kernel operations, such as IPsec

**Mechanism** – Is the application of a mode of an algorithm for a particular purpose.

For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.

**Metaslot** – Is a single slot that presents a union of the capabilities of other slots which are loaded in the framework. The metaslot eases the work of dealing with all of the capabilities of the providers that are available through the framework. When an application that uses the metaslot requests an operation, the metaslot figures out which actual slot should perform the operation. Metaslot capabilities are configurable, but configuration is not required. The metaslot is on by default. To configure the metaslot, see the `cryptoadm(1M)` man page.

**Mode** – Is a version of a cryptographic algorithm. For example, CBC (Cipher Block Chaining) is a different mode from ECB (Electronic Code Book). The AES algorithm has two modes, CKM_AES_CBC and CKM_AES_CBC.

**Policy** – Is the choice, by an administrator, of which mechanisms to make available for use. By default, all providers and all mechanisms are available for use. The disabling of any mechanism would be an application of policy. The enabling of a disabled mechanism would also be an application of policy.

**Providers** – Are cryptographic services that consumers use. Providers plug in to the framework, so are also called *plugins*.

Examples of providers include:
- PKCS #11 libraries, such as `pkcs11_softtoken.so`
- Modules of cryptographic algorithms, such as `aes` and `arcfour`
- Device drivers and their associated hardware accelerators, such as the `dca/0` accelerator

**Slot** – Is an interface to one or more cryptographic devices. Each slot, which corresponds to a physical reader or other device interface, might contain a token. A token provides a logical view of a cryptographic device in the framework.

**Token** – In a slot, a token provides a logical view of a cryptographic device in the framework.
Scope of the Solaris Cryptographic Framework

The framework provides commands for administrators, for users, and for developers who supply providers:

- **Administrative commands** – The `cryptoadm` command provides a `list` subcommand to list the available providers and their capabilities. Ordinary users can run the `cryptoadm list` and the `cryptoadm --help` commands.

  All other `cryptoadm` subcommands require you to assume a role that includes the Crypto Management rights profile, or to become superuser. Subcommands such as `disable`, `install`, and `uninstall` are available for administrating the framework. For more information, see the `cryptoadm(1M)` man page.

  The `svcadm` command is used to manage the `kcfd` daemon, and to refresh cryptographic policy in the kernel. For more information, see the `svcadm(1M)` man page.

- **User-level commands** – The `digest` and `mac` commands provide file integrity services. The `encrypt` and `decrypt` commands protect files from eavesdropping. To use these commands, see “Protecting Files With the Solaris Cryptographic Framework (Task Map)” on page 256.

- **Binary signatures for third-party providers** – The `elfsign` command enables third parties to sign binaries for use within the framework. Binaries that can be added to the framework are PKCS#11 libraries, kernel algorithm modules, and hardware device drivers. To use the `elfsign` command, see Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

Administrative Commands in the Solaris Cryptographic Framework

The `cryptoadm` command administers a running cryptographic framework. The command is part of the Crypto Management rights profile. This profile can be assigned to a role for secure administration of the cryptographic framework. The `cryptoadm` command manages the following:

- Displaying cryptographic provider information
- Disabling or enabling provider mechanisms
- Solaris 10 1/06: Disabling or enabling the metaslot

The `svcadm` command is used to enable, refresh, and disable the cryptographic services daemon, `kcfd`. This command is part of the Solaris service management facility, `smf`.

`srv:/system/cryptosvcs` is the service instance for the cryptographic framework. For more information, see the `smf(5)` and `svcadm(1M)` man pages.
User-Level Commands in the Solaris Cryptographic Framework

The Solaris Cryptographic Framework provides user-level commands to check the integrity of files, to encrypt files, and to decrypt files. A separate command, elfsign, enables providers to sign binaries for use with the framework.

- **digest command** – Computes a message digest for one or more files or for stdin. A digest is useful for verifying the integrity of a file. SHA1 and MD5 are examples of digest functions.

- **mac command** – Computes a message authentication code (MAC) for one or more files or for stdin. A MAC associates data with an authenticated message. A MAC enables a receiver to verify that the message came from the sender and that the message has not been tampered with. The sha1_mac and md5_hmac mechanisms can compute a MAC.

- **encrypt command** – Encrypts files or stdin with a symmetric cipher. The encrypt -l command lists the algorithms that are available. Mechanisms that are listed under a user-level library are available to the encrypt command. The framework provides AES, DES, 3DES (Triple-DES), and ARCFOUR mechanisms for user encryption.

- **decrypt command** – Decrypts files or stdin that were encrypted with the encrypt command. The decrypt command uses the identical key and mechanism that were used to encrypt the original file.

Binary Signatures for Third-Party Software

The elfsign command provides a means to sign providers to be used with the Solaris Cryptographic Framework. Typically, this command is run by the developer of a provider.

The elfsign command has subcommands to request a certificate from Sun and to sign binaries. Another subcommand verifies the signature. Unsigned binaries cannot be used by the Solaris Cryptographic Framework. To sign one or more providers requires the certificate from Sun and the private key that was used to request the certificate. For more information, see Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

Plugins to the Solaris Cryptographic Framework

Third parties can plug their providers into the Solaris Cryptographic Framework. A third-party provider can be one of the following objects:

- PKCS #11 shared library
- Loadable kernel software module, such as an encryption algorithm, MAC function, or digest function
- Kernel device driver for a hardware accelerator
The objects from a provider must be signed with a certificate from Sun. The certificate request is based on a private key that the third party selects, and a certificate that Sun provides. The certificate request is sent to Sun, which registers the third party and then issues the certificate. The third party then signs its provider object with the certificate from Sun.

The loadable kernel software modules and the kernel device drivers for hardware accelerators must also register with the kernel. Registration is through the Solaris Cryptographic Framework SPI (service provider interface).

To install the provider, the third party provides a package that installs the signed object and the certificate from Sun. The package must include the certificate, and enable the administrator to place the certificate in a secure directory. For more information, see the Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

**Cryptographic Services and Zones**

The global zone and each non-global zone has its own /system/cryptosvc service. When the cryptographic service is enabled or refreshed in the global zone, the kcfd daemon starts in the global zone, user-level policy for the global zone is set, and kernel policy for the system is set. When the service is enabled or refreshed in a non-global zone, the kcfd daemon starts in the zone, and user-level policy for the zone is set. Kernel policy was set by the global zone.

For more information on zones, see Part II, “Zones,” in System Administration Guide: Solaris Containers-Resource Management and Solaris Zones. For more information on the service management facility that manages persistent applications, see Chapter 14, “Managing Services (Overview),” in System Administration Guide: Basic Administration and the smf(5) man page.
This chapter describes how to use the Solaris Cryptographic Framework. The following is a list of information in this chapter.

- “Using the Cryptographic Framework (Task Map)” on page 255
- “Protecting Files With the Solaris Cryptographic Framework (Task Map)” on page 256
- “Administering the Cryptographic Framework (Task Map)” on page 263

### Using the Cryptographic Framework (Task Map)

The following task map points to tasks for using the cryptographic framework.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protect individual files or sets of files</td>
<td>Ensures that file content has not been tampered with. Prevents files from being read by intruders. These procedures can be done by ordinary users.</td>
<td>“Protecting Files With the Solaris Cryptographic Framework (Task Map)” on page 256</td>
</tr>
<tr>
<td>Administer the framework</td>
<td>Adds, configures, and removes software providers. Disables and enables hardware provider mechanisms. These procedures are administrative procedures.</td>
<td>“Administering the Cryptographic Framework (Task Map)” on page 263</td>
</tr>
<tr>
<td>Sign a provider</td>
<td>Enables a provider to be added to the Solaris Cryptographic Framework. These procedures are developer procedures.</td>
<td>Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.</td>
</tr>
</tbody>
</table>
Protecting Files With the Solaris Cryptographic Framework (Task Map)

The Solaris Cryptographic Framework can help you protect your files. The following task map points to procedures for listing the available algorithms, and for protecting your files cryptographically.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate a symmetric key</td>
<td>Generates a random key for use with the encrypt command or the mac command.</td>
<td>“How to Generate a Symmetric Key” on page 256</td>
</tr>
<tr>
<td>Provide a checksum that ensures the integrity of a file</td>
<td>Verifies that the receiver’s copy of a file is identical to the file that was sent.</td>
<td>“How to Compute a Digest of a File” on page 258</td>
</tr>
<tr>
<td>Protect a file with a message authentication code (MAC)</td>
<td>Verifies to the receiver of your message that you were the sender.</td>
<td>“How to Compute a MAC of a File” on page 259</td>
</tr>
<tr>
<td>Encrypt a file, and then decrypt the encrypted file</td>
<td>Protects the content of files by encrypting the file. Provides the encryption parameters to decrypt the file.</td>
<td>“How to Encrypt and Decrypt a File” on page 261</td>
</tr>
</tbody>
</table>

Protecting Files With the Solaris Cryptographic Framework

This section describes how to generate symmetric keys, how to create checksums for file integrity, and how to protect files from eavesdropping. The commands in this section can be run by ordinary users. Developers can write scripts that use these commands.

▼ How to Generate a Symmetric Key

A key is needed to encrypt files, and to generate the MAC of a file. The key should be derived from a random pool of numbers.

If your site has a random number generator, use the generator. Otherwise, you can use the dd command with the Solaris /dev/urandom device as input. For more information, see the dd(1M) man page.

1. Determine the key length that your algorithm requires.

   a. List the available algorithms.

   ```
   % encrypt -l
   Algorithm  Keysize:  Min  Max (bits)
   --------------------------------------
   aes        128  128
   arcfour    8   128
   des        64   64
   ```
b. **Determine the key length in bytes to pass to the dd command.**

Divide the minimum and maximum key sizes by 8. When the minimum and maximum key sizes are different, intermediate key sizes are possible. For example, the value 8, 16, or 64 can be passed to the dd command for the sha1_hmac and md5_hmac functions.

2 **Generate the symmetric key.**

```bash
% dd if=/dev/urandom of=keyfile bs=n count=n
```

- `if=file` is the input file. For a random key, use the `/dev/urandom` file.
- `of=keyfile` is the output file that holds the generated key.
- `bs=n` is the key size in bytes. For the length in bytes, divide the key length in bits by 8.
- `count=n` is the count of the input blocks. The number for `n` should be 1.

3 **Store your key in a protected directory.**

The key file should not be readable by anyone but the user.

```bash
% chmod 400 keyfile
```

**Example 14–1 Creating a Key for the AES Algorithm**

In the following example, a secret key for the AES algorithm is created. The key is also stored for later decryption. AES mechanisms use a 128-bit key. The key is expressed as 16 bytes in the `dd` command.

```bash
% ls -a ~/
```

```bash
% dd if=/dev/urandom of=$HOME/keyf/05.07.aes16 bs=16 count=1
```

```bash
% chmod 400 $HOME/keyf/05.07.aes16
```

**Example 14–2 Creating a Key for the DES Algorithm**

In the following example, a secret key for the DES algorithm is created. The key is also stored for later decryption. DES mechanisms use a 64-bit key. The key is expressed as 8 bytes in the `dd` command.

```bash
% dd if=/dev/urandom of=$HOME/keyf/05.07.des8 bs=8 count=1
```
Example 14–3  Creating a Key for the 3DES Algorithm

In the following example, a secret key for the 3DES algorithm is created. The key is also stored for later decryption. 3DES mechanisms use a 192-bit key. The key is expressed as 24 bytes in the dd command.

\%
\texttt{dd if=/dev/urandom of=$HOME/keyf/05.07.3des.24 bs=24 count=1}
\%
\texttt{chmod 400 ~/keyf/05.07.3des.24}

Example 14–4  Creating a Key for the MD5 Algorithm

In the following example, a secret key for the MD5 algorithm is created. The key is also stored for later decryption. The key is expressed as 64 bytes in the dd command.

\%
\texttt{dd if=/dev/urandom of=$HOME/keyf/05.07.mack64 bs=64 count=1}
\%
\texttt{chmod 400 ~/keyf/05.07.mack64}

\section*{How to Compute a Digest of a File}

When you compute a digest of a file, you can check to see that the file has not been tampered with by comparing digest outputs. A digest does not alter the original file.

1  List the available digest algorithms.

\%
\texttt{digest -l}
md5
sha1

2  Compute the digest of the file and save the digest listing.

Provide an algorithm with the \texttt{digest} command.

\%
\texttt{digest -v -a \textit{algorithm} input-file > digest-listing}

- \texttt{-v} Displays the output in the following format:

\textit{algorithm} (input-file) = digest

- \texttt{-a \textit{algorithm}} Is the algorithm to use to compute a digest of the file. Type the algorithm as the algorithm appears in the output of Step 1.

\textit{input-file} Is the input file for the \texttt{digest} command.

\textit{digest-listing} Is the output file for the \texttt{digest} command.

Example 14–5  Computing a Digest With the MD5 Mechanism

In the following example, the \texttt{digest} command uses the MD5 mechanism to compute a digest for an email attachment.
When the `-v` option is not used, the digest is saved with no accompanying information:

```bash
% digest -a md5 email.attach >> $HOME/digest.emails.05.07
% cat ~/digest.emails.05.07
85c0a53d1a5cc71ea34d9ee7b1b28b01
```

### Example 14–6  Computing a Digest With the SHA1 Mechanism

In the following example, the digest command uses the SHA1 mechanism to provide a directory listing. The results are placed in a file.

```bash
% digest -v -a sha1 docs/* > $HOME/digest.docs.legal.05.07
% more ~/digest.docs.legal.05.07
sha1 (docs/legal1) = 1df50e8ad219e34f0b911e097b7b588e31f9b435
sha1 (docs/legal2) = 68efa5a636291bde8f33e046eb33508c94842c38
sha1 (docs/legal3) = 085d991238d61bd0ca2946c183be8e32cccf6c9
sha1 (docs/legal4) = f3005eae7e2c8d008816564f82027d1001d983
```

## How to Compute a MAC of a File

A message authentication code, or MAC, computes a digest for the file and uses a secret key to further protect the digest. A MAC does not alter the original file.

### 1 List the available mechanisms.

```bash
% mac -l
Algorithm     Keysize:  Min  Max
---------------  --------  -------
des_mac        64       64
sha1_hmac      8        512
md5_hmac       8        512
```

### 2 Generate a symmetric key of the appropriate length.

You have two options. You can provide a passphrase from which a key will be generated. Or you can provide a key.

- If you provide a passphrase, you must store or remember the passphrase. If you store the passphrase online, the passphrase file should be readable only by you.
- If you provide a key, it must be the correct size for the mechanism. For the procedure, see "How to Generate a Symmetric Key" on page 256.
3 Create a MAC for a file.

Provide a key and use a symmetric key algorithm with the mac command.

% mac -v -a algorithm [-k keyfile] input-file

- `v` Displays the output in the following format:

```
algorithm (input-file) = mac
```

- `algorithm` Is the algorithm to use to compute the MAC. Type the algorithm as the algorithm appears in the output of the `mac -l` command.

- `k keyfile` Is the file that contains a key of algorithm-specified length.

`input-file` Is the input file for the MAC.

Example 14–7 Computing a MAC With DES_MAC and a Passphrase

In the following example, the email attachment is authenticated with the DES_MAC mechanism and a key that is derived from a passphrase. The MAC listing is saved to a file. If the passphrase is stored in a file, the file should not be readable by anyone but the user.

```
% mac -v -a des_mac email.attach
Enter key: <Type passphrase>
des_mac (email.attach) = dd27870a
% echo "des_mac (email.attach) = dd27870a" >> ~/desmac.daily.05.07
```

Example 14–8 Computing a MAC With MD5_HMAC and a Key File

In the following example, the email attachment is authenticated with the MD5_HMAC mechanism and a secret key. The MAC listing is saved to a file.

```
% mac -v -a md5_hmac -k $HOME/keyf/05.07.mack64 email.attach
md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c
% echo "md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c" >> ~/mac.daily.05.07
```

Example 14–9 Computing a MAC With SHA1_HMAC and a Key File

In the following example, the directory manifest is authenticated with the SHA1_HMAC mechanism and a secret key. The results are placed in a file.

```
% mac -v -a sha1_hmac -k $HOME/keyf/05.07.mack64 docs/* > $HOME/mac.docs.legal.05.07
% more ~/mac.docs.legal.05.07
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cd1a8c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aeb3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```
**How to Encrypt and Decrypt a File**

When you encrypt a file, the original file is not removed or changed. The output file is encrypted.

For solutions to common errors from the `encrypt` command, see the section that follows the examples.

1. **Create a symmetric key of the appropriate length.**
   You have two options. You can provide a passphrase from which a key will be generated. Or you can provide a key.
   - If you provide a passphrase, you must store or remember the passphrase. If you store the passphrase online, the passphrase file should be readable only by you.
   - If you provide a key, it must be the correct size for the mechanism. For the procedure, see "How to Generate a Symmetric Key" on page 256.

2. **Encrypt a file.**
   Provide a key and use a symmetric key algorithm with the `encrypt` command.
   ```
   % encrypt -a algorithm [ -k keyfile ] -i input-file -o output-file
   
   -a algorithm Is the algorithm to use to encrypt the file. Type the algorithm as the algorithm appears in the output of the `encrypt -l` command.
   
   -k keyfile Is the file that contains a key of algorithm-specified length. The key length for each algorithm is listed, in bits, in the output of the `encrypt -l` command.
   
   -i input-file Is the input file that you want to encrypt. This file is left unchanged by the command.
   
   -o output-file Is the output file that is the encrypted form of the input file.
   
   **Example 14–10** Encrypting and Decrypting With AES and a Passphrase
   In the following example, a file is encrypted with the AES algorithm. The key is generated from the passphrase. If the passphrase is stored in a file, the file should not be readable by anyone but the user.
   ```
   % encrypt -a aes -i ticket.to.ride -o ~/enc/e.ticket.to.ride
   Enter key: <Type passphrase>
   
   The input file, ticket.to.ride, still exists in its original form.
   
   To decrypt the output file, the user uses the same passphrase and encryption mechanism that encrypted the file.
   ```
   % decrypt -a aes -i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
   Enter key: <Type passphrase>
Encrypting and Decrypting With AES and a Key File

In the following example, a file is encrypted with the AES algorithm. AES mechanisms use a key of 128 bits, or 16 bytes.

```
% encrypt -a aes -k ~/keyf/05.07.aes16 \
- i ticket.to.ride -o ~/enc/e.ticket.to.ride
```

The input file, ticket.to.ride, still exists in its original form.

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a aes -k ~/keyf/05.07.aes16 \
- i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
```

Encrypting and Decrypting With ARCFOUR and a Key File

In the following example, a file is encrypted with the ARCFOUR algorithm. The ARCFOUR algorithm accepts a key of 8 bits (1 byte), 64 bits (8 bytes), or 128 bits (16 bytes).

```
% encrypt -a arcfour -i personal.txt \
- k ~/keyf/05.07.rc4.8 -o ~/enc/e.personal.txt
```

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a arcfour -i ~/enc/e.personal.txt \
- k ~/keyf/05.07.rc4.8 -o ~/personal.txt
```

Encrypting and Decrypting With 3DES and a Key File

In the following example, a file is encrypted with the 3DES algorithm. The 3DES algorithm requires a key of 192 bits, or 24 bytes.

```
% encrypt -a 3des -k ~/keyf/05.07.des24 \
- i ~/personal2.txt -o ~/enc/e.personal2.txt
```

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a 3des -k ~/keyf/05.07.des24 \
- i ~/enc/e.personal2.txt -o ~/personal2.txt
```

Troubleshooting

The following messages indicate that the key that you provided to the encrypt command is not permitted by the algorithm that you are using.

- encrypt: unable to create key for crypto operation: CKR_ATTRIBUTE_VALUE_INVALID
- encrypt: failed to initialize crypto operation: CKR_KEY_SIZE_RANGE
If you pass a key that does not meet the requirements of the algorithm, you must supply a better key.

- One option is to use a passphrase. The framework then provides a key that meets the requirements.
- The second option is to pass a key size that the algorithm accepts. For example, the DES algorithm requires a key of 64 bits. The 3DES algorithm requires a key of 192 bits.

**Administering the Cryptographic Framework (Task Map)**

The following task map points to procedures for administering software and hardware providers in the Solaris Cryptographic Framework.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List the providers in the Solaris</td>
<td>Lists the algorithms, libraries, and hardware devices that are available for use in the Solaris Cryptographic Framework.</td>
<td>&quot;How to List Available Providers&quot; on page 264</td>
</tr>
<tr>
<td>Add a software provider</td>
<td>Adds a PKCS #11 library or a kernel module to the Solaris Cryptographic Framework. The provider must be signed.</td>
<td>&quot;How to Add a Software Provider&quot; on page 266</td>
</tr>
<tr>
<td>Prevent the use of a user-level</td>
<td>Removes a software mechanism from use. The mechanism can be enabled again.</td>
<td>&quot;How to Prevent the Use of a User-Level Mechanism&quot; on page 268</td>
</tr>
<tr>
<td>Temporarily disable mechanisms from a</td>
<td>Temporarily removes a mechanism from use. Usually used for testing.</td>
<td>&quot;How to Prevent the Use of a Kernel Software Provider&quot; on page 269</td>
</tr>
<tr>
<td>kernel module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uninstall a provider</td>
<td>Removes a kernel software provider from use.</td>
<td>Example 14–22</td>
</tr>
<tr>
<td>List available hardware providers</td>
<td>Shows the attached hardware, shows the mechanisms that the hardware provides, and shows which mechanisms are enabled for use.</td>
<td>&quot;How to List Hardware Providers&quot; on page 272</td>
</tr>
<tr>
<td>Disable mechanisms from a hardware</td>
<td>Ensures that selected mechanisms on a hardware accelerator are not used.</td>
<td>&quot;How to Disable Hardware Provider Mechanisms and Features&quot; on page 272</td>
</tr>
<tr>
<td>provider</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restart or refresh cryptographic services</td>
<td>Ensures that cryptographic services are available.</td>
<td>&quot;How to Refresh or Restart All Cryptographic Services&quot; on page 274</td>
</tr>
</tbody>
</table>

Chapter 14 • Solaris Cryptographic Framework (Tasks) 263
Administering the Cryptographic Framework

This section describes how to administer the software providers and the hardware providers in the Solaris Cryptographic Framework. Software providers and hardware providers can be removed from use when desirable. For example, you can disable the implementation of an algorithm from one software provider. You can then force the system to use the algorithm from a different software provider.

How to List Available Providers

The Solaris Cryptographic Framework provides algorithms for several types of consumers:

- User-level providers provide a PKCS #11 cryptographic interface to applications that are linked with the libpkcs11 library
- Kernel software providers provide algorithms for IPsec, Kerberos, and other Solaris kernel components
- Kernel hardware providers provide algorithms that are available to kernel consumers and to applications through the pkcs11_kernel library

1 List the providers in a brief format.

Only those mechanisms at the user level are available for use by ordinary users.

```
cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa

kernel hardware providers:
    dca/0
```

2 List the providers and their mechanisms in the Solaris Cryptographic Framework.

All mechanisms are listed in the following output. However, some of the listed mechanisms might be unavailable for use. To list only the mechanisms that the administrator has approved for use, see Example 14–15.
The output is reformatted for display purposes.

% cryptoadm list -m
user-level providers:

/usr/lib/security/$ISA/pkcs11_kernel.so: CKM_MDS, CKM_MDS_HMAC, CKM_MDS_HMAC_GENERAL, CKM_SHA_1, CKM_SHA_1_HMAC, CKM_SHA_1_HMAC_GENERAL, ...

/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN, CKM_AES_CBC, CKM_AES_CBC_PAD, CKM_AES_ECB, CKM_AES_KEY_GEN, ...

kernel software providers:

des: CKM_DES_ECB, CKM_DES_CBC, CKM_DES3_ECB, CKM_DES3_CBC
aes: CKM_AES_ECB, CKM_AES_CBC
blowfish: CKM_BF_ECB, CKM_BF_CBC
arcfour: CKM_RC4
sha1: CKM_SHA_1, CKM_SHA_1_HMAC, CKM_SHA_1_HMAC_GENERAL
md5: CKM_MD5, CKM_MD5_HMAC, CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS, CKM_RSA_AES, CKM_RSA X 509, CKM_MD5_RSA_PKCS, CKM_SHA1_RSA_PKCS
swrand: No mechanisms presented.

kernel hardware providers:

dca/0: CKM_MDS, CKM_MDS_HMAC, CKM_MDS_HMAC_GENERAL, ...

Example 14–14  Finding the Existing Cryptographic Mechanisms

In the following example, all mechanisms that the user-level library, pkcs11_softtoken, offers are listed.

% cryptoadm list -m provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN, ...
CKM_SSL3_KEY_AND_MAC_DERIVE, CKM_TLS_KEY_AND_MAC_DERIVE

Example 14–15  Finding the Available Cryptographic Mechanisms

Policy determines which mechanisms are available for use. The administrator sets the policy. An administrator can choose to disable mechanisms from a particular provider. The -p option displays the list of mechanisms that are permitted by the policy that the administrator has set.

% cryptoadm list -p
user-level providers:

The output is reformatted for display purposes.

% cryptoadm list -m
user-level providers:

/usr/lib/security/$ISA/pkcs11_kernel.so: CKM_MDS, CKM_MDS_HMAC, CKM_MDS_HMAC_GENERAL, CKM_SHA_1, CKM_SHA_1_HMAC, CKM_SHA_1_HMAC_GENERAL, ...

/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN, CKM_AES_CBC, CKM_AES_CBC_PAD, CKM_AES_ECB, CKM_AES_KEY_GEN, ...

kernel software providers:

des: CKM_DES_ECB, CKM_DES_CBC, CKM_DES3_ECB, CKM_DES3_CBC
aes: CKM_AES_ECB, CKM_AES_CBC
blowfish: CKM_BF_ECB, CKM_BF_CBC
arcfour: CKM_RC4
sha1: CKM_SHA_1, CKM_SHA_1_HMAC, CKM_SHA_1_HMAC_GENERAL
md5: CKM_MD5, CKM_MD5_HMAC, CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS, CKM_RSA_AES, CKM_RSA X 509, CKM_MD5_RSA_PKCS, CKM_SHA1_RSA_PKCS
swrand: No mechanisms presented.

kernel hardware providers:

dca/0: CKM_MDS, CKM_MDS_HMAC, CKM_MDS_HMAC_GENERAL, ...

Example 14–14  Finding the Existing Cryptographic Mechanisms

In the following example, all mechanisms that the user-level library, pkcs11_softtoken, offers are listed.

% cryptoadm list -m provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN, ...
CKM_SSL3_KEY_AND_MAC_DERIVE, CKM_TLS_KEY_AND_MAC_DERIVE

Example 14–15  Finding the Available Cryptographic Mechanisms

Policy determines which mechanisms are available for use. The administrator sets the policy. An administrator can choose to disable mechanisms from a particular provider. The -p option displays the list of mechanisms that are permitted by the policy that the administrator has set.

% cryptoadm list -p
user-level providers:

The output is reformatted for display purposes.

% cryptoadm list -m
user-level providers:

/usr/lib/security/$ISA/pkcs11_kernel.so: CKM_MDS, CKM_MDS_HMAC, CKM_MDS_HMAC_GENERAL, CKM_SHA_1, CKM_SHA_1_HMAC, CKM_SHA_1_HMAC_GENERAL, ...

/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN, CKM_AES_CBC, CKM_AES_CBC_PAD, CKM_AES_ECB, CKM_AES_KEY_GEN, ...

kernel software providers:

des: CKM_DES_ECB, CKM_DES_CBC, CKM_DES3_ECB, CKM_DES3_CBC
aes: CKM_AES_ECB, CKM_AES_CBC
blowfish: CKM_BF_ECB, CKM_BF_CBC
arcfour: CKM_RC4
sha1: CKM_SHA_1, CKM_SHA_1_HMAC, CKM_SHA_1_HMAC_GENERAL
md5: CKM_MD5, CKM_MD5_HMAC, CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS, CKM_RSA_AES, CKM_RSA X 509, CKM_MD5_RSA_PKCS, CKM_SHA1_RSA_PKCS
swrand: No mechanisms presented.

kernel hardware providers:

dca/0: CKM_MDS, CKM_MDS_HMAC, CKM_MDS_HMAC_GENERAL, ...

Example 14–14  Finding the Existing Cryptographic Mechanisms

In the following example, all mechanisms that the user-level library, pkcs11_softtoken, offers are listed.

% cryptoadm list -m provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN, CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN, ...
CKM_SSL3_KEY_AND_MAC_DERIVE, CKM_TLS_KEY_AND_MAC_DERIVE

Example 14–15  Finding the Available Cryptographic Mechanisms

Policy determines which mechanisms are available for use. The administrator sets the policy. An administrator can choose to disable mechanisms from a particular provider. The -p option displays the list of mechanisms that are permitted by the policy that the administrator has set.

% cryptoadm list -p
user-level providers:
/usr/lib/security/$ISA/pkcs11_kernel.so: all mechanisms are enabled. 
random is enabled. 
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled. 
random is enabled.

kernel software providers:
==============================
des: all mechanisms are enabled.
aes: all mechanisms are enabled.
blowfish: all mechanisms are enabled.
arcfour: all mechanisms are enabled.
sha1: all mechanisms are enabled.
md5: all mechanisms are enabled.
rsa: all mechanisms are enabled.
swrand: random is enabled.

kernel hardware providers:
==============================
dca/0: all mechanisms are enabled. random is enabled.


How to Add a Software Provider

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and 
   assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in 
   System Administration Guide: Basic Administration.

2 List the software providers that are available to the system.
   
   # cryptoadm list
   
   user-level providers:
   /usr/lib/security/$ISA/pkcs11_kernel.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so
   
   kernel software providers:
   des
   aes
   blowfish
   arcfour
   sha1
   md5
   rsa
   swrand
   
   kernel hardware providers:
   dca/0
Add the provider’s package by using the pkgadd command.

```
# pkgadd -d /path/to/package pkginst
```

The package must include software that has been signed by a certificate from Sun. To request a certificate from Sun and to sign a provider, see Appendix F, "Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

The package should have scripts that notify the cryptographic framework that another provider with a set of mechanisms is available. For information on the packaging requirements, see Appendix F, "Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

Refresh the providers.

You need to refresh providers if you added a software provider, or if you added hardware and specified policy for the hardware.

```
# svcadm refresh svc:/system/cryptosvc
```

Locate the new provider on the list.

In this case, a new kernel software provider was installed.

```
# cryptoadm list
...
kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand
    ecc  <-- added provider
...
```

Adding a User-Level Software Provider

In the following example, a signed PKCS #11 library is installed.

```
# pkgadd -d /cdrom/cdrom0/SolarisNew
Answer the prompts
# svcadm refresh system/cryptosvc
# cryptoadm list
user-level providers:
==================================
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so
    /opt/SUNWconn/lib/$ISA/libpkcs11.so.1  <-- added provider
```

Developers who are testing a library with the cryptographic framework can install the library manually.
Cryptoadm install provider=/opt/SUNWconn/lib/'$ISA'/libpkcs11.so.1

How to Prevent the Use of a User-Level Mechanism

If some of the cryptographic mechanisms from a library provider should not be used, you can remove selected mechanisms. This procedure uses the DES mechanisms in the pkcs11_softtoken library as an example.

1. Become superuser or assume a role that includes the Crypto Management rights profile.
   To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.

2. List the mechanisms that are offered by a particular user-level software provider.
   
   % cryptoadm list -m provider=/usr/lib/security/’$ISA’/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so:
   CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN,
   CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN,
   CKM_AES_CBC, CKM_AES_CBC_PAD, CKM_AES_ECB, CKM_AES_KEY_GEN,
   ...

3. List the mechanisms that are available for use.
   
   $ cryptoadm list -p
   user-level providers:
   .................................................................
   ...
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
   random is enabled.
   ...

4. Disable the mechanisms that should not be used.
   
   $ cryptoadm disable provider=/usr/lib/security/’$ISA’/pkcs11_softtoken.so \ 
   > mechanism=CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB

5. List the mechanisms that are available for use.
   
   $ cryptoadm list -p provider=/usr/lib/security/’$ISA’/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled, 
   except CKM_DES_ECB, CKM_DES_CBC_PAD, CKM_DES_CBC. random is enabled.

Example 14–17 Enabling a User-Level Software Provider Mechanism

In the following example, a disabled DES mechanism is again made available for use.

$ cryptoadm list -m provider=/usr/lib/security/’$ISA’/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so:
   CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN,
 CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN,
...
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB, CKM_DES_CBC_PAD, CKM_DES_CBC. random is enabled.
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
> mechanism=CKM_DES_ECB
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_CBC_PAD, CKM_DES_CBC. random is enabled.

Example 14–18 Enabling All User-Level Software Provider Mechanisms

In the following example, all mechanisms from the user-level library are enabled.

$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so all
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.

Example 14–19 Permanently Removing User-Level Software Provider Availability

In the following example, the libpkcs11.so.1 library is removed.

$ cryptoadm uninstall provider=/opt/SUNWconn/lib/\$ISA/libpkcs11.so.1
$ cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
...

▼ How to Prevent the Use of a Kernel Software Provider

If the cryptographic framework provides multiple modes of a provider such as AES, you might
remove a slow mechanism from use, or a corrupted mechanism. This procedure uses the AES
algorithm as an example.

1 Become superuser or assume a role that includes the Crypto Management rights profile.
   To create a role that includes the Crypto Management rights profile and assign the role
to a user, see Example 9–7.

2 List the mechanisms that are offered by a particular kernel software provider.
   $ cryptoadm list -m provider=aes
   aes: CKM_AES_ECB, CKM_AES_CBC
3 List the mechanisms that are available for use.
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled.

4 Disable the mechanism that should not be used.
   $ cryptoadm disable provider=aes mechanism=CKM_AES_ECB

5 List the mechanisms that are available for use.
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled, except CKM_AES_ECB.

Example 14–20 Enabling a Kernel Software Provider Mechanism

In the following example, a disabled AES mechanism is again made available for use.

   cryptoadm list -m provider=aes
   aes: CKM_AES_ECB, CKM_AES_CBC
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled, except CKM_AES_ECB.
   $ cryptoadm enable provider=aes mechanism=CKM_AES_ECB
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled.

Example 14–21 Temporarily Removing Kernel Software Provider Availability

In the following example, the AES provider is temporarily removed from use. The unload subcommand is useful to prevent a provider from being loaded automatically while the provider is being uninstalled. For example, the unload subcommand would be used when installing a patch that affects the provider.

   $ cryptoadm unload provider=aes
   $ cryptoadm list
   ...
   kernel software providers:
   des
   aes (inactive)
   blowfish
   arcfour
   sha1
   md5
   rsa
   swrand

   The AES provider is unavailable until the cryptographic framework is refreshed.

   $ svcadm refresh system/cryptosvc
   $ cryptoadm list
   ...
If a kernel consumer is using the kernel software provider, the software is not unloaded. An error message is displayed and the provider continues to be available for use.

**Example 14–22** Permanently Removing Software Provider Availability

In the following example, the AES provider is removed from use. Once removed, the AES provider does not appear in the policy listing of kernel software providers.

```
$ cryptoadm uninstall provider=aes
$ cryptoadm list
... kernel software providers:
   des
   blowfish
   arcfour
   sha1
   md5
   rsa
   swrand
```

If a kernel consumer is using the kernel software provider, an error message is displayed and the provider continues to be available for use.

**Example 14–23** Reinstalling a Removed Kernel Software Provider

In the following example, the AES kernel software provider is reinstalled.

```
$ cryptoadm install provider=aes mechanism=CKM_AES_ECB,CKM_AES_CBC
$ cryptoadm list
... kernel software providers:
   des
   aes
   blowfish
   arcfour
   sha1
   md5
   rsa
   swrand
```
How to List Hardware Providers

Hardware providers are automatically located and loaded. For more information, see driver.conf(4) man page.

Before You Begin

When you add hardware that expects to be used within the Solaris Cryptographic Framework, the hardware registers with the SPI in the kernel. The framework checks that the hardware driver is signed. Specifically, the framework checks that the object file of the driver is signed with a certificate that Sun issues.

1. List the hardware providers that are available on the system.
   
   ```
   % cryptoadm list
   ...
   kernel hardware providers:
   dca/0
   ```

2. List the mechanisms that the board provides.
   
   ```
   % cryptoadm list -m provider=dca/0
   dca/0: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL,...
   ```

3. List the mechanisms that are available for use on the board.
   
   ```
   % cryptoadm list -p provider=dca/0
   dca/0: all mechanisms are enabled. random is enabled.
   ```

How to Disable Hardware Provider Mechanisms and Features

You can selectively disable mechanisms and the random number feature from a hardware provider. To enable them again, see Example 14–24.

1. List the mechanisms and features that are available from the board.
   
   ```
   % cryptoadm list -p provider=dca/0
   dca/0: all mechanisms are enabled. random is enabled.
   ```

2. Become superuser or assume a role that includes the Crypto Management rights profile.
   
   To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.
3 Choose the mechanisms or feature to disable:

- Disable selected mechanisms.
  
  ```
  # cryptoadm list -m provider=dca/0
  dca/0: CKM_MD5, CKM_MD5_HMAC, CKM_MD5_HMAC_GENERAL,...
  CKM_DES_ECB, CKM_DES3_ECB...
  random is enabled.
  # cryptoadm disable provider=dca/0 mechanism=CKM_DES_ECB, CKM_DES3_ECB
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled except CKM_DES_ECB, CKM_DES3_ECB. random is enabled.
  ```

- Disable the random number generator.
  
  ```
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is enabled.
  # cryptoadm disable provider=dca/0 random
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is disabled.
  ```

- Disable all mechanisms. Do not disable the random number generator.
  
  ```
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is enabled.
  # cryptoadm disable provider=dca/0 mechanism=all
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are disabled. random is enabled.
  ```

- Disable every feature and mechanism on the hardware.
  
  ```
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is enabled.
  # cryptoadm disable provider=dca/0 all
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are disabled. random is disabled.
  ```

Example 14–24 Enabling Mechanisms and Features on a Hardware Provider

In the following examples, disabled mechanisms on a piece of hardware are selectively enabled.

```
# cryptoadm list -p provider=dca/0
 dca/0: all mechanisms are enabled except CKM_DES_ECB, CKM_DES3_ECB. random is enabled.
# cryptoadm enable provider=dca/0 mechanism=CKM_DES3_ECB
# cryptoadm list -p provider=dca/0
 dca/0: all mechanisms are enabled except CKM_DES_ECB. random is enabled.
```

In the following example, only the random generator is enabled.

```
# cryptoadm list -p provider=dca/0
 dca/0: all mechanisms are enabled, except CKM_MD5, CKM_MD5_HMAC,...
```
random is disabled.

```bash
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
```

dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is enabled.

In the following example, only the mechanisms are enabled. The random generator continues to be disabled.

```bash
# cryptoadm list -p provider=dca/0
```

dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.

```bash
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
```

dca/0: all mechanisms are enabled. random is disabled.

In the following example, every feature and mechanism on the board is enabled.

```bash
# cryptoadm list -p provider=dca/0
```

dca/0: all mechanisms are enabled, except CKM_DES ECB,CKM_DES3 ECB.
random is disabled.

```bash
# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
```

dca/0: all mechanisms are enabled. random is enabled.

▶ How to Refresh or Restart All Cryptographic Services

By default, the Solaris Cryptographic Framework is enabled. When the kcfd daemon fails for any reason, the service management facility can be used to restart cryptographic services. For more information, see the smf(5) and svcadm(1M) man pages. For the effect on zones of restarting cryptographic services, see “Cryptographic Services and Zones” on page 254.

1 Check the status of cryptographic services.

```bash
% svcs */cryptosvc/*
STATE STIME FMRI
offline Dec_09 svc:/system/cryptosvc:default
```

2 Become superuser or assume an equivalent role to enable cryptographic services.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.

```bash
# svcadm enable svc:/system/cryptosvc
```

Example 14–25 Refreshing Cryptographic Services

In the following example, cryptographic services are refreshed in the global zone. Therefore, kernel-level cryptographic policy in every non-global zone is also refreshed.

```bash
# svcadm refresh system/cryptosvc
```
This section discusses authentication services that can be configured on a non-networked system, or between two systems. To configure a network of authenticated users and systems, see Part VI.
This chapter provides information about how to use Secure RPC to authenticate a host and a user across an NFS mount. The following is a list of the topics in this chapter.

- “Overview of Secure RPC” on page 277
- “Administering Secure RPC (Task Map)” on page 281

Overview of Secure RPC

Secure RPC (Remote Procedure Call) protects remote procedures with an authentication mechanism. The Diffie-Hellman authentication mechanism authenticates both the host and the user who is making a request for a service. The authentication mechanism uses Data Encryption Standard (DES) encryption. Applications that use Secure RPC include NFS and the name services, NIS and NIS+.

NFS Services and Secure RPC

NFS enables several hosts to share files over the network. Under the NFS service, 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 who are logged in to the client systems can access the file systems by mounting the file systems from the server. To the user on the client system, it appears as if the files are local to the client. One of the most common uses of NFS allows systems to be installed in offices, while storing all user files in a central location. Some features of the NFS service, such as the -nosuid option to the mount command, can be used to prohibit the opening of devices and file systems by unauthorized users.

The NFS service uses Secure RPC to authenticate users who make requests over the network. This process is known as Secure NFS. The Diffie-Hellman authentication mechanism, AUTH_DH, uses DES encryption to ensure authorized access. The AUTH_DH mechanism has also been called AUTH_DES. For more information, see the following:
To set up and administer Secure NFS, see “Administering the Secure NFS System” in System Administration Guide: Network Services.

To set up the NIS+ tables and enter names in the cred table, see System Administration Guide: Naming and Directory Services (NIS+).

For an outline of the transactions that are involved in RPC authentication, see “Implementation of Diffie-Hellman Authentication” on page 279.

**DES Encryption With Secure NFS**

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 by using the key to encipher and decipher text. DES is a relatively fast encryption mechanism. A DES chip makes the encryption even faster. However, 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 that were 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 need to change the keys frequently.

**Kerberos Authentication**

Kerberos is an authentication system that was developed at MIT. Some encryption in Kerberos is based on DES. Kerberos V4 support is no longer supplied as part of Secure RPC. However, a client-side and server-side implementation of Kerberos V5, which uses RPCSEC_GSS, is included with this release. For more information, see Chapter 20.

**Diffie-Hellman Authentication**

The Diffie-Hellman (DH) method of authenticating a user is nontrivial for an intruder to crack. The client and the server have their own private key, which they use with the public key to devise a common key. The private key is also known as the secret key. The client and the server use the common key to communicate with each other. The common key is encrypted with an agreed-upon encryption function, such as DES.

Authentication is based on the ability of the sending system to use the common key to encrypt the current time. Then, the receiving system can decrypt and check against its current time. The time on the client and the server must be synchronized. For more information, see “Managing Network Time Protocol (Tasks)” in System Administration Guide: Network Services.

The public keys and private keys are stored in an NIS or NIS+ database. NIS stores the keys in the publickey map. 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 maps or NIS+ tables, and for generating a public key and a private key for each user. The private key is stored in encrypted form with the user’s password. This process 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 that use Diffie-Hellman authentication (AUTH_DH).

**Generating the Public Keys and Secret Keys**

Sometime prior to a transaction, the administrator runs either the **newkey** or the **nisaddcred** command to generate 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. The **chkey** command changes the key pair.

**Running the keylogin Command**

Normally, the login password is identical to the Secure RPC password. In this case, the **keylogin** command is not required. However, if the passwords are different, the users have to log in and then run the **keylogin** command.

The **keylogin** command prompts the user for a Secure RPC password. The command then uses the password to decrypt the secret key. The **keylogin** command then passes the decrypted secret key to the **keyserver** program. The keys server is an RPC service with a local instance on every computer. The keys server saves the decrypted secret key and waits for the user to initiate a Secure RPC transaction with a server.

If both the login password and the RPC password are the same, the login process passes the secret key to the keys server. If the passwords are required to be different, then the user must always run the **keylogin** command. When the **keylogin** command is included in the user’s environment configuration file, such as the ~/.login, ~/.cshrc, or ~/.profile file, the **keylogin** command runs automatically whenever the user logs in.

**Generating the Conversation Key**

When the user initiates a transaction with a server, the following occurs:

1. The keys server randomly generates a conversation key.
2. The kernel uses the conversation key, plus other material, to encrypt the client’s timestamp.
3. The keys server looks up the server’s public key in the public key database. For more information, see the publickey(4) man page.
4. The keys server uses the client’s secret key and the server’s public key to create a common key.
5. The keys server encrypts the conversation key with the common key.
Initially Contacting the Server

The transmission, which includes the encrypted timestamp 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 network name
- The conversation key, which is encrypted with the common key
- A “window,” which is encrypted with the conversation key

The window is the difference in time that the client says should be allowed between the server’s clock and the client’s timestamp. If the difference between the server’s clock and the timestamp is greater than the window, the server rejects the client’s request. Under normal circumstances, this rejection does not happen, because the client first synchronizes with the server before starting the RPC session.

The client’s verifier contains the following:

- The encrypted timestamp
- An encrypted verifier of the specified window, which is decremented by 1

The window verifier is needed in case somebody wants to impersonate a user. The impersonator can write a program that, instead of filling in the encrypted fields of the credential and verifier, just inserts random bits. The server decrypts the conversation key into some random key. The server then uses the key to try to decrypt the window and the timestamp. The result is random numbers. After a few thousand trials, however, the random window/timestamp pair is likely to pass the authentication system. The window verifier lessens the chance that a fake credential could be authenticated.

Decrypting the Conversation Key

When the server receives the transmission from the client, the following occurs:

1. The keyserver that is local to the server looks up the client’s public key in the public key database.
2. The keyserver uses the client’s public key and the server’s secret key to deduce the common key. The common key is the same common key that is computed by the client. Only the server and the client can calculate the common key because the calculation requires knowing one of the secret keys.
3. The kernel uses the common key to decrypt the conversation key.
4. The kernel calls the keyserver to decrypt the client’s timestamp with the decrypted conversation key.

Storing Information on the Server

After the server decrypts the client’s timestamp, the server stores four items of information in a credential table:

- The client’s computer name
- The conversation key
- The window
The client’s timestamp

The server stores the first three items for future use. The server stores the client’s timestamp to protect against replays. The server accepts only timestamps that are chronologically greater than the last timestamp seen. As a result, any replayed transactions are guaranteed to be rejected.

**Note** – Implicit in these transactions is the name of the caller, who must be authenticated in some manner. The key server cannot use DES authentication to authenticate the caller because the use of DES by the key server would create a deadlock. To avoid a deadlock, the key server stores the secret keys by user ID (UID) and grants requests only to local root processes.

Returning the Verifier to the Client

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

- The index ID, which the server records in its credential cache
- The client’s timestamp minus 1, which is encrypted by the conversation key

The reason for subtracting 1 from the client’s timestamp is to ensure that the timestamp is out of date. An out-of-date timestamp cannot be reused as a client verifier.

Authenticating 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 timestamp the client sent.

Handling Transactions

With every transaction after the first transaction, the client returns the index ID to the server in its next transaction. The client also sends another encrypted timestamp. The server sends back the client’s timestamp minus 1, which is encrypted by the conversation key.

---

**Administering Secure RPC (Task Map)**

The following task map points to procedures that configure Secure RPC for NIS, NIS+, and NFS.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start the keyserver.</td>
<td>Ensures that keys can be created so that users can be authenticated.</td>
<td>“How to Restart the Secure RPC Keyserver” on page 282</td>
</tr>
<tr>
<td>2. Set up credentials on a NIS+ host.</td>
<td>Ensures that the root user on a host can be authenticated in an NIS+ environment.</td>
<td>“How to Set Up a Diffie-Hellman Key for an NIS+ Host” on page 282</td>
</tr>
</tbody>
</table>
Administering Authentication With Secure RPC

By requiring authentication for use of mounted NFS file systems, you increase the security of your network.

**How to Restart the Secure RPC Keyserver**

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Verify that the keyserv daemon is running.**
   ```
   # svcs \*keyserv\*
   STATE STIME FMRI
   disabled Dec_14 svc:/network/rpc/keyserv
   ```

3. **Enable the keyserv daemon service if the service is not online.**
   ```
   # svcadm enable network/rpc/keyserv
   ```

**How to Set Up a Diffie-Hellman Key for an NIS+ Host**

This procedure should be done on every host in the NIS+ domain. After *root* has run the `keylogin` command, the server has GSS-API acceptor credentials for `mech_dh` and the client has GSS-API initiator credentials.

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

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Give an NIS+ user a key.</td>
<td>Enables a user to be authenticated in an NIS+ environment.</td>
<td>“How to Set Up a Diffie-Hellman Key for an NIS+ User” on page 284</td>
</tr>
<tr>
<td>4. Set up credentials on an NIS host.</td>
<td>Ensures that the <em>root</em> user on a host can be authenticated in an NIS environment.</td>
<td>“How to Set Up a Diffie-Hellman Key for an NIS Host” on page 284</td>
</tr>
<tr>
<td>5. Give an NIS user a key.</td>
<td>Enables a user to be authenticated in an NIS environment.</td>
<td>“How to Set Up a Diffie-Hellman Key for an NIS User” on page 285</td>
</tr>
</tbody>
</table>
1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration.*

2 **Enable the public key table in the name service.**

   Add the following line to the `/etc/nsswitch.conf` file:
   
   publickey: nisplus

3 **Initialize the NIS+ client.**

   ```
   # nisinit -cH hostname
   ```

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

4 **Add the client to the cred table.**

   Type 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.
   
   ```
   # keylogin
   Password:
   ```

---

**Example 15–1 Setting Up a New Key for root on an 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 system is in the example.com. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@example.com
Adding new key for unix.earth@example.com (earth.example.com.)
Network password: <Type password>
Warning, password differs from login password.
Retype password: <Retype password>
# keylogin
Password: <Type password>
#
```
How to Set Up a Diffie-Hellman Key for an NIS+ User

This procedure should be done on every user in the NIS+ domain.

1 **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Add the user to the cred table on the root master server.**
   Type the following command:
   
   ```
   # nisaddcred -p unix.UID@domain-name -P username.domain-name. des
   ```
   
   Note that, in this case, the *username.domain-name* must end with a dot (.)

3 **Verify the setup by logging in as the client and typing the keylogin command.**

Example 15–2 Setting Up a New Key for an NIS+ User

In the following example, a key for Diffie-Hellman authentication is given to the user jdoe.

```
# nisaddcred -p unix.1234@example.com -P jdoe.example.com. des
DES principal name : unix.1234@example.com
Adding new key for unix.1234@example.com (jdoe.example.com.)
Password: <Type password>
Retype password: <Retype password>
# rlogin rootmaster -l jdoe
% keylogin
% Password: <Type password>
%```
Create a new key pair by using the `newkey` command.

```
# newkey -h hostname
```

where `hostname` is the name of the client.

**Example 15–3 Setting Up a New Key for root on an NIS Client**

In the following example, `earth` is set up as a secure NIS client.

```
# newkey -h earth
Adding new key for unix.earth@example.com
New Password: <Type password>
Retype password: <Retype password>
Please wait for the database to get updated...
Your new key has been successfully stored away.
```

▼ **How to Set Up a Diffie-Hellman Key for an NIS User**

This procedure should be done for every user in the NIS domain.

**Before You Begin** Only system administrators, when logged in to the NIS master server, can generate a new key for a user.

1 **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 **Create a new key for a user.**

   ```
   # newkey -u username
   ```

   where `username` is the name of the user. The system prompts for a password. You can type a generic password. The private key is stored in an encrypted form by using the generic password.

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

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

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

**Example 15–4 Setting Up and Encrypting a New User Key in NIS**

In this example, superuser sets up the key.
# newkey -u jdoe
Adding new key for unix.12345@example.com
New Password: <Type password>
Retype password: <Retype password>
Please wait for the database to get updated...
Your new key has been successfully stored away.
#

Then the user jdoe re-encrypts the key with a private password.
%
chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@example.com
Please enter the Secure-RPC password for jdoe: <Type password>
Please enter the login password for jdoe: <Type password>
Sending key change request to centralexample...

▼ How to Share NFS Files With Diffie-Hellman Authentication

This procedure protects shared file systems on an NFS server by requiring authentication for access.

Before You Begin  Diffie-Hellman public key authentication must be enabled on the network. To enable authentication on the network, do one of the following:

- “How to Set Up a Diffie-Hellman Key for an NIS+ Host” on page 282
- “How to Set Up a Diffie-Hellman Key for an NIS Host” on page 284

1 Become superuser or assume a role that includes the System Management profile.  
The System Administrator role includes the System Management profile. To create the role and assign the role to a user, see “Configuring RBAC (Task Map)” on page 188.

2 On the NFS server, share a file system with Diffie-Hellman authentication.  
# share -F nfs -o sec=dh /filesystem
where filesystem is the file system that is being shared.

The -o sec=dh option means that AUTH_DH authentication is now required to access the file system.

3 On an NFS client, mount a file system with Diffie-Hellman authentication.  
# mount -F nfs -o sec=dh server:filesystem mount-point
server Is the name of the system that is sharing filesystem
filesystem Is the name of the file system that is being shared, such as opt
mount-point Is the name of the mount point, such as /opt
The `sec=dh` option mounts the file system with AUTH_DH authentication.
This chapter covers the Pluggable Authentication Module (PAM) framework. PAM provides a method to “plug in” authentication services into the Solaris Operating System (Solaris OS). PAM provides support for multiple authentication services when accessing a system.

- “PAM (Overview)” on page 289
- “PAM (Tasks)” on page 292
- “PAM Configuration File (Reference)” on page 294

## PAM (Overview)

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

## Benefits of Using PAM

The PAM framework enables you to configure the use of system entry services (such as `ftp`, `login`, `telnet`, or `rsh`) for user authentication. Some benefits that PAM provides are as follows:

- Flexible configuration policy
  - Per-application authentication policy
  - The ability to choose a default authentication mechanism
  - The ability to require multiple passwords on high-security systems
- Ease of use for the end user
  - No retyping of passwords if the passwords are the same for different authentication services
  - The ability to prompt the user for passwords for multiple authentication services without requiring the user to type multiple commands
- The ability to pass optional options to the user authentication services
- The ability to implement a site-specific security policy without having to change the system entry services

**PAM Components**

The PAM software consists of a library, various service modules, and a configuration file. Solaris commands or daemons that take advantage of these PAM interfaces are also included.

The following figure illustrates the relationship between the system entry applications, the PAM library, the `pam.conf` file, and the PAM service modules.

![Figure 16-1: How PAM Works](image)

The system entry applications, such as `ftp`, `telnet`, and `login`, use the PAM library to call the configuration policy. The configuration policy is defined in the `pam.conf` file. The `pam.conf` file defines which modules to use, and in what order the modules are to be used with each application. Results from the modules are based on the module responses and the configured control flags. These results are passed back through the library to the application.

**PAM Framework**

The PAM framework provides a method for authenticating users with multiple services by 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.

The PAM library provides the framework to load the appropriate modules and to manage the stacking process. The PAM library provides a generic structure to which all of the modules can plug in. See the `pam_sm(3PAM)` man page for more information.
Changes to PAM for the Solaris 10 Release

The Solaris 10 release includes the following changes to the Pluggable Authentication Module (PAM) framework:

- The `pam_authtok_check` module now allows for strict password checking using new tunable parameters in the `/etc/default/passwd` file. The new parameters define:
  - A list of comma separated dictionary files used for checking common dictionary words in a password
  - The minimum differences required between a new password and an old password
  - The minimum number of alphabetic or nonalphabetic characters that must be used in a new password
  - The minimum number of uppercase or lowercase letters that must be used in a new password
  - The number of allowable consecutive repeating characters

- The `pam_unix_auth` module implements account locking for local users. Account locking is enabled by the `LOCK_AFTER_RETRIES` parameter in `/etc/security/policy.conf` and the `lock_after-retries` key in `/etc/user_attr`. See the `policy.conf(4)` and the `user_attr(4)` man pages for more information.

- A new binding control flag has been defined. This control flag is documented in the `pam.conf(4)` man page and in “PAM Control Flags” on page 295.

- The `pam_unix` module has been removed and replaced by a set of service modules of equivalent or greater functionality. Many of these modules were introduced in the Solaris 9 release. Here is a list of the replacement modules:
  - `pam_authtok_check`
  - `pam_authtok_get`
  - `pam_authtok_store`
  - `pam_dhkeys`
  - `pam_passwd_auth`
  - `pam_unix_account`
  - `pam_unix_auth`
  - `pam_unix_credential`
  - `pam_unix_session`

- The functionality of the `pam_unix_auth` module has been split into two modules. The `pam_unix_auth` module now verifies that the password is correct for the user. The new `pam_unix_credential` module provides functions that establish user credential information.

- Additions to the `pam_krb5` module have been made to manage the Kerberos credentials cache using the PAM framework.

- A new `pam_deny` module has been added. The module can be used to deny access to services. By default, the `pam_deny` module is not used. For more information, see the `pam_deny(5)` man page.
This section discusses some tasks that might be required to make the PAM framework use a particular security policy. You should be aware of some security issues that are associated with the PAM configuration file. For information about the security issues, see “Planning for Your PAM Implementation” on page 292.

### PAM (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan for your PAM installation.</td>
<td>Consider configuration issues and make decisions about them before you start the software configuration process.</td>
<td>&quot;Planning for Your PAM Implementation” on page 292</td>
</tr>
<tr>
<td>Add new PAM modules.</td>
<td>Sometimes, site-specific modules must be written and installed to cover requirements that are not part of the generic software. This procedure explains how to install these new PAM modules.</td>
<td>&quot;How to Add a PAM Module&quot; on page 293</td>
</tr>
<tr>
<td>Block access through ~/.rhosts.</td>
<td>Further increase security by preventing access through ~/.rhosts.</td>
<td>&quot;How to Prevent Rhost-Style Access From Remote Systems With PAM” on page 294</td>
</tr>
<tr>
<td>Initiate error logging.</td>
<td>Start the logging of PAM error messages through syslog.</td>
<td>&quot;How to Log PAM Error Reports” on page 294</td>
</tr>
</tbody>
</table>

### Planning for Your PAM Implementation

As delivered, the `pam.conf` configuration file implements the standard Solaris security policy. This policy should work in many situations. If you need to implement a different security policy, here are the issues that you should focus on:

- Determine what your needs are, especially which PAM service modules you should select.
- Identify the services that need special configuration options. Use `other` if appropriate.
- Decide the order in which the modules should be run.
- Select the control flag for each module. See “PAM Control Flags” on page 295 for more information about all of the control flags.
- Choose any options that are necessary for each module. The man page for each module should list any special options.

Here are some suggestions to consider before you change the PAM configuration file:

- Use `other` entries for each module type so that every application does not have to be included in `/etc/pam.conf`.
Make sure to consider the security implications of the binding, sufficient, and optional control flags.

Review the man pages that are associated with the modules. These man pages can help you understand how each module functions, what options are available, and the interactions between stacked modules.

Caution – If the PAM configuration file is misconfigured or the file becomes corrupted, no user might be able to log in. Because the su!login command does not use PAM, the root password would then be required to boot the machine into single-user mode and fix the problem.

After you change the /etc/pam.conf file, review the file as much as possible while you still have system access to correct problems. Test all the commands that might have been affected by your changes. An example is adding a new module to the telnet service. In this example, you would use the telnet command and verify that your changes make the service behave as expected.

How to Add a PAM Module

This procedure shows how to add a new PAM module. New modules can be created to cover site-specific security policies or to support third party applications.

1 Become superuser or assume an equivalent role.
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.

2 Determine which control flags and which other options should be used.
   Refer to “PAM Modules” on page 297 for information on the modules.

3 Ensure that the ownership and permissions are set so that the module file is owned by root and the permissions are 555.

4 Edit the PAM configuration file, /etc/pam.conf, and add this module to the appropriate services.

5 Verify that the module has been added properly.
   You must test before the system is rebooted in case the configuration file is misconfigured. Login using a direct service, such as r!login or telnet, and run the su command, before you reboot the system. The service might be a daemon that is spawned only once when the system is booted. Then, you must reboot the system before you can verify that the module has been added.
PAM Configuration File (Reference)

How to Prevent Rhost-Style Access From Remote Systems With PAM

1 Become superuser or assume an equivalent role.
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.

2 Remove all of the lines that include rhosts_auth.so.1 from the PAM configuration file.
   This step prevents the reading of the ~/.rhosts files during an rlogin session. Therefore, this step 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.

3 Disable the rsh service.
   To prevent other unauthenticated access to the ~/.rhosts files, remember to disable the rsh service.
   # svcadm disable network/shell

How to Log PAM Error Reports

1 Become superuser or assume an equivalent role.
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 188.

2 Configure the /etc/syslog.conf file for the level of logging that you need.
   See the syslog.conf(4) for more information about the logging levels.

3 Refresh the configuration information for the syslog daemon.
   # svcadm refresh system/system-log

PAM Configuration File (Reference)

The PAM configuration file, pam.conf, determines the authentication service modules to be used, and the order in which the modules are used. This file can be modified to select authentication modules for each system entry application.

PAM Configuration File Syntax

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

```
service-name module-type control-flag module-path module-options
```
service-name Is the name of the system entry service, for example, ftp, login, telnet.
module-type Is the module type for the service. For more information, see “PAM Module Types” on page 295.
control-flag Determines the continuation or failure behavior for the module.
module-path Specifies the path to the library object that implements the security policy.
module-options Specifies the 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 – If an error is found in an entry in the PAM configuration file, a syslog error message is generated. If the error is in an entry for a requested service, then the service might return an error.

Service Names for PAM
The specific service names for each service should be documented in the man page for that service. For instance the sshd(1M) man page lists all of the PAM service names for the sshd command.

PAM Module Types
You need to understand the PAM module types because the types define the interface to the module. Here are the types of PAM modules:

- **Account modules** check for password aging, account expiration, and access restrictions. After the user’s identity is authenticated through the authentication modules, the account modules determine if the user should be given access to the system.
- **Authentication modules** provide authentication for the users. The modules also allow for credentials to be set, refreshed, or destroyed.
- **Password modules** allow for changes to the user’s password.
- **Session modules** manage the opening and the closing of a login session. These modules also can log activity or provide cleanup after the session is over.

PAM Control Flags
A request using a PAM service module returns one of three states:

- success – my security policy has been met
- failure – my security policy has not been met
ignore – this request is not participating in the policy request

Each module in a stack can determine the success or failure of a request. To determine the continuation or failure behavior for a module, you must select a control flag for each entry in the PAM configuration file.

Continuation behavior defines if any following modules are checked. Depending on the response from a particular module, you can decide to skip any additional modules.

Failure behavior defines how error messages are logged or reported. Failures are either optional or required. A required failure causes that request to fail, even if other modules succeed. An optional failure does not always cause the request to fail.

The control flags are as follows:

- binding – With this control flag, if the module is successful and no preceding modules that are flagged as required have failed, then PAM skips the remaining modules and returns success. If a failure is returned, PAM records a required failure and then continues processing the stack.
  
The binding control flag is similar to the required control flag, except that no additional module checking is done if the module is successful. A failure in a module that uses this flag prevents the request from being successful, regardless of the response of any other modules. A success in a module that uses this flag makes the request successful if no preceding required modules failed.

- required – With this control flag, if the module is successful, PAM records a required success and continues checking any following modules. If the module fails, and if this failure is the first required failure, PAM saves the error message and continues checking the stack. If this failure is not the first failure, PAM just continues checking the stack. This flag allows for the entire sequence to be processed, so that information that could assist an attacker is not disclosed. All the attacker can find out is that the request failed.
  
The required control flag should be used when a particular module must succeed for the request to be successful. A failure in a module that uses this flag prevents the request from being successful, regardless of the response of any other modules. A success in a module that uses this flag does not mean that the request is successful. All of the responses from the other modules in the stack with required, requisite, or binding control flags must be successful for the request to succeed.

- requisite – With this control flag, if the module is successful, PAM records a required success and continues checking any following modules. If the module fails, PAM records a required failure, returns the error message of the first required failure, and then skips any additional checking.
  
The requisite control flag is similar to the required control flag, except that no additional module checking is done if the module fails. A failure in a module that uses this flag prevents the request from being successful, regardless of the response of any other modules. A success in a module that uses this flag does not mean that the request is successful. All of the responses from the other modules in the stack with required, requisite, or binding control flags must be successful for the request to succeed.
• **optional** – With this control flag, if the module is successful, PAM records an optional success and continues checking the stack. If the module fails, PAM records an optional failure and continues checking the stack.

The optional control flag should be used when successful authentication in the stack is adequate for a user to be authenticated. This flag should only be used if this particular service does not need to succeed. The success or failure of the request is determined by any required failures or successes.

If your users need to have permissions associated with a specific service to get their work done, then you should not label the module as optional.

• **sufficient** – With this control flag, if the module is successful, and no preceding modules that are flagged as required have failed, then PAM skips the remaining modules and returns success. If the module fails, PAM records an optional failure and continues checking the stack.

The sufficient control flag is similar to the optional control flag, except that no additional module checking is done if the module succeeds. A success in a module that uses this flag makes the request successful if no preceding required modules failed. A failure in a module that uses this flag causes the request to fail if no other modules succeeded.

More information about these control flags is provided in the following section, which describes a generic /etc/pam.conf file.

### PAM Modules

Every PAM module implements a specific function. When you set up PAM authentication, you need to specify both the module and the module type, which defines what the module does. More than one module type, such as auth, account, session, or password, can be implemented by a single module.

The path of each module is determined by the instruction set that is available in the installed Solaris release. For 32-bit modules, the path to the modules is /usr/lib/security. For 64-bit modules, the path is /usr/lib/security/$ISA. See the isalist(5) man page for more information.

A complete list of the Solaris PAM modules can be found by looking in /usr/lib/security/$ISA. Each module has an associated man page which describes the module types that apply. The man page also describes any special options.

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 does not load the module.

### Examples From the Generic `pam.conf` File

The generic /etc/pam.conf file includes the following entries:

```plaintext
login auth requisite pam_authtok_get.so.1
login auth required pam_dhkeys.so.1
```
When the `login` command is run, authentication must succeed for the `pam_authtok_get`, `pam_dhkeys`, `pam_auth_cred`, `pam_auth_unix`, and `pam_dial_auth` modules. The requisite flag on the `pam_authtok_get` entry means that if this module fails, no additional module checking is done. However, if the module is successful, then the rest of the modules are checked. If authentication for any of the modules fails, then the request for authentication also fails.

For the `rlogin` command, the sufficient control flag indicates that authentication through the `pam_rhosts_auth` module is adequate for the authentication request to succeed. No additional checking needs to be done. Authentication through the `pam_authtok_get`, `pam_dhkeys`, `pam_auth_cred`, and `pam_auth_unix` modules must succeed if authentication through `pam_rhosts_auth` fails. A failure in the `pam_rhosts_auth` module does not prevent successful authentication, although a failure in the other modules would. Also, as in the entries for `login`, the requisite control flag on the `pam_authtok_get` entry means that if this module fails, the authentication request fails and no additional module checking is done.

The other service name allows a default service to be set for any other commands that are not included in the `pam.conf` file. The other service name simplifies administration of the file, because many services that are using the same module can be covered by only one entry. Also, the other service name, when used as a "catch-all," can ensure that each access is covered by one module.

The entry for the `module-path` is "root-relative." If the file name that you specify for `module-path` does not begin with a slash (/), the path `/usr/lib/security/$ISA` precedes the file name. A full path name must be used for modules that are located in other directories. The values for the `module-options` can be found in the man page for each module.
This chapter includes information about the Simple Authentication and Security Layer (SASL).

- “SASL (Overview)” on page 299
- “SASL (Reference)” on page 300

**SASL (Overview)**

The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. An application calls the SASL library, 
/usr/lib/libsasl.so, which provides a glue layer between the application and the various SASL mechanisms. The mechanisms are used in the authentication process and in providing optional security services. The version of SASL delivered with the Solaris 10 release is derived from the Cyrus SASL with a few changes.

SASL provides the following services:

- Loading of any plug-ins
- Determining the necessary security options from the application to aid in the choice of a security mechanism
- Listing of plug-ins that are available to the application
- Choosing the best mechanism from a list of available mechanisms for a particular authentication attempt
- Routing the authentication data between the application and the chosen mechanism
- Providing information about the SASL negotiation back to the application
SASL (Reference)

The following section provides information about the implementation of SASL for the Solaris 10 release.

SASL Plug-ins

SASL plug-ins provide support for security mechanisms, user-canonicalization, and auxiliary property retrieval. By default, the dynamically loaded 32-bit plug-ins are installed in `/usr/lib/sasl`, and the 64-bit plug-ins are installed in `/usr/lib/sasl/$ISA`. The following security mechanism plug-ins are provided in the Solaris 10 release:

- `crammd5.so.1` CRAM-MD5, which supports authentication only, no authorization
- `digestmd5.so.1` DIGEST-MD5, which supports authentication, integrity, and privacy, as well as authorization
- `gssapi.so.1` GSSAPI, which supports authentication, integrity, and privacy, as well as authorization. The GSSAPI security mechanism requires a functioning Kerberos infrastructure.
- `plain.so.1` PLAIN, which supports authentication and authorization.

In addition, the EXTERNAL security mechanism plug-in and the INTERNAL user canonicalization plug-ins are built into `libsasl.so.1`. The EXTERNAL mechanism supports authentication and authorization. The mechanism supports integrity and privacy if the external security source provides it. The INTERNAL plug-in adds the realm name if necessary to the username.

The Solaris 10 release is not supplying any auxprop plug-ins at this time. For the CRAM-MD5 and DIGEST-MD5 mechanism plug-ins to be fully operational on the server side, the user must provide an auxprop plug-in to retrieve clear text passwords. The PLAIN plug-in requires additional support to verify the password. The support for password verification can be one of the following: a callback to the server application, an auxprop plug-in, `saslauthd`, or `pwcheck`. The `saslauthd` and `pwcheck` daemons are not provided in the Solaris releases. For better interoperability, restrict server applications to those mechanisms that are fully operational by using the `mech_list` SASL option.

SASL Environment Variable

By default, the client authentication name is set to `getenv("LOGNAME")`. This variable can be reset by the client or by the plug-in.
SASL Options

The behavior of \textit{libsasl} and the plug-ins can be modified on the server side by using options that can be set in the /etc/sasl/app.conf file. The variable \textit{app} is the server-defined name for the application. The documentation for the server \textit{app} should specify the application name.

The following options are supported in the Solaris 10 release:

- \textbf{auto\_transition} Automatically transitions the user to other mechanisms when the user does a successful plain text authentication.
- \textbf{auxprop\_login} Lists the name of auxiliary property plug-ins to use.
- \textbf{canon\_user\_plugin} Selects the canon\_user plug-in to use.
- \textbf{mech\_list} Lists the mechanisms that are allowed to be used by the server application.
- \textbf{pwcheck\_method} Lists the mechanisms used to verify passwords. Currently, auxprop is the only allowed value.
- \textbf{reauth\_timeout} Sets the length of time, in minutes, that authentication information is cached for a fast reauthentication. This option is used by the DIGEST-MD5 plug-in. Setting this option to 0 disables reauthentication.

The following options are not supported in the Solaris 10 release:

- \textbf{plugin\_list} Lists available mechanisms. Not used because the option changes the behavior of the dynamic loading of plugins.
- \textbf{saslauthd\_path} Defines the location of the saslauthd door, which is used for communicating with the saslauthd daemon. The saslauthd daemon is not included in the Solaris 10 release. So, this option is also not included.
- \textbf{keytab} Defines the location of the keytab file used by the GSSAPI plug-in. Use the KRB5\_KTNAME environment variable instead to set the default keytab location.

The following options are options not found in Cyrus SASL. However, they have been added for the Solaris 10 release:

- \textbf{use\_authid} Acquire the client credentials rather than use the default credentials when creating the GSS client security context. By default, the default client Kerberos identity is used.
- \textbf{log\_level} Sets the desired level of logging for a server.
Solaris Secure Shell enables a user to securely access a remote host over an unsecured network. The shell provides commands for remote login and remote file transfer. The following is a list of topics in this chapter.

- “Solaris Secure Shell (Overview)” on page 303
- “Solaris Secure Shell Enhancements in the Solaris 10 Release” on page 306
- “Configuring Solaris Secure Shell (Task Map)” on page 307
- “Using Solaris Secure Shell (Task Map)” on page 311

For reference information, see Chapter 19.

Solaris Secure Shell (Overview)

In Solaris Secure Shell, authentication is provided by the use of passwords, public keys, or both. All network traffic is encrypted. Thus, Solaris Secure Shell prevents a would-be intruder from being able to read an intercepted communication. Solaris Secure Shell also prevents an adversary from spoofing the system.

Solaris Secure Shell can also be used as an on-demand virtual private network (VPN). A VPN can forward X Window system traffic or can connect individual port numbers between the local machines and remote machines over an encrypted network link.

With Solaris Secure Shell, you can perform these actions:

- Log in to 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 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. Version 1 is provided only to assist users who are 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 is used to represent version 2.

Solaris Secure Shell Authentication

Solaris Secure Shell provides public key and password methods for authenticating the connection to the remote host. Public key authentication is a stronger authentication mechanism than password authentication, because the private key never travels over the network.

The authentication methods are tried in the following order. When the configuration does not satisfy an authentication method, the next method is tried.

- **GSS-API** – Uses credentials for GSS-API mechanisms such as mech_krb5 (Kerberos V) and mech_dh (AUTH_DH) to authenticate clients and servers. For more information on GSS-API, see “Introduction to GSS-API” in Solaris Security for Developers Guide.

- **Host-based authentication** – Uses host keys and rhosts files. Uses the client’s RSA and DSA public/private host keys to authenticate the client. Uses the rhosts files to authorize clients to users.

- **Public key authentication** – Authenticates users with their RSA and DSA public/private keys.

- **Password authentication** – Uses PAM to authenticate users. Keyboard authentication method in v2 allows for arbitrary prompting by PAM. For more information, see the SECURITY section in the sshd(1M) man page.

The following table shows the requirements for authenticating a user who is trying to log into a remote host. The user is on the local host, the client. The remote host, the server, is running the sshd daemon. The table shows the Solaris Secure Shell authentication methods, the compatible protocol versions, and the host requirements.

<table>
<thead>
<tr>
<th>Authentication Method (Protocol Version)</th>
<th>Local Host (Client) Requirements</th>
<th>Remote Host (Server) Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSS-API (v2)</td>
<td>Initiator credentials for the GSS mechanism.</td>
<td>Acceptor credentials for the GSS mechanism. For more information, see &quot;Acquiring GSS Credentials in Solaris Secure Shell&quot; on page 324.</td>
</tr>
</tbody>
</table>
### TABLE 18–1 Authentication Methods for Solaris Secure Shell (Continued)

<table>
<thead>
<tr>
<th>Authentication Method (Protocol Version)</th>
<th>Local Host (Client) Requirements</th>
<th>Remote Host (Server) Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host-based (v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Local host private key in /etc/ssh/ssh_host_rsa_key or /etc/ssh/ssh_host_dsa_key</td>
<td>Local host public key in /etc/ssh/known_hosts or ~/.ssh/known_hosts</td>
</tr>
<tr>
<td></td>
<td>HostbasedAuthentication yes in /etc/ssh/sshd_config</td>
<td>HostbasedAuthentication yes in /etc/ssh/sshd_config</td>
</tr>
<tr>
<td></td>
<td>User account</td>
<td>IgnoreRhosts no in /etc/ssh/sshd_config</td>
</tr>
<tr>
<td></td>
<td>User's public key in ~/.ssh/authorized_keys</td>
<td>Local host entry in /etc/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.hosts</td>
</tr>
<tr>
<td>RSA or DSA public key (v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Private key in ~/.ssh/id_rsa or ~/.ssh/id_dsa</td>
<td>User's public key in ~/.ssh/authorized_keys</td>
</tr>
<tr>
<td></td>
<td>User's public key in ~/.ssh/id_rsa.pub or ~/.ssh/id_dsa.pub</td>
<td></td>
</tr>
<tr>
<td>RSA public key (v1)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Private key in ~/.ssh/identity</td>
<td>User's public key in ~/.ssh/authorized_keys</td>
</tr>
<tr>
<td></td>
<td>User's public key in ~/.ssh/identity.pub</td>
<td></td>
</tr>
<tr>
<td>Keyboard-interactive (v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Supports PAM, including arbitrary prompting and password changing when password aging is triggered.</td>
<td></td>
</tr>
<tr>
<td>Password-based (v1 or v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Supports PAM.</td>
<td></td>
</tr>
<tr>
<td>.rhosts only (v1)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>IgnoreRhosts no in /etc/ssh/sshd_config</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Local host entry in /etc/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.hosts</td>
<td></td>
</tr>
<tr>
<td>.rhosts with RSA (v1) on server only</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Local host public key in /etc/ssh/ssh_host_rsa1_key</td>
<td>Local host public key in /etc/ssh/ssh_known_hosts or ~/.ssh/known_hosts</td>
</tr>
<tr>
<td></td>
<td>IgnoreRhosts no in /etc/ssh/sshd_config</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Local host entry in /etc/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.hosts</td>
<td></td>
</tr>
</tbody>
</table>
Solaris Secure Shell in the Enterprise

For a comprehensive discussion of Secure Shell on a Solaris system, see Secure Shell in the Enterprise, by Jason Reid, ISBN 0-13-142900-0, June 2003. The book is part of the Sun BluePrints Series, which is published by Sun Microsystems Press.

For online information, navigate to Sun’s BigAdmin System Administration Portal web site, http://www.sun.com/bigadmin Click Docs, then Sun BluePrints under Misc./Comprehensive. Click Sun BluePrints OnLine, then Archives by Subject, then Security. The archives include the following articles:

- Role Based Access Control and Secure Shell – A Closer Look At Two Solaris Operating Environment Security Features
- Integrating the Secure Shell Software
- Configuring the Secure Shell Software

Solaris Secure Shell Enhancements in the Solaris 10 Release

Since the Solaris 9 release, the following changes have been introduced to Solaris Secure Shell:

- Solaris Secure Shell is based on OpenSSH 3.5p1. The Solaris implementation also includes features and bug fixes from versions up to OpenSSH 3.8p1.
- The default value of X11Forwarding is yes in the /etc/ssh/sshd_config file.
- The following keywords have been introduced:
  - GSSAPIAuthentication
  - GSSAPITKeyExchange
  - GSSAPIDelegateCredentials
  - GSSAPIStoreDelegatedCredentials
  - KbdInteractiveAuthentication

The GSSAPI keywords enable Solaris Secure Shell to use GSS credentials for authentication. The KbdInteractiveAuthentication keyword supports arbitrary prompting and password changing in PAM. For a complete list of keywords and their default values, see “Keywords in Solaris Secure Shell” on page 326.
- The ARCFOUR and AES128-CTR ciphers are now available. ARCFOUR is also known as RC4. The AES cipher is AES in counter mode.
- The sshd daemon uses the variables in /etc/default/login and the login command. The /etc/default/login variables can be overridden by values in the sshd_config file. For more information, see “Solaris Secure Shell and Login Environment Variables” on page 330 and the sshd_config(4) man page.
Solaris Secure Shell (Task Map)

The following task map points to task maps for configuring Solaris Secure Shell and for using Solaris Secure Shell.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Solaris Secure Shell</td>
<td>Guides administrators in configuring Solaris Secure Shell for users.</td>
<td>&quot;Configuring Solaris Secure Shell (Task Map)&quot; on page 307</td>
</tr>
<tr>
<td>Use Solaris Secure Shell</td>
<td>Guides users in using Solaris Secure Shell.</td>
<td>&quot;Using Solaris Secure Shell (Task Map)&quot; on page 311</td>
</tr>
</tbody>
</table>

Configuring Solaris Secure Shell (Task Map)

The following task map points to procedures for configuring Solaris Secure Shell.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure host-based authentication</td>
<td>Configures host-based authentication on the client and server.</td>
<td>&quot;How to Set Up Host-Based Authentication for Solaris Secure Shell&quot; on page 307</td>
</tr>
<tr>
<td>Configure a host to use v1 and v2</td>
<td>Creates public key files for hosts that use v1 and v2 protocols.</td>
<td>&quot;How to Enable Solaris Secure Shell v1&quot; on page 310</td>
</tr>
<tr>
<td>Configure port forwarding</td>
<td>Enables users to use port forwarding.</td>
<td>&quot;How to Configure Port Forwarding in Solaris Secure Shell&quot; on page 310</td>
</tr>
</tbody>
</table>

Configuring Solaris Secure Shell

By default, host-based authentication and the use of both protocols are not enabled in Solaris Secure Shell. Changing these defaults requires administrative intervention. Also, for port forwarding to work requires administrative intervention.

How to Set Up Host-Based Authentication for Solaris Secure Shell

The following procedure sets up a public key system where the client’s public key is used for authentication on the server. The user must also create a public/private key pair.

In the procedure, the terms client and local host refer to the machine where a user types the ssh command. The terms server and remote host refer to the machine that the client is trying to reach.
1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and
   assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in
   System Administration Guide: Basic Administration.

2 On the client, enable host-based authentication.
   In the client configuration file, /etc/ssh/ssh_config, type the following entry:
   ```
   HostbasedAuthentication yes
   ```

3 On the server, enable host-based authentication.
   In the server configuration file, /etc/ssh/sshd_config, type the same entry:
   ```
   HostbasedAuthentication yes
   ```

4 On the server, configure a file that enables the client to be recognized as a trusted host.
   For more information, see the FILES section of the sshd(1M) man page.
   - Add the client as an entry to the server's /etc/hosts.equiv file.
     ```
     client-host
     ```
   - Or, you can instruct users to add an entry for the client to their ~/.ssh/known_hosts file on the server.
     ```
     client-host
     ```

5 On the server, ensure that the sshd daemon can access the list of trusted hosts.
   Set IgnoreRhosts to no in the /etc/ssh/sshd_config file.
   ```
   # sshd_config
   IgnoreRhosts no
   ```

6 Ensure that users of Solaris Secure Shell at your site have accounts on both hosts.

7 Do one of the following to put the client's public key on the server.
   - Modify the sshd config file on the server, then instruct your users to add the client's public host
     keys to their ~/.ssh/known_hosts file.
     ```
     # sshd_config
     IgnoreUserKnownHosts no
     ```
     For user instructions, see “How to Generate a Public/Private Key Pair for Use With Solaris Secure
     Shell” on page 312.
Copy the client’s public key to the server.
The host keys are stored in the `/etc/ssh` directory. The keys are typically generated by the `sshd` daemon on first boot.

a. Add the key to the `/etc/ssh/ssh_known_hosts` file on the server.
   On the client, type the command on one line with no backslash.
   
   ```
   # cat /etc/ssh/ssh_host_dsa_key | ssh RemoteHost \n   ’cat >> /etc/ssh/ssh_known_hosts && echo “Host key copied”’
   ```

b. When you are prompted, supply your login password.
   When the file is copied, the message “Host key copied” is displayed.

**Example 18–1  Setting Up Host-based Authentication**

In the following example, each host is configured as a server and as a client. A user on either host can initiate an ssh connection to the other host. The following configuration makes each host a server and a client:

- On each host, the Solaris Secure Shell configuration files contain the following entries:

  ```
  # /etc/ssh/ssh_config
  HostBasedAuthentication yes
  #
  # /etc/ssh/sshd_config
  HostBasedAuthentication yes
  IgnoreRhosts no
  ```

- On each host, the `hosts.equiv` file contains an entry for the other host:

  ```
  # /etc/hosts.equiv on machine2
  machine1
  # /etc/hosts.equiv on machine1
  machine2
  ```

- The public key for each host is in the `/etc/ssh/ssh_known_hosts` file on the other host:

  ```
  # /etc/ssh/ssh_known_hosts on machine2
  ... machine1
  # /etc/ssh/ssh_known_hosts on machine1
  ... machine2
  ```

- Users have an account on both hosts:

  ```
  # /etc/passwd on machine1
  jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
  # /etc/passwd on machine2
  jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
## How to Enable Solaris Secure Shell v1

This procedure is useful when a host interoperates with hosts that run v1 and v2.

1. **Assume the Primary Administrator role, or become superuser.**
   
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration.*

2. **Configure the host to use both Solaris Secure Shell protocols.**
   
   Edit the `/etc/ssh/sshd_config` file.

   ```
   # Protocol 2
   Protocol 2,1
   ```

3. **Provide a separate file for the host key for v1.**
   
   Add a HostKey entry to the `/etc/ssh/sshd_config` file.

   ```
   HostKey /etc/ssh/ssh_host_rsa_key
   HostKey /etc/ssh/ssh_host_dsa_key
   HostKey /etc/ssh/ssh_host_rsa1_key
   ```

4. **Generate a host key for v1.**
   
   ```
   # ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_rsa1_key -N ''
   ```

   - `-t rsa1` Indicates the RSA algorithm for v1.
   - `-f` Indicates the file that holds the host key.
   - `-N ''` Indicates that no passphrase is required.

5. **Restart the sshd daemon.**
   
   ```
   # svcadm restart network/ssh:default
   ```

   You can also reboot the system.

## How to Configure Port Forwarding in Solaris Secure Shell

Port forwarding enables a local port be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. Similarly, a port can be specified on the remote side.

---

**Note** – Solaris Secure Shell port forwarding must use TCP connections. Solaris Secure Shell does not support UDP connections for port forwarding.
1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Configure the remote Solaris Secure Shell server to allow port forwarding.**

Change the value of AllowTcpForwarding to yes in the `/etc/ssh/sshd_config` file.

```bash
# Port forwarding
AllowTcpForwarding yes
```

3 **Restart the Solaris Secure Shell service.**

remoteHost# `svcadm restart network/ssh:default`

For information on managing persistent services, see Chapter 14, “Managing Services (Overview),” in *System Administration Guide: Basic Administration* and the `svcadm(1M)` man page.

4 **Verify that port forwarding can be used.**

remoteHost# `/usr/bin/pgrep -lf sshd`

```bash
1296 ssh -L 2001:remoteHost:23 remoteHost
```

---

**Using Solaris Secure Shell (Task Map)**

The following task map points to user procedures for using Solaris Secure Shell.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a public/private key pair</td>
<td>Enables access to Solaris Secure Shell for sites that require public-key authentication.</td>
<td>&quot;How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell&quot; on page 312</td>
</tr>
<tr>
<td>Change your passphrase</td>
<td>Changes the phrase that authenticates your private key.</td>
<td>&quot;How to Change the Passphrase for a Solaris Secure Shell Private Key&quot; on page 314</td>
</tr>
<tr>
<td>Log in with Solaris Secure Shell</td>
<td>Provides encrypted Solaris Secure Shell communication when logging in remotely. The process is similar to using the <code>rsh</code> command.</td>
<td>&quot;How to Log In to a Remote Host With Solaris Secure Shell&quot; on page 315</td>
</tr>
<tr>
<td>Log in to Solaris Secure Shell without being prompted for a password</td>
<td>Enables login by using an agent which provides your password to Solaris Secure Shell.</td>
<td>&quot;How to Reduce Password Prompts in Solaris Secure Shell&quot; on page 316</td>
</tr>
</tbody>
</table>

---

Chapter 18 • Using Solaris Secure Shell (Tasks) 311
Using Solaris Secure Shell

Solaris Secure Shell provides secure access between a local shell and a remote shell. For more information, see the ssh_config(4) and ssh(1) man pages.

How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell

Users must generate a public/private key pair when their site implements host-based authentication or user public-key authentication. For additional options, see the ssh-keygen(1) man page.

Before You Begin

Determine from your system administrator if host-based authentication is configured.

1. Start the key generation program.
   myLocalHost% ssh-keygen -t rsa
   Generating public/private rsa key pair.
   ...
   where -t is the type of algorithm, one of rsa, dsa, or rsa1.

2. Specify 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. You can select this file by pressing the Return key. Or, you can type an alternative file name.
   Enter file in which to save the key (/home/jdoe/.ssh/id_rsa): <Press Return>
   The file name of the public key is created automatically by appending the string .pub to the name of the private key file.

3. Type a passphrase for using your key.
   This passphrase is used for encrypting your private key. A null entry is strongly discouraged. Note that
   the passphrase is not displayed when you type it in.
   Enter passphrase (empty for no passphrase): <Type passphrase>
4 Retype the passphrase to confirm it.

Enter same passphrase again: <Type passphrase>
Your identification has been saved in /home/jdoe/.ssh/id_rsa.
Your public key has been saved in /home/jdoe/.ssh/id_rsa.pub.
The key fingerprint is:

5 Check the results.

Check that the path to the key file is correct.

% ls ~/.ssh
id_rsa
id_rsa.pub

At this point, you have created a public/private key pair.

6 Choose the appropriate option:

- If your administrator has configured host-based authentication, you might need to copy the local host's public key to the remote host.

You can now log in to the remote host. For details, see "How to Log In to a Remote Host With Solaris Secure Shell" on page 315.

a. Type the command on one line with no backslash.

% cat /etc/ssh/ssh_host_dsa_key | ssh RemoteHost 'cat >> ~/.ssh/known_hosts && echo "Host key copied"'

b. When you are prompted, supply your login password.

Enter password: <Type password>
Host key copied
%

- If your site uses user authentication with public keys, populate your authorized_keys file on the remote host.

a. Copy your public key to the remote host.

Type the command on one line with no backslash.

myLocalHost% cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost 'cat >> .ssh/authorized_keys && echo "Key copied"'

b. When you are prompted, supply your login password.

When the file is copied, the message "Key copied" is displayed.

Enter password: Type login password
Key copied
myLocalHost%
7  (Optional) Reduce the prompting for passphrases.
   For a procedure, see “How to Reduce Password Prompts in Solaris Secure Shell” on page 316. For more information, see the ssh-agent(1) and ssh-add(1) man pages.

Example 18–2  Establishing a v1 RSA Key for a User

In the following example, the user can contact hosts that run v1 of the Solaris Secure Shell protocol. To be authenticated by v1 hosts, the user creates a v1 key, then copies the public key portion to the remote host.

```
myLocalHost% ssh-keygen -t rsa -f /home/j doe/.ssh/identity
Generating public/private rsa key pair.
...
Enter passphrase (empty for no passphrase):  <Type passphrase>
Enter same passphrase again:  <Type passphrase>
Your identification has been saved in /home/j doe/.ssh/identity.
Your public key has been saved in /home/j doe/.ssh/identity.pub.
The key fingerprint is:
...
myLocalHost% ls ~/.ssh
id_rsa
id_rsa.pub
identity
identity.pub
myLocalHost% cat $HOME/.ssh/identity.pub | ssh myRemoteHost \ 'cat >> .ssh/authorized_keys & & echo "Key copied"'
```

\[ How to Change the Passphrase for a Solaris Secure Shell Private Key \]

The following procedure does not change the private key. The procedure changes the authentication mechanism for the private key, the passphrase. For more information, see the ssh-keygen(1) man page.

Change your passphrase.

Type the ssh-keygen command with the -p option, and answer the prompts.

```
myLocalHost% ssh-keygen -p
Enter file which contains the private key (/home/j doe/.ssh/id_rsa):  <Press Return>
Enter passphrase (empty for no passphrase):  <Type passphrase>
Enter same passphrase again:  <Type passphrase>
```

where -p requests changing the passphrase of a private key file.
How to Log In to a Remote Host With Solaris Secure Shell

1 Start a Solaris Secure Shell session.
Type the ssh command, and specify the name of the remote host.
myLocalHost% ssh myRemoteHost

A prompt questions the authenticity of the remote host:
The authenticity of host 'myRemoteHost' can't be established.
Are you sure you want to continue connecting(yes/no)?

This prompt is normal for initial connections to remote hosts.

2 If prompted, verify the authenticity of the remote host key.
- If you cannot confirm the authenticity of the remote host, type no and contact your system administrator.
  Are you sure you want to continue connecting(yes/no)? no
  The administrator is responsible for updating the global /etc/ssh/ssh_known_hosts file. An updated ssh_known_hosts file prevents this prompt from appearing.
- If you confirm the authenticity of the remote host, answer the prompt and continue to the next step.
  Are you sure you want to continue connecting(yes/no)? yes

3 Authenticate yourself to Solaris Secure Shell.
  a. When prompted, type your passphrase.
     Enter passphrase for key '/home/jdoe/.ssh/id_rsa': <Type passphrase>
  b. When prompted, type your account password.
     jdoe@myRemoteHost's password: <Type password>
     Last login: Fri Jul 20 14:24:10 2001 from myLocalHost
     myRemoteHost%

4 Conduct transactions on the remote host.
The commands that you send are encrypted. Any responses that you receive are encrypted.

5 Close the Solaris Secure Shell connection.
When you are finished, type exit or use your usual method for exiting your shell.
myRemoteHost% exit
myRemoteHost% logout
How to Reduce Password Prompts in Solaris Secure Shell

If you do not want to type your passphrase and your password to use Solaris Secure Shell, you can use the agent daemon. Start the daemon at the beginning of the session. Then, store your private keys with the agent daemon by using the `ssh-add` command. If you have different accounts on different hosts, add the keys that you need for the session.

You can start the agent daemon manually when needed, as described in the following procedure. Or, you can set the agent daemon to run automatically at the start of every session as described in "How to Set Up the ssh-agent Command to Run Automatically" on page 317.

1. **Start the agent daemon.**
   ```
   myLocalHost% ssh-agent
   ```

2. **Verify that the agent daemon has been started.**
   ```
   myLocalHost% eval `ssh-agent`
   Agent pid 9892
   ```

3. **Add your private key to the agent daemon.**
   Type the `ssh-add` command.
   ```
   myLocalHost% ssh-add
   Enter passphrase for /home/jdoe/.ssh/id_rsa: <Type passphrase>
   Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
   myLocalHost%
   ```

4. **Start a Solaris Secure Shell session.**
   ```
   myLocalHost% ssh myRemoteHost
   ```
   You are not prompted for a passphrase.

**Example 18–3 Using ssh - add Options**

In this example, jdoe adds two keys to the agent daemon. The `-l` option is used to list all keys that are stored in the daemon. At the end of the session, the `-D` option is used to remove all the keys from the agent daemon.

```
myLocalHost% ssh-agent
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa: <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
myLocalHost%
```
How to Set Up the `ssh-agent` Command to Run Automatically

You can avoid providing your passphrase and password whenever you use Solaris Secure Shell by automatically starting an agent daemon, `ssh-agent`. You can start the agent daemon from the `.dtprofile` script. To add your passphrase and password to the agent daemon, see Example 18–3.

1. **Start the agent daemon automatically in a user startup script.**
   
   Add the following lines to the end of the `$HOME/.dtprofile` script:
   ```bash
   if [ "SSH_AUTH_SOCK" = "" -a -x /usr/bin/ssh-agent ]; then
     eval '/usr/bin/ssh-agent'
   fi
   ```

2. **Terminate the agent daemon when you exit the CDE session.**
   
   Add the following lines to the `$HOME/.dt/sessions/sessionexit` script:
   ```bash
   if [ "SSH_AGENT_PID" != "" -a -x /usr/bin/ssh-agent ]; then
     /usr/bin/ssh-agent -k
   fi
   ```

   This entry ensures that no one can use the Solaris Secure Shell agent after a CDE session is terminated.

How to Use Port Forwarding in Solaris Secure Shell

You can specify that a local port 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 with IMAP4. Similarly, a port can be specified on the remote side.

**Before You Begin**

To use port forwarding, the administrator must have enabled port forwarding on the remote Solaris Secure Shell server. For details, see “How to Configure Port Forwarding in Solaris Secure Shell” on page 310.

- To use secure port forwarding, choose one of the following options:
  - To set a local port to receive secure communication from a remote port, specify both ports.
    Specify the local port that listens for remote communication. Also, specify the remote host and the remote port that forward the communication.
    ```
    myLocalHost% ssh -L localPort:remoteHost:remotePort
    ```
  - To set a remote port to receive a secure connection from a local port, specify both ports.
    Specify the remote port that listens for remote communication. Also, specify the local host and the local port that forward the communication.
    ```
    myLocalHost% ssh -R remotePort:localhost:localPort
    ```

**Example 18–4** Using Local Port Forwarding to Receive Mail

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

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

This command forwards connections from port 9143 on `myLocalHost` to port 143. Port 143 is the IMAP v2 server port on `myRemoteHost`. When the user launches a mail application, the user needs to specify the local port number, as shown in the following dialog box.
Do not confuse `localhost` in the dialog box with `myLocalHost`. `myLocalHost` is a hypothetical host name. `localhost` is a keyword that identifies your local system.

**Example 18–5 Using Remote Port Forwarding 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 command forwards connections from port 9022 on `myOutsideHost` to port 22, the `sshd` server, on the local host.

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

**How to Copy Files With Solaris Secure Shell**

The following procedure shows how to use the `scp` command to copy encrypted files between hosts. You can copy encrypted files either between a local host and a remote host, or between two remote hosts. The command operates similarly to the `rcp` command, except that the `scp` command prompts for authentication. For more information, see the `scp(1)` man page.

You can also use the `sftp`, a more secure form of the `ftp` command. For more information, see the `sftp(1)` man page.

1. **Start the secure copy program.**
   Specify the source file, the user name at the remote destination, and the destination directory.
   
   ```
   myLocalHost% scp myfile.1 jdoe@myRemoteHost:~
   ```
Supply your passphrase when prompted.

Enter passphrase for key '/home/jdoe/.ssh/id_rsa': <Type passphrase>

myfile.1  25% |********| 640 KB  0:20 ETA
myfile.1

After you type the passphrase, a progress meter is displayed. See the second line in the preceding output. The progress meter displays:

- The file name
- The percentage of the file that has been transferred
- A series of asterisks that indicate the percentage of the file that has been transferred
- The quantity of data transferred
- The estimated time of arrival, or ETA, of the complete file (that is, the remaining amount of time)

How to Set Up Default Connections to Hosts Outside a Firewall

You can use Solaris Secure Shell to make a connection from a host inside a firewall to a host outside the firewall. This task is done by specifying a proxy command for ssh either in a configuration file or as an option on the command line. For the command-line option, see Example 18–6.

In general, you can customize your ssh interactions through a configuration file.

- You can customize either your own personal file in ~/.ssh/config.
- Or, you can use the settings in the administrative configuration file, /etc/ssh/ssh_config.

The files can be customized with two types of proxy commands. One proxy command is for HTTP connections. The other proxy command is for SOCKS5 connections. For more information, see the ssh_config(4) man page.

Specify the 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
```

- Limits the proxy command specification to instances when a remote host name is specified on the command line. If you use a wildcard for outside-host, you apply the proxy command specification to a set of hosts.

- Specifies the proxy command. The command can be either of the following:
  - /usr/lib/ssh/ssh-http-proxy-connect for HTTP connections
For SOCKS5 connections

- `proxy-server` and `-p proxy-port`
  These options specify a proxy server and a proxy port, respectively. If present, the proxies override any environment variables that specify proxy servers and proxy ports, such as `HTTP_PROXY`, `HTTP_PROXYPORT`, `SOCKS5_PORT`, `SOCKS5_SERVER`, and `http_proxy`. The `http_proxy` variable specifies a URL. If the options are not used, then the relevant environment variables must be set. For more information, see the `ssh-socks5-proxy-connect(1)` and `ssh-http-proxy-connect(1)` man pages.

**outside-host**
Designates a specific host to connect to. Use the `%h` substitution argument to specify the host on the command line.

**outside-port**
Designates a specific port to connect to. Use the `%p` substitution argument to specify the port on the command line. By specifying `%h` and `%p` without using the Host `outside-host` option, the proxy command is applied to the host argument whenever the ssh command is invoked.

2 **Run Solaris Secure Shell, specifying the outside host.**

For example, type the following:

```
myLocalHost% ssh myOutsideHost
```

This command looks for a proxy command specification for `myOutsideHost` in your personal configuration file. If the specification is not found, then the command looks in the system-wide configuration file, `/etc/ssh/ssh_config`. The proxy command is substituted for the ssh command.

---

**Example 18-6 Connecting to Hosts Outside a Firewall From the Command Line**

"How to Set Up Default Connections to Hosts Outside a Firewall" on page 320 explains how to specify a proxy command in a configuration file. In this example, a proxy command is specified on the ssh command line.

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

The `-o` option to the `ssh` command provides a command-line method of specifying a proxy command. This example command does the following:

- Substitutes the HTTP proxy command for `ssh`
- Uses port `8080` and `myProxyServer` as the proxy server
- Connects to port `22` on `myOutsideHost`
Solaris Secure Shell (Reference)

This chapter describes the configuration options in Solaris Secure Shell. The following is a list of the reference information in this chapter.

- “A Typical Solaris Secure Shell Session” on page 323
- “Client and Server Configuration in Solaris Secure Shell” on page 325
- “Keywords in Solaris Secure Shell” on page 326
- “Maintaining Known Hosts in Solaris Secure Shell” on page 331
- “Solaris Secure Shell Packages and Initialization” on page 331
- “Solaris Secure Shell Files” on page 332
- “Solaris Secure Shell Commands” on page 334

For procedures to configure Solaris Secure Shell, see Chapter 18.

A Typical Solaris Secure Shell Session

The Solaris Secure Shell daemon (sshd) is normally started at boot time when network services are started. The daemon listens for connections from clients. A Solaris Secure Shell session begins when the user runs an ssh, scp, or sftp command. 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 configuration files and server-side configuration files. Command-line arguments can override the settings in the configuration files.

The client and server must authenticate themselves to each other. After successful authentication, the user can execute commands remotely and copy data between hosts.

Session Characteristics in Solaris Secure Shell

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

The behavior on the client side is controlled by Solaris Secure Shell keywords in this order of precedence:

- Command-line options
- User's configuration file, ~/.ssh/config
- System-wide configuration file, /etc/ssh/ssh_config

For example, a user can override a system-wide configuration Cipher setting of blowfish by specifying -c 3des on the command line.

**Authentication and Key Exchange in Solaris Secure Shell**

The Solaris Secure Shell protocols, v1 and v2, both support client user/host authentication and server host authentication. Both protocols involve the exchange of session cryptographic keys for the protection of Solaris Secure Shell sessions. Each protocol provides various methods for authentication and key exchange. Some methods are optional. Solaris Secure Shell supports a number of client authentication mechanisms, as shown in Table 18-1. Servers are authenticated by using known host public keys.

For the v1 protocol, Solaris Secure Shell supports user authentication with passwords. The protocol also supports user public keys and authentication with trusted host public keys. Server authentication is done with a host public key. For the v1 protocol, all public keys are RSA keys. Session key exchanges involve the use of an ephemeral server key that is periodically regenerated.

For the v2 protocol, Solaris Secure Shell supports user authentication and generic interactive authentication, which usually involves passwords. The protocol also supports authentication with user public keys and with trusted host public keys. The keys can be RSA or DSA. Session key exchanges consist of Diffie-Hellman ephemeral key exchanges that are signed in the server authentication step. Additionally, Solaris Secure Shell can use GSS credentials for authentication.

**Acquiring GSS Credentials in Solaris Secure Shell**

To use GSS-API for authentication in Solaris Secure Shell, the server must have GSS-API acceptor credentials and the client must have GSS-API initiator credentials. Support is available for mech_dh and for mech_krb5.

For mech_dh, the server has GSS-API acceptor credentials if root has run the keylogin command.

For mech_krb5, the server has GSS-API acceptor credentials when the host principal that corresponds to the server has a valid entry in /etc/krb5/krb5.keytab.

The client has initiator credentials for mech_dh if one of the following has been done:
The keylogin command has been run.
- The pam_dhkeys module is used in the pam.conf file.

The client has initiator credentials for mech_krb5 if one of the following has been done:
- The kinit command has been run.
- The pam_krb5 module is used in the pam.conf file.

For the use of mech_dh in secure RPC, see Chapter 15. For the use of mech_krb5, see Chapter 20. For more information on mechanisms, see the mech(4) and mech_spnego(5) man pages.

**Command Execution and Data Forwarding in Solaris Secure Shell**

After authentication is complete, the user can use Solaris Secure Shell, generally by requesting a shell or executing a command. Through the ssh command options, the user can make requests. Requests can include allocating a pseudo-tty, forwarding X11 connections or TCP/IP connections, or enabling an ssh-agent authentication program 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. On the server side, data is sent through the shell or a command.
2. When data transfer is complete, the user program terminates.
3. All X11 forwarding and TCP/IP forwarding is stopped, except for those connections that already exist. Existing X11 connections and TCP/IP connections remain open.
4. The server sends an exit status message to the client. When all connections are closed, such as forwarded ports that had remained open, the client closes the connection to the server. Then, the client exits.

**Client and Server Configuration in Solaris Secure Shell**

The characteristics of a Solaris Secure Shell session are controlled by configuration files. The configuration files can be overridden to a certain degree by options on the command line.

**Client Configuration in Solaris Secure Shell**

In most cases, the client-side characteristics of a Solaris Secure Shell session are governed by the system-wide configuration file, /etc/ssh/ssh_config. The settings in the ssh_config file can be overridden by the user's configuration file, ~/.ssh/config. In addition, the user can override both configuration files on the command line.
The settings in the server's `/etc/ssh/sshd_config` file determine which client requests are permitted by the server. For a list of server configuration settings, see “Keywords in Solaris Secure Shell” on page 326. For detailed information, see the `sshd_config(4)` man page.

The keywords in the client configuration file are listed in “Keywords in Solaris Secure Shell” on page 326. If the keyword has a default value, the value is given. These keywords are described in detail in the `ssh(1)`, `scp(1)`, `sftp(1)`, and `sshd_config(4)` man pages. For a list of keywords in alphabetical order and their equivalent command-line overrides, see Table 19–8.

**Server Configuration in Solaris Secure Shell**

The server-side characteristics of a Solaris Secure Shell session are governed by the `/etc/ssh/sshd_config` file. The keywords in the server configuration file are listed in “Keywords in Solaris Secure Shell” on page 326. If the keyword has a default value, the value is given. For a full description of the keywords, see the `sshd_config(4)` man page.

**Keywords in Solaris Secure Shell**

The following tables list the keywords and their default values, if any. The keywords are in alphabetical order. The location of keywords on the client is the `ssh_config` file. Keywords that apply to the server are in the `sshd_config` file. Some keywords are set in both files. If the keyword applies to only one protocol version, the version is listed.

**TABLE 19–1 Keywords in Solaris Secure Shell Configuration Files (A to Escape)**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowGroups</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>AllowTcpForwarding</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>AllowUsers</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>AuthorizedKeysFile</td>
<td>~/.ssh/authorized_keys</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Banner</td>
<td>/etc/issue</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Batchmode</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>BindAddress</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>CheckHostIP</td>
<td>yes</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>Cipher</td>
<td>blowfish, 3des</td>
<td>Client</td>
<td>v1</td>
</tr>
<tr>
<td>Ciphers</td>
<td>aes128-ctr, aes128-cbc, 3des-cbc, blowfish-cbc, arcfour</td>
<td>Both</td>
<td>v2</td>
</tr>
</tbody>
</table>
### TABLE 19–1 Keywords in Solaris Secure Shell Configuration Files (A to Escape) (Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearAllForwardings</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ClientAliveInterval</td>
<td>0</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>ClientAliveCountMax</td>
<td>3</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>Compression</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>CompressionLevel</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ConnectionAttempts</td>
<td>1</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>DenyGroups</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>DenyUsers</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>DynamicForward</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>EscapeChar</td>
<td>~</td>
<td>Client</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 19–2 Keywords in Solaris Secure Shell Configuration Files (Fall to Local)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>FallBackToRsh</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ForwardAgent</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ForwardX11</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>GatewayPorts</td>
<td>no</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>GlobalKnownHostsFile</td>
<td>/etc/ssh/ssh_known_hosts</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>GSSAPIAuthentication</td>
<td>yes</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>GSSAPIDelegateCredentials</td>
<td>no</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>GSSAPIKeyExchange</td>
<td>yes</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>GSSAPISStoreDelegateCredentials</td>
<td>no</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>Host</td>
<td>* For more information, see “Host-Specific Parameters in Solaris Secure Shell” on page 330.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>HostbasedAuthentication</td>
<td>no</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>HostbasedUsesNamesFromPacketOnly</td>
<td>no</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>HostKey</td>
<td>/etc/ssh/ssh_host_key</td>
<td>Server</td>
<td>v1</td>
</tr>
</tbody>
</table>
### Keywords in Solaris Secure Shell Configuration Files (Fall to Local)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostKey</td>
<td>/etc/ssh/host_rsa_key, /etc/ssh/host_dsa_key</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>HostKeyAlgorithms</td>
<td>ssh-rsa, ssh-dss</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>HostKeyAlias</td>
<td>No default.</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>IdentityFile</td>
<td>~/.ssh/identity</td>
<td>Client</td>
<td>v1</td>
</tr>
<tr>
<td>IdentityFile</td>
<td>~/.ssh/id_dsa, ~/.ssh/id_rsa</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>IgnoreRhosts</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>IgnoreUserKnownHosts</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>KbdInteractiveAuthentication</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>KeepAlive</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>KeyRegenerationInterval</td>
<td>3600 (seconds)</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>ListenAddress</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>LocalForward</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
</tbody>
</table>

### Keywords in Solaris Secure Shell Configuration Files (Login to R)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoginGraceTime</td>
<td>600 (seconds)</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>LogLevel</td>
<td>info</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>LookupClientHostname</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>MACs</td>
<td>hmac-sha1, hmac-md5</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>MaxAuthTries</td>
<td>6</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>MaxAuthTriesLog</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>MaxStartups</td>
<td>10:30:60</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>NoHostAuthenticationForLocalHost</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>NumberOfPasswordPrompts</td>
<td>3</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>PAMAuthenticationViaKBDInt</td>
<td>yes</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>PasswordAuthentication</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>PermitEmptyPasswords</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 19–3 Keywords in Solaris Secure Shell Configuration Files (Login to R)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>DefaultValue</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>PermitRootLogin</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>PermitUserEnvironment</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>PreferredAuthentications</td>
<td>gssapi-keyex, gssapi-with-mic, hostbased, publickey, keyboard-interactive, password</td>
<td>Client v2</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>22</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>PrintMotd</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td>2</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>ProxyCommand</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>PubkeyAuthentication</td>
<td>yes</td>
<td>Both v2</td>
<td></td>
</tr>
<tr>
<td>RemoteForward</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>RhostsAuthentication</td>
<td>no</td>
<td>Both v1</td>
<td></td>
</tr>
<tr>
<td>RhostsRSAAuthentication</td>
<td>no</td>
<td>Both v1</td>
<td></td>
</tr>
<tr>
<td>RSAAuthentication</td>
<td>no</td>
<td>Both v1</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 19–4 Keywords in Solaris Secure Shell Configuration Files (S to X)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>DefaultValue</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerKeyBits</td>
<td>768</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>StrictHostKeyChecking</td>
<td>ask</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>StrictModes</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Subsystem</td>
<td>sftp /usr/lib/ssh/sftp-server</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>SyslogFacility</td>
<td>auth</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>UseLogin</td>
<td>no Deprecated and ignored.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>UserKnownHostsFile</td>
<td>~/.ssh/known_hosts</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>VerifyReverseMapping</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>X11Forwarding</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
</tbody>
</table>
**Keywords in Solaris Secure Shell Configuration Files (StoX)**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>X11DisplayOffset</td>
<td>10</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>X11UseLocalHost</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>XAuthLocation</td>
<td>No default.</td>
<td>Both</td>
<td></td>
</tr>
</tbody>
</table>

**Host-Specific Parameters in Solaris Secure Shell**

If it is useful to have different Solaris 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 task is done by grouping entries in the file by `Host` keyword. If the `Host` keyword is not used, the entries in the client configuration file apply to whichever local host a user is working on.

**Solaris Secure Shell and Login Environment Variables**

When the following Solaris Secure Shell keywords are not set in the `sshd_config` file, they get their value from equivalent entries in the `/etc/default/login` file:

<table>
<thead>
<tr>
<th>Entry in <code>/etc/default/login</code></th>
<th>Keyword and Value in <code>sshd_config</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSOLE=*</td>
<td>PermitRootLogin=without-password</td>
</tr>
<tr>
<td>#CONSOLE=*</td>
<td>PermitRootLogin=yes</td>
</tr>
<tr>
<td>PASSREQ=YES</td>
<td>PermitEmptyPasswords=no</td>
</tr>
<tr>
<td>PASSREQ=NO</td>
<td>PermitEmptyPasswords=yes</td>
</tr>
<tr>
<td>#PASSREQ</td>
<td></td>
</tr>
<tr>
<td>TIMEOUT=&lt;secs</td>
<td>LoginGraceTime=&lt;secs</td>
</tr>
<tr>
<td>#TIMEOUT</td>
<td>LoginGraceTime=300</td>
</tr>
<tr>
<td>RETRIES and SYSLOG_FAILED_LOGINS</td>
<td>Apply only to password and keyboard-interactive authentication methods.</td>
</tr>
</tbody>
</table>

When the following variables are set by the `login` command, the `sshd` daemon uses those values. When the variables are not set, the daemon uses the default value.

**TIMEZONE** Controls the setting of the `TZ` environment variable. When not set, the `sshd` daemon uses value of `TZ` when the daemon was started.
ALTSHELL  Controls the setting of the SHELL environment variable. The default is ALTSHELL=YES, where the sshd daemon uses the value of the user's shell. When ALTSHELL=NO, the SHELL value is not set.

PATH  Controls the setting of the PATH environment variable. When the value is not set, the default path is /usr/bin.

SUPATH  Controls the setting of the PATH environment variable for root. When the value is not set, the default path is /usr/sbin:/usr/bin.

For more information, see the \login(1) and sshd(1M) man pages.

Maintaining Known Hosts in Solaris Secure Shell

Each host that needs to communicate securely with another host must have the server's public key stored in the local host's /etc/ssh/ssh_known_hosts file. Although a script could be used to update the /etc/ssh/ssh_known_hosts files, such a practice is heavily discouraged because a script opens a major security vulnerability.

The /etc/ssh/ssh_known_hosts file should only be distributed by a secure mechanism as follows:

- Over a secure connection, such as Solaris Secure Shell, IPsec, or Kerberized ftp from a known and trusted machine
- At system install time

To avoid the possibility of an intruder gaining access by inserting bogus public keys into a known_hosts file, you should use a JumpStart server as the known and trusted source of the ssh_known_hosts file. The ssh_known_hosts file can be distributed during installation. Later, scripts that use the scp command can be used to pull in the latest version. This approach is secure because each host already has the public key from the JumpStart server.

Solaris Secure Shell Packages and Initialization

Solaris Secure Shell depends on core Solaris packages and the following packages:

- SUNWgss – Contains Generic Security Service (GSS) software
- SUNWtcpd – Contains TCP wrappers
- SUNWopenssl-libraries – Contains OpenSSL libraries
- SUNWzlib – Contains the zip compression library

The following packages install Solaris Secure Shell:

- SUNWssh – Contains client files and utilities for the root (/) directory
- SUNWsshd – Contains server files and utilities for the root (/) directory
- SUNWsshcu – Contains common source files for the /usr directory
- **SUNWsshd** – Contains server files for the `/usr` directory
- **SUNWssh** – Contains client files and utilities for the `/usr` directory

Upon reboot after installation, the `sshd` daemon is running. The daemon creates host keys on the system. A Solaris system that runs the `sshd` daemon is a Solaris Secure Shell server.

## Solaris Secure Shell Files

The following table shows the important Solaris Secure Shell files and the suggested file permissions.

<table>
<thead>
<tr>
<th><strong>FILE NAME</strong></th>
<th><strong>DESCRIPTION</strong></th>
<th><strong>SUGGESTED PERMISSIONS AND OWNER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/etc/ssh/sshd_config</code></td>
<td>Contains configuration data for <code>sshd</code>, the Solaris Secure Shell daemon.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td><code>/etc/ssh/ssh_host_key</code></td>
<td>Contains the host private key (v1).</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td><code>/etc/ssh/ssh_host_dsa_key</code> or <code>/etc/ssh/ssh_host_rsa_key</code></td>
<td>Contains the host private key (v2).</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td><code>host-private-key.pub</code></td>
<td>Contains the host public key, for example, <code>/etc/ssh/ssh_host_rsa_key.pub</code>. Is used to copy the host key to the local <code>known_hosts</code> file.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td><code>/var/run/sshd.pid</code></td>
<td>Contains the process ID of the Solaris Secure Shell daemon, <code>sshd</code>. If multiple daemons are running, the file contains the last daemon that was started.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td><code>~/.ssh/authorized_keys</code></td>
<td>Holds the public keys of the user who is allowed to log in to the user account.</td>
<td>-rw-r--r-- username</td>
</tr>
<tr>
<td><code>/etc/ssh/ssh_known_hosts</code></td>
<td>Contains the host public keys for all hosts with which the client can communicate securely. The file is populated by the administrator.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td><code>~/.ssh/known_hosts</code></td>
<td>Contains the host public keys for all hosts with which the client can communicate securely. The file is maintained automatically. Whenever the user connects with an unknown host, the remote host key is added to the file.</td>
<td>-rw-r--r-- username</td>
</tr>
<tr>
<td><code>/etc/default/login</code></td>
<td>Provides defaults for the <code>sshd</code> daemon when corresponding <code>sshd_config</code> parameters are not set.</td>
<td>-r--r--r-- root</td>
</tr>
<tr>
<td><code>/etc/nologin</code></td>
<td>If this file exists, the <code>sshd</code> daemon only permits root to log in. The contents of this file are displayed to users who are attempting to log in.</td>
<td>-rw-r--r-- root</td>
</tr>
</tbody>
</table>
Table 19–5: Solaris Secure Shell Files (Continued)

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Suggested Permissions and Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>~/.rhosts</td>
<td>Contains the host-user name pairs that specify the hosts to which the user can log in without a password. This file is also used by the rlogind and rshd daemons.</td>
<td>-rw-r--r-- username</td>
</tr>
<tr>
<td>~/.ssh/hosts</td>
<td>Contains the host-user name pairs that specify the hosts to which the user can log in without a password. This file is not used by other utilities. For more information, see the sshd(1M) man page in the FILES section.</td>
<td>-rw-r--r-- username</td>
</tr>
<tr>
<td>/etc/hosts.equiv</td>
<td>Contains the hosts that are used in .rhosts authentication. This file is also used by the rlogind and rshd daemons.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td>/etc/ssh/hosts.equiv</td>
<td>Contains the hosts that are used in host-based authentication. This file is not used by other utilities.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td>~/.ssh/environment</td>
<td>Contains initial assignments at login. By default, this file is not read. The PermitUserEnvironment keyword in the sshd_config file must be set to yes for this file to be read.</td>
<td>-rw------- username</td>
</tr>
<tr>
<td>~/.ssh/rc</td>
<td>Contains initialization routines that are run before the user shell starts. For a sample initialization routine, see the sshd man page.</td>
<td>-rw------- username</td>
</tr>
<tr>
<td>/etc/ssh/sshrc</td>
<td>Contains host-specific initialization routines that are specified by an administrator.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td>/etc/ssh/ssh_config</td>
<td>Configures system settings on the client system.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td>~/.ssh/config</td>
<td>Configures user settings. Overrides system settings.</td>
<td>-rw------- username</td>
</tr>
</tbody>
</table>

The following table lists the Solaris Secure Shell files that can be overridden by keywords or command options.

Table 19–6: Overrides for the Location of Solaris Secure Shell Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Keyword Override</th>
<th>Command-Line Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/ssh/ssh_config</td>
<td></td>
<td>ssh -F config-file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>scp -F config-file</td>
</tr>
<tr>
<td>~/.ssh/config</td>
<td></td>
<td>ssh -F config-file</td>
</tr>
<tr>
<td>/etc/ssh/host_rsa_key</td>
<td>HostKey</td>
<td></td>
</tr>
<tr>
<td>/etc/ssh/host_dsa_key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File Name</td>
<td>Keyword Override</td>
<td>Command-Line Override</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>~/.ssh/identity</td>
<td>IdentityFile</td>
<td>ssh -i id-file</td>
</tr>
<tr>
<td>~/.ssh/id_dsa ~/.ssh/id_rsa</td>
<td></td>
<td>scp -i id-file</td>
</tr>
<tr>
<td>~/.ssh/authorized_keys</td>
<td>AuthorizedKeysFile</td>
<td></td>
</tr>
<tr>
<td>/etc/ssh/ssh_known_hosts</td>
<td>GlobalKnownHostsFile</td>
<td></td>
</tr>
<tr>
<td>~/.ssh/known_hosts</td>
<td>UserKnownHostsFile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IgnoreUserKnownHosts</td>
<td></td>
</tr>
</tbody>
</table>

### Solaris Secure Shell Commands

The following table summarizes the major Solaris Secure Shell commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh</td>
<td>Logs a user in to a remote machine and securely executes commands on a remote machine. This command is the Solaris Secure Shell replacement for the <code>rlogin</code> and <code>rsh</code> commands. The <code>ssh</code> command enables 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.</td>
<td>ssh(1)</td>
</tr>
<tr>
<td>ssd</td>
<td>Is the daemon for Solaris Secure Shell. The daemon listens for connections from clients and enables secure encrypted communications between two untrusted hosts over an insecure network.</td>
<td>sshd(1M)</td>
</tr>
<tr>
<td>ssh-add</td>
<td>Adds RSA or DSA identities to the authentication agent, <code>ssh-agent</code>. Identities are also called <code>keys</code>.</td>
<td>ssh-add(1)</td>
</tr>
<tr>
<td>ssh-agent</td>
<td>Holds private keys that are used for public key authentication. The <code>ssh-agent</code> program is started at the beginning of an X-session or a login session. All other windows and other programs are started as clients of the <code>ssh-agent</code> program. Through the use of environment variables, the agent can be located and used for authentication when users use the <code>ssh</code> command to log in to other systems.</td>
<td>ssh-agent(1)</td>
</tr>
<tr>
<td>ssh-keygen</td>
<td>Generates and manages authentication keys for Solaris Secure Shell.</td>
<td>ssh-keygen(1)</td>
</tr>
<tr>
<td>ssh-keyscan</td>
<td>Gathers the public keys of a number of Solaris Secure Shell hosts. Aids in building and verifying <code>ssh_known_hosts</code> files.</td>
<td>ssh-keyscan(1)</td>
</tr>
<tr>
<td>ssh-keysign</td>
<td>Is used by the <code>ssh</code> command to access the host keys on the local host. Generates the digital signature that is required during host-based authentication with Solaris Secure Shell v2. The command is invoked by the <code>ssh</code> command, not by the user.</td>
<td>ssh-keysign(1M)</td>
</tr>
</tbody>
</table>
TABLE 19–7 Commands in Solaris Secure Shell (Continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scp</td>
<td>Securely copies files between hosts on a network over an encrypted ssh transport. Unlike the rcp command, the scp command prompts for passwords or passphrases, if password information is needed for authentication. scp(1)</td>
</tr>
<tr>
<td>sftp</td>
<td>Is an interactive file transfer program that is similar to the ftp command. Unlike the ftp command, the sftp command performs all operations over an encrypted ssh transport. The command connects, logs in to the specified host name, and then enters interactive command mode. sftp(1)</td>
</tr>
</tbody>
</table>

The following table lists the command options that override Solaris Secure Shell keywords. The keywords are specified in the ssh_config and sshd_config files.

TABLE 19–8 Command-Line Equivalents for Solaris Secure Shell Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>ssh Command-Line Override</th>
<th>scp Command-Line Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchMode</td>
<td>ssh -b bind-addr</td>
<td>scp -B</td>
</tr>
<tr>
<td>BindAddress</td>
<td>ssh -b bind-addr</td>
<td>scp -a bind-addr</td>
</tr>
<tr>
<td>Cipher</td>
<td>ssh -c cipher</td>
<td>scp -c cipher</td>
</tr>
<tr>
<td>Ciphers</td>
<td>ssh -c cipher-spec</td>
<td>scp -c cipher-spec</td>
</tr>
<tr>
<td>Compression</td>
<td>ssh -C</td>
<td>scp -C</td>
</tr>
<tr>
<td>DynamicForward</td>
<td>ssh -D SOCKS4-port</td>
<td></td>
</tr>
<tr>
<td>EscapeChar</td>
<td>ssh -e escape-char</td>
<td></td>
</tr>
<tr>
<td>ForwardAgent</td>
<td>ssh -A to enable</td>
<td>scp -A</td>
</tr>
<tr>
<td></td>
<td>ssh -a to disable</td>
<td></td>
</tr>
<tr>
<td>ForwardX11</td>
<td>ssh -X to enable</td>
<td>scp -X</td>
</tr>
<tr>
<td></td>
<td>ssh -x to disable</td>
<td></td>
</tr>
<tr>
<td>GatewayPorts</td>
<td>ssh -g</td>
<td></td>
</tr>
<tr>
<td>IPv4</td>
<td>ssh -4</td>
<td>scp -4</td>
</tr>
<tr>
<td>IPv6</td>
<td>ssh -6</td>
<td>scp -6</td>
</tr>
<tr>
<td>LocalForward</td>
<td>ssh -L localport:remotehost:remoteport</td>
<td></td>
</tr>
<tr>
<td>MACS</td>
<td>ssh -m mac-spec</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>ssh -p port</td>
<td>scp -P port</td>
</tr>
</tbody>
</table>
### Solaris Secure Shell Commands

**TABLE 19–8 Command-Line Equivalents for Solaris Secure Shell Keywords**  
*(Continued)*

<table>
<thead>
<tr>
<th>Keyword</th>
<th>ssh Command-Line Override</th>
<th>scp Command-Line Override</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Protocol</strong></td>
<td>ssh -1 for v1 only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ssh -2 for v2 only</td>
<td></td>
</tr>
<tr>
<td><strong>RemoteForward</strong></td>
<td>ssh -R remotsport:localhost:localport</td>
<td></td>
</tr>
</tbody>
</table>
This section provides information on the configuration, management and use of the Kerberos service.
Introduction to the Kerberos Service

This chapter introduces the Kerberos Service. The following is a list of the overview information in this chapter.

- “What Is the Kerberos Service?” on page 339
- “How the Kerberos Service Works” on page 340
- “Kerberos Security Services” on page 346
- “The Components of Various Kerberos Releases” on page 347

What Is the Kerberos Service?

The Kerberos service is a client-server architecture that provides secure transactions over networks. The service offers strong user authentication, as well as integrity and privacy. Authentication guarantees that the identities of both the sender and the recipient of a network transaction are true. The service can also verify the validity of data being passed back and forth (integrity) and encrypt the data during transmission (privacy). Using the Kerberos service, you can log in to other machines, execute commands, exchange data, and transfer files securely. Additionally, the service provides authorization services, which allows administrators to restrict access to services and machines.

Moreover, as a Kerberos user, you can regulate other people’s access to your account.

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

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

The Kerberos service allows for flexibility in running Solaris applications. You can configure the service to allow both Kerberos-based and non-Kerberos-based requests for network services such as the NFS service, `telnet`, and `ftp`. As a result, current Solaris applications still work even if they are running on systems on which the Kerberos service is not enabled. Of course, you can also configure the Kerberos service to allow only Kerberos-based network requests.

The Kerberos service provides a security mechanism which allows the use of Kerberos for authentication, integrity, and privacy when using applications that use the Generic Security Service Application Programming Interface (GSS-API). However, applications do not have to remain committed to the Kerberos service if other security mechanisms are developed. Because the service is designed to integrate modularly into the GSS-API, applications that use the GSS-API can utilize whichever security mechanism best suits their needs.

---

How the Kerberos Service Works

The following is an overview of the Kerberos authentication system. For a more detailed description, see “How the Kerberos Authentication System Works” on page 493.

From the user’s standpoint, the Kerberos service is mostly invisible after the Kerberos session has been started. Commands such as `rsh` or `ftp` work about the same. Initializing a Kerberos session often involves no more than logging in and providing a Kerberos password.

The Kerberos system revolves around the concept of a ticket. A ticket is a set of electronic information that identifies a user or a service such as the NFS service. Just as your driver’s license identifies you and indicates what driving privileges you have, so a ticket identifies you and your network access privileges. When you perform a Kerberos-based transaction (for example, if you remote log in to another machine), you transparently send a request for a ticket to a Key Distribution Center, or KDC. The KDC accesses a database to authenticate your identity and returns a ticket that grants you permission to access the other machine. “Transparency” means that you do not need to explicitly request a ticket. The request happens as part of the `rlogin` command. Because only an authenticated client can get a ticket for a specific service, another client cannot use `rlogin` under an assumed identity.

Tickets have certain attributes associated with them. For example, a ticket can be `forwardable`, which means that it can be used on another machine without a new authentication process. A ticket can also be `postdated`, which means that it is not valid until a specified time. How tickets can be used, for example, to specify which users are allowed to obtain which types of ticket, is set by `policies`. Policies are determined when the Kerberos service 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 Kerberos” on page 494.

The following sections further explain the Kerberos 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.

The following figure shows how the initial authentication takes place.

1. At login (or with `kinit`), Client requests a TGT that allows it to obtain tickets for services.
   
   ![Diagram](image)
   
   TGT = Ticket-granting ticket
   KDC = Key Distribution Center
   
   FIGURE 20-1 Initial Authentication for a Kerberos Session

   1. A client (a user, or a service such as NFS) begins a Kerberos session by requesting a *ticket-granting ticket* (TGT) from the Key Distribution Center (KDC). This request is often done automatically at login.
A ticket-granting ticket is needed to obtain other tickets for specific services. 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.

Another analogy for the ticket-granting ticket is that of a three-day ski pass that is good at four different ski resorts. You show the pass at whichever resort you decide to go to and you receive a lift ticket for that resort, as long as the pass has not expired. Once you have the lift ticket, you can ski all you want at that resort. If you go to another resort the next day, you once again show your pass, and you get an additional lift ticket for the new resort. The difference is that the Kerberos-based commands notice that you have the weekend ski pass, and they get the lift ticket for you. So 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 by 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, such as rlogin or telnet, for as long as the ticket–granting ticket lasts. This ticket usually lasts for a few hours. Each time the client performs a unique network operation, it requests a ticket for that operation from the KDC.

**Subsequent Kerberos Authentications**

After the client has received the initial authentication, each subsequent authentication follows the pattern that is shown in the following figure.
1. Client requests ticket for server; sends TGT to KDC as proof of identity.

2. KDC sends client ticket for server.

3. Client sends ticket to server.

4. Server allows access for client.

**FIGURE 20-2 Obtaining Access to a Service Using Kerberos Authentication**

1. The client requests a ticket for a particular service, for example, to remote log in to another machine, from the KDC by 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 an NFS file system that has been shared with krb5 authentication required. Because he is already authenticated (that is, he already has a ticket-granting ticket), as he attempts to access the files, the NFS client system automatically and transparently obtains a ticket from the KDC for the NFS service.

   For example, suppose the user joe uses rlogin on the server boston. Because he is already authenticated, that is, he already has a ticket-granting ticket, he automatically and transparently obtains a ticket as part of the rlogin command. This ticket allows him to remote log in to boston as often as he wants until the ticket expires. If joe wants to remote log in to the machine denver, he obtains another ticket, as in Step 1.

3. The client sends the ticket to the server.

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

4. The server allows the client access.

TGT = Ticket-granting ticket  
KDC = Key Distribution Center
These steps make it appear that the server doesn’t ever communicate with the KDC. The server does, though; it registers itself with the KDC, just as the first client does. For simplicity's sake, that part has been left out.

The Kerberos Remote Applications

The Kerberos-based (or “Kerberized”) commands that a user such as joe can use are the following:

- ftp
- rcp
- rdist
- rlogin
- rsh
- ssh
- telnet

These applications are the same as the Solaris applications of the same name. However, they have been extended to use Kerberos principals to authenticate transactions, thereby providing Kerberos-based security. See “Kerberos Principals” on page 344 for information on principals.

These commands are discussed further in “Kerberos User Commands” on page 477.

Kerberos Principals

A client in the Kerberos service 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 or telnet.

By convention, a principal name is divided into three components: the primary, the instance, and the realm. A typical Kerberos principal would be, for example, joe/admin@ENG.EXAMPLE.COM. In this example:

- joe is the primary. The primary can be a user name, as shown here, or a service, such as nfs. The primary can also be the word host, which signifies that this principal is a service principal that is set up to provide various network services, ftp, rcp, rlogin, and so on.
- 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 the Kerberos service treats joe and joe/admin as two completely different principals.
    
In the case of a service principal, the instance is the fully qualified host name.

- bigmachine.eng.example.com is an example of such an instance. The primary/instance for this example might be ftp/bigmachine.eng.example.com or host/bigmachine.eng.example.com.
- ENG.EXAMPLE.COM is the Kerberos realm. Realms are discussed in “Kerberos Realms” on page 345.
The following are all valid principal names:

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

**Kerberos Realms**

A *realm* is a logical network, similar to a domain, that defines a group of systems under the same *master KDC*. Figure 20–3 shows how realms can relate to one another. Some realms are hierarchical, where one realm is a superset of the other realm. Otherwise, the realms are nonhierarchical (or “direct”) and the mapping between the two realms must be defined. A feature of the Kerberos service is that it permits authentication across realms. Each realm only needs to have a principal entry for the other realm in its KDC. This Kerberos feature is called *cross-realm authentication*.

**Kerberos Realms and Servers**

Each realm must include a server that maintains the master copy of the principal database. This server 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 KDC server and the slave KDC server create tickets that are used to establish authentication.
The realm can also include two additional types of Kerberos servers. A Kerberos network application server is a server that provides access to Kerberized applications (such as ftp, telnet and rsh). Realms can also include NFS servers, which provide NFS services by using Kerberos authentication. If you have installed SEAM 1.0 or 1.0.1, the realm might include a Kerberos network application server.

The following figure shows what a hypothetical realm might contain.

![Diagram of a hypothetical realm](EXAMPLE.COM)

- Master KDC
- Clients
- Slave KDCs
- Application servers

**FIGURE 20–4 A Typical Kerberos Realm**

**Kerberos Security Services**

In addition to providing secure authentication of users, the Kerberos service 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. Integrity is done through cryptographic checksumming of the data. Integrity also includes user authentication.

- **Privacy** – Privacy takes security a step further. Privacy not only includes verifying the integrity of transmitted data, but it encrypts the data before transmission, protecting it from eavesdroppers. Privacy authenticates users, as well.

Currently, of the various Kerberized applications which are part of the Kerberos service, only the ftp command allows users to change security service at runtime (“on the fly”). Developers can design their RPC-based applications to choose a security service by using the RPCSEC_GSS programming interface.
The Components of Various Kerberos Releases

Components of the Kerberos service have been included in many releases. Originally, the Kerberos service and changes to the base operating system to support the Kerberos service were released using the product name “Sun Enterprise Authentication Mechanism” which was shortened to SEAM. As more parts of the SEAM product were included in the Solaris software, the contents of the SEAM release decreased. For the Solaris 10 release, all parts of the SEAM product are included, so there is no longer a need for the SEAM product. The SEAM product name exists in the documentation for historical reasons.

The following table describes which components are included in each release. Each product release is listed in chronological order. All components are described in the following sections.

<table>
<thead>
<tr>
<th>Release Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEAM 1.0 in Solaris Easy Access Server 3.0</td>
<td>Full release of the Kerberos service for the Solaris 2.6 and 7 releases</td>
</tr>
<tr>
<td>The Kerberos service in the Solaris 8 release</td>
<td>Kerberos client software only</td>
</tr>
<tr>
<td>SEAM 1.0.1 in the Solaris 8 Admin Pack</td>
<td>Kerberos KDC and remote applications for the Solaris 8 release</td>
</tr>
<tr>
<td>The Kerberos service in the Solaris 9 release</td>
<td>Kerberos KDC and client software only</td>
</tr>
<tr>
<td>SEAM 1.0.2</td>
<td>Kerberos remote applications for the Solaris 9 release</td>
</tr>
<tr>
<td>The Kerberos service in the Solaris 10 release</td>
<td>Full release of the Kerberos service with enhancements</td>
</tr>
</tbody>
</table>

Kerberos Components

Similar to the MIT distribution of the Kerberos V5 product, the Solaris Kerberos service includes the following:

- 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, and kdestroy.
- User program for changing your Kerberos password – kpasswd.
- Remote applications – ftp, rcp, rdist, rlogin, rsh, ssh, and telnet.
Remote application daemons – ftph, rlogin, rshd, sshd, and telnetd.

Administration utilities – ktutil and kdb5_util.

The Generic Security Service Application Programming Interface (GSS-API) – Enables applications to use multiple security mechanisms without requiring you 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) – Enables NFS services to use Kerberos authentication. RPCSEC_GSS is a 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 that use RPCSEC_GSS.

Several libraries.

In addition, the Solaris Kerberos service includes the following:

- SEAM Administration Tool (gkadmin) – Enables you to administer the KDC. This Java™ technology-based GUI enables an administrator to perform the tasks that are usually performed through the kadmin command.
- The Pluggable Authentication Module (PAM) – Enables applications to use various authentication mechanisms. PAM can be used to make logins and logouts transparent to the user.
- Kernel modules – Provides kernel implementations of the GSS-API and RPCSEC_GSS APIs for NFS.

Kerberos Enhancements in the Solaris 10 Release

These Kerberos enhancements are included in the Solaris 10 Release. Several of the enhancements were introduced in prior Software Express releases and updated in the Solaris 10 Beta releases.

- Kerberos protocol support is provided in remote applications, such as ftp, rcp, rdist, rlogin, rsh, ssh, and telnet. See the man pages for each command or daemon and the krb5_auth_rules(5) man page for more information.
- The Kerberos principal database can now be transferred by incremental update instead of by transferring the entire database each time. Incremental propagation provides these advantages:
  - Increased database consistencies across servers
  - The need for fewer resources (network, CPU, and so forth)
  - Much more timely propagation of updates
  - An automated method of propagation
- A new script to help automatically configure a Kerberos client is now available. The script helps an administrator quickly and easily set up a Kerberos client. For procedures using the new script, see “Configuring Kerberos Clients” on page 382. Also, see the kclient(1M) man page for more information.
Several new encryption types have been added to the Kerberos service. These new encryption types increase security and enhance compatibility with other Kerberos implementations that support these encryption types. See “Using Kerberos Encryption Types” on page 497 for more information. The encryption types include:

- The AES encryption type can be used for high speed, high security encryption of Kerberos sessions. The use of AES is enabled through the Cryptographic Framework.
- ARCFOUR-HMAC provides better compatibility with other Kerberos implementations.
- Triple DES (3DES) with SHA1 increases security. This encryption type also enhances interoperability with other Kerberos implementations that support this encryption type.

The KDC software, the user commands, and user applications now support the use of the TCP network protocol. This enhancement provides more robust operation and better interoperability with other Kerberos implementations, including Microsoft’s Active Directory. The KDC now listens on both the traditional UDP ports as well as TCP ports so it can respond to requests using either protocol. The user commands and applications first try UDP when sending a request to the KDC, and if that fails, then try TCP.

Support for IPv6 was added to the KDC software, which includes the kinit, klist and kprop commands. Support for IPv6 addresses is provided by default. There are no configuration parameters to change to enable IPv6 support. No IPv6 support is available for the kadmin and kadmind commands.

A new -e option has been included to several subcommands of the kadmin command. This new option allows for the selection of the encryption type during the creation of principals. See the kadmin(1M) man page for more information.

Additions to the pam_krb5 module manage the Kerberos credentials cache by using the PAM framework. See the pam_krb5(5) man page for more information.

Support is provided for auto-discovery of the Kerberos KDC, admin server, kpasswd server, and host or domain name-to-realm mappings by using DNS lookups. This enhancement reduces some of the steps needed to install a Kerberos client. The client is able to locate a KDC server by using DNS instead of by reading a configuration file. See the krb5.conf(4) man page for more information.

A new PAM module called pam_krb5_migrate has been introduced. The new module helps in the automatic migration of users to the local Kerberos realm, if they do not already have Kerberos accounts. See the pam_krb5_migrate(5) man page for more information.

The ~/.k5login file can now be used with the GSS applications ftp and ssh. For more information, see the gss_auth_rules(5) man page.

The kproplog utility has been updated to output all attribute names per log entry. For more information, see the kproplog(1M) man page.

A new configuration file option makes the strict TGT verification feature optionally configurable on a per-realm basis. See the krb5.conf(4) man page for more information.

Extensions to the password-changing utilities enable the Solaris Kerberos V5 administration server to accept password change requests from clients that do not run Solaris software. See the kadmin(1M) man page for more information.
The default location of the replay cache has been moved from RAM-based file systems to persistent storage in /var/krb5/rcache/. The new location protects against replays if a system is rebooted. Performance enhancements were made to the rcache code. However, overall replay cache performance might be slower due to the use of persistent storage.

The replay cache can now be configured to use file or memory only storage. Refer to the krb5envvar(5) man page for more information about environment variables that can be configured for key table and credential cache types or locations.

The GSS credential table is no longer necessary for the Kerberos GSS mechanism. For more information, see "Mapping GSS Credentials to UNIX Credentials" on page 357 or the gsscred(1M), gssd(1M), and gsscred.conf(4) man pages.

The Kerberos utilities, kinit and ktutil, are now based on MIT Kerberos version 1.2.1. This change added new options to the kinit command and new subcommands to the ktutil command. For more information, see the kinit(1) and ktutil(1) man pages.

The Solaris Kerberos Key Distribution Center (KDC) and kadmind is now based on MIT Kerberos version 1.2.1. The KDC now defaults to a btree-based database, which is more reliable than the current hash-based database. See the kdb5_util(1M) man page for more information.

The kpropd, kadmind, krb5kdc and ktkt_warn daemons are managed by the Service Management Facility. Administrative actions on this service, such as enabling, disabling, or restarting, can be performed using the svcadm command. The service's status for all daemons can be queried using the svcsc command. For an overview of the Service Management Facility refer to Chapter 14, “Managing Services (Overview),” in System Administration Guide: Basic Administration.

Kerberos Additions for the Solaris 10 6/06 Release

In the Solaris 10 6/06 release, the ktkt_warn daemon can automatically renew credentials, rather than just warn the user when the credential is about to expire. The user must be logged in for the credential to be renewed automatically.

Kerberos Components in the Solaris 9 Release

The Solaris 9 release includes all components included in “Kerberos Components” on page 347, except for the remote applications.

SEAM 1.0.2 Components

The SEAM 1.0.2 release includes the remote applications. These applications are the only part of SEAM 1.0 that have not been incorporated into the Solaris 9 release. The components for the remote applications are as follows:

- Client applications – ftp, rcp, rlogin, rsh, and telnet
Kerberos Components in the Solaris 8 Release

The Solaris 8 release includes only the client-side portions of the Kerberos service, so many components are not included. This product enables systems that run the Solaris 8 release to become Kerberos clients without requiring you to install SEAM 1.0.1 separately. To use these capabilities, you must install a KDC that uses either Solaris Easy Access Server 3.0 or the Solaris 8 Admin Pack, the MIT distribution, or Windows 2000. The client-side components are not useful without a configured KDC to distribute tickets. The following components are included in this release:

- User programs for obtaining, viewing, and destroying tickets – kinit, klist, and kdestroy.
- User program for changing your Kerberos password – kpasswd.
- Key table administration utility – ktutil.
- Additions to the Pluggable Authentication Module (PAM) – Enables applications to use various authentication mechanisms. PAM can be used to make logins 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 components of the SEAM 1.0 release that are not already included in the Solaris 8 release. The components are as follows:

- 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.
- Remote applications – ftp, rcp, rlogin, rsh, and telnet.
- Remote application daemons – ftpd, rlogind, rshd, and telnetd.
- Administration utility – kdb5_util.
- SEAM Administration Tool (gkadmin) – Enables you to administer the KDC. This Java technology-based GUI enables an administrator to perform the tasks that are usually performed through the kadmin command.
- A preconfiguration procedure – Enables you to set the parameters for installing and configuring SEAM 1.0.1, which makes SEAM installation automatic. This procedure is especially useful for multiple installations.
- Several libraries.
SEAM 1.0 Components

The SEAM 1.0 release includes all of the items included in “Kerberos Components” on page 347 as well as the following:

- A utility (gsscred) and a daemon (gssd) – These programs help map UNIX user IDs (UIDs) to principal names. These programs are needed because NFS servers use UNIX UIDs to identify users and not principal names, which are stored in a different format.

- The Generic Security Service Application Programming Interface (GSS-API) – Enables applications to use multiple security mechanisms without requiring you 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) – Enables NFS services to use Kerberos authentication. RPCSEC_GSS is a 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 that use RPCSEC_GSS.

- A preconfiguration procedure – Enables you to set the parameters for installing and configuring SEAM 1.0, which makes installation automatic. This procedure is especially useful for multiple installations.
This chapter should be studied by administrators who are involved in the installation and maintenance of the Kerberos service. The chapter discusses several installation and configuration options that administrators must resolve before they install or configure the service.

This is a list of the topics that a system administrator or other knowledgeable support staff should study:

- “Why Plan for Kerberos Deployments?” on page 353
- “Kerberos Realms” on page 354
- “Mapping Host Names Onto Realms” on page 355
- “Client and Service Principal Names” on page 355
- “Ports for the KDC and Admin Services” on page 356
- “The Number of Slave KDCs” on page 356
- “Which Database Propagation System to Use” on page 357
- “Clock Synchronization Within a Realm” on page 358
- “Client Installation Options” on page 358
- “Kerberos Encryption Types” on page 358
- “Online Help URL in the SEAM Administration Tool” on page 359

**Why Plan for Kerberos Deployments?**

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

Deploying a Kerberos infrastructure involves such tasks as installing KDCs, creating keys for your hosts, and migrating users. Reconfiguring a Kerberos deployment can be as hard as performing an initial deployment, so plan a deployment carefully to avoid having to re-configure.
Kerberos Realms

A realm is logical network, similar to a domain, that defines a group of systems that are 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 for cross-realm authentication should be resolved before you configure the Kerberos service.

Realm Names

Realm names can consist of any ASCII string. Usually, the realm name is the same as your DNS domain name, except that the realm name is in uppercase. This convention helps differentiate problems with the Kerberos service from problems with the DNS namespace, while using a name that is familiar. If you do not use DNS or you choose to use a different string, then you can use any string. However, the configuration process requires more work. The use of 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 that you split the realm. The primary factors that determine the number of clients that can be supported are as follows:
  - The amount of Kerberos traffic that each client generates
  - The bandwidth of the physical network
  - The speed of the hosts

Because each installation will have different limitations, no rule exists for determining the maximum number of clients.

- How far apart the clients are. Setting up several small realms might make sense if the clients are in different geographic regions.

- The number of hosts that are available to be installed as KDCs. Each realm should have at least two KDC servers, one master server and one slave server.

Alignment of Kerberos realms with administrative domains is recommended. Note that a Kerberos V realm can span multiple sub-domains of the DNS domain to which the realm corresponds.

Realm Hierarchy

When you are configuring multiple realms for cross-realm authentication, you need to decide how to tie the realms together. You can establish a hierarchical relationship among the realms, which provides automatic paths to the related domains. Of course, all realms in the hierarchical chain must
be 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 realms or when no hierarchical relationship exists. The connection must be defined in the /etc/krb5/krb5.conf file on all hosts that use the connection. So, some additional work is required. For an introduction, see “Kerberos Realms” on page 345. For the configuration procedures for multiple realms, see “Configuring Cross-Realm Authentication” on page 372.

Mapping Host Names Onto Realms

The mapping of host names 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.

DNS can also be used to look up information about the KDCs. Using DNS makes it easier to change the information because you will not need to edit the krb5.conf file on all of the clients each time you make a change. See the krb5.conf(4) man page for more information.

Client and Service Principal Names

When you are using the Kerberos service, it is strongly recommended that DNS services already be configured and running on all hosts. If DNS is used, it must be enabled on all hosts 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 require that DNS is configured and use the FQDN for each host.

For the principal names that include the FQDN of a host, it is important to match the string that describes the DNS domain name in the /etc/resolv.conf file. The Kerberos service requires that the DNS domain name be in lowercase letters when you are specifying the FQDN for a principal. The DNS domain name can include uppercase and lowercase letters, but only use lowercase letters when you are creating a host principal. For example, it doesn’t matter if the DNS domain name is example.com, Example.COM, or any other variation. The principal name for the host would still be host/boston.example.com@EXAMPLE.COM.

The Kerberos service can run without DNS services. However, some key capabilities, such as the ability to communicate with 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.
In addition, the Service Management Facility has been configured so that many of the daemons or commands do not start if the DNS service is not running. The kdb5_util, kadmind, and kpropd daemons, as well as the kprop command all are configured to depend on the DNS service. To fully utilize the features available using the Kerberos service and SMF, you must configure DNS on all hosts.

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. However, if you change the port numbers, then the /etc/services and /etc/krb5/krb5.conf files must be changed on every client. In addition, the /etc/krb5/kdc.conf file on each KDC must be updated.

The Number of Slave KDCs

Slave KDCs generate credentials for clients just as the master KDC does. Slave KDCs provide backup if the master becomes 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 so, a KDC must 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. By adding more slave KDC servers, you 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 that are installed, the longer it can take to get the data updated throughout the realm. Also, because each slave retains a copy of the KDC database, more slaves increase the risk of a security breach.

In addition, one or more slave KDCs can easily be configured to be swapped with the master KDC. The advantage of configuring at least one slave KDC in this way 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 KDC. For instructions on how to configure a swappable slave KDC, see “Swapping a Master KDC and a Slave KDC” on page 395.
Mapping GSS Credentials to UNIX Credentials

The Kerberos service provides a default mapping of GSS credential names to UNIX user IDs (UIDs) for GSS applications that require this mapping, such as NFS. GSS credential names are equivalent to Kerberos principal names when using the Kerberosservice. The default mapping algorithm is to take a one component Kerberos principal name and use that component, which is the primary name of the principal, to look up the UID. The look up occurs in the default realm or any realm that is allowed by using the auth_to_local_realm parameter in /etc/krb5/krb5.conf. For example, the user principal name bob@EXAMPLE.COM is mapped to the UID of the UNIX user named bob using the password table. The user principal name bob/admin@EXAMPLE.COM would not be mapped, because the principal name includes an instance component of admin. If the default mappings for the user credentials are sufficient, the GSS credential table does not need to be populated. In past releases, populating the GSS credential table was required to get the NFS service to work. If the default mapping is not sufficient, for example if you want to map a principal name which contains an instance component, then other methods should be used. For more information see:

- "How to Create a Credential Table” on page 379
- "How to Add a Single Entry to the Credential Table" on page 379
- "How to Provide Credential Mapping Between Realms” on page 380
- "Observing Mapping from GSS Credentials to UNIX Credentials” on page 430

Automatic User Migration to a Kerberos Realm

UNIX users who do not have valid user accounts in the default Kerberos realm can be automatically migrated using the PAM framework. Specifically, the pam_krb5_migrate module would be used in the authentication stack of the PAM service. Services would be setup up so that whenever a user, who does not have a Kerberos principal, performs a successful login to a system using their password, a Kerberos principal would be automatically created for that user. The new principal would use the same password. See "Configuring Automatic Migration of Users in a Kerberos Realm” on page 392 for instructions on how to use the pam_krb5_migrate module.

Which Database Propagation System to Use

The database that is stored on the master KDC must be regularly propagated to the slave KDCs. You can configure the propagation of the database to be incremental. The incremental process propagates only updated information to the slave KDCs, rather than the entire database. For more information about database propagation, see "Administering the Kerberos Database” on page 400.

If you do not use incremental propagation, one of the first issues to resolve is how often to update the slave KDCs. The need to have up-to-date information that is available to all clients must be weighed against the amount of time it takes to complete the update.

In large installations with many KDCs in one realm, one or more slaves can propagate the data so that the process is done in parallel. This strategy reduces the amount of time that the update takes, but it also increases the level of complexity in administering the realm. For a complete description of this strategy, see "Setting Up Parallel Propagation” on page 412.
Clock Synchronization Within a Realm

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time. Known as clock skew, this feature 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 the clocks is to use the Network Time Protocol (NTP) software. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 394 for more information. Other ways of synchronizing the clocks are available, so the use of NTP is not required. However, some form of synchronization should be used to prevent access failures because of clock skew.

Client Installation Options

A new feature in the Solaris 10 release is the kclient installation utility. The utility can be run in interactive mode or noninterative mode. In interactive mode, the user is prompted for Kerberos-specific parameter values, which allows the user to make changes to the existing installation when installing the client. In noninteractive mode, a file with previously set parameter values is used. Also, command-line options can be used in the noninteractive mode. Both interactive and noninteractive modes require less steps than the manual process, which should make the process quicker and less prone to error. See “Configuring Kerberos Clients” on page 382 for a description of all the client installation processes.

Kerberos Encryption Types

An encryption type is an identifier that specifies the encryption algorithm, encryption mode, and hash algorithms used in the Kerberos service. The keys in the Kerberos service have an associated encryption type to identify the cryptographic algorithm and mode to be used when the service performs cryptographic operations with the key. Here are the supported encryption types in the Solaris 10 release:

- des-cbc-md5
- des-cbc-crc
- des3-cbc-sha1
- arcfour-hmac-md5
- arcfour-hmac-md5-exp
- aes128-cts-hmac-sha1-96

Note – In addition, the aes256-cts-hmac-sha1-96 encryption type can be used with the Kerberos service if the unbundled Strong Cryptographic packages are installed.

If you want to change the encryption type, you should do so when creating a new principal database. Because of the interaction between the KDC, the server, and the client, changing the encryption type...
on an existing database is difficult. Leave these parameters unset unless you are re-creating the
database. Refer to “Using Kerberos Encryption Types” on page 497 for more information.

Note - If you have a master KDC installed that is not running the Solaris 10 release, the slave KDCs
must be upgraded to the Solaris 10 release before you upgrade the master KDC. A Solaris 10 master
KDC will use the new encryption types, which an older slave will not be able to handle.

Online Help URL in the SEAM Administration Tool

The online help URL is used by the SEAM Administration Tool, so the URL should be defined
properly to enable the “Help Contents” menu to work. The HTML version of this manual can be
installed on any appropriate server. Alternately, you can decide to use the collections at
http://docs.sun.com

The URL is specified in the krb5.conf file when configuring a host to use the Kerberos service. The
URL should point to the section titled “SEAM Administration Tool” in the “Administering
Principals and Policies (Tasks)” chapter in this book. You can choose another HTML page, if another
location is more appropriate.
This chapter provides configuration procedures for KDC servers, network application servers, NFS servers, and Kerberos clients. Many of these procedures require superuser access, so they should be used by system administrators or advanced users. Cross-realm configuration procedures and other topics related to KDC servers are also covered.

The following topics are covered.

- “Configuring the Kerberos Service (Task Map)” on page 361
- “Configuring KDC Servers” on page 363
- “Configuring Cross-Realm Authentication” on page 372
- “Configuring Kerberos Network Application Servers” on page 375
- “Configuring Kerberos NFS Servers” on page 376
- “Configuring Kerberos Clients” on page 382
- “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 394
- “Swapping a Master KDC and a Slave KDC” on page 395
- “Administering the Kerberos Database” on page 400
- “Increasing Security on Kerberos Servers” on page 414

### Configuring the Kerberos Service (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 the Kerberos service. Other procedures are not dependent on any order, and can be done when appropriate. The following task map shows a suggested order for a Kerberos installation.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Plan for your Kerberos installation.</td>
<td>Lets you resolve configuration issues before you start the software configuration process. Planning ahead saves you time and other resources in the long run.</td>
<td>Chapter 21</td>
</tr>
</tbody>
</table>
Configuring Additional Kerberos Services (Task Map)

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. (Optional) Install NTP.</td>
<td>Configures the Network Time Protocol (NTP) software, or another clock synchronization protocol. In order for the Kerberos service to work properly, the clocks on all systems in the realm must be synchronized.</td>
<td>“Synchronizing Clocks Between KDCs and Kerberos Clients” on page 394</td>
</tr>
<tr>
<td>3. Configure the master KDC server.</td>
<td>Configures and builds the master KDC server and database for a realm.</td>
<td>“How to Configure a Master KDC” on page 363</td>
</tr>
<tr>
<td>5. (Optional) Increase security on the KDC servers.</td>
<td>Prevents security breaches on the KDC servers.</td>
<td>“How to Restrict Access to KDC Servers” on page 415</td>
</tr>
<tr>
<td>6. (Optional) Configure swappable KDC servers.</td>
<td>Makes the task of swapping the master KDC and a slave KDC easier.</td>
<td>“How to Configure a Swappable Slave KDC” on page 396</td>
</tr>
</tbody>
</table>

Configuring Additional Kerberos Services (Task Map)

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure cross-realm authentication.</td>
<td>Enables communications from one realm to another realm.</td>
<td>“Configuring Cross-Realm Authentication” on page 372</td>
</tr>
<tr>
<td>Configure Kerberos application servers.</td>
<td>Enables a server to support services such as ftp, telnet, and rsh using Kerberos authentication.</td>
<td>“Configuring Kerberos Network Application Servers” on page 375</td>
</tr>
<tr>
<td>Configure Kerberos clients.</td>
<td>Enables a client to use Kerberos services.</td>
<td>“Configuring Kerberos Clients” on page 382</td>
</tr>
<tr>
<td>Configure Kerberos NFS server.</td>
<td>Enables a server to share a file system that requires Kerberos authentication.</td>
<td>“Configuring Kerberos NFS Servers” on page 376</td>
</tr>
<tr>
<td>Increase security on an application server.</td>
<td>Increases security on an application server by restricting access to authenticated transactions only.</td>
<td>“How to Enable Only Kerberized Applications” on page 414</td>
</tr>
</tbody>
</table>
Configuring KDC Servers

After you install the Kerberos 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 the Kerberos service, so the KDCs must be installed before you attempt other tasks.

The most significant difference between a master KDC and a slave KDC is that only the master KDC 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 KDC and master KDC generate credentials. This feature provides redundancy in case the master KDC cannot respond.

▼ How to Configure a Master KDC

In this procedure, incremental propagation is configured. In addition, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- admin principal = kws/admin

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in “Online Help URL in the SEAM Administration Tool” on page 359.

Before You Begin

This procedure requires that DNS must be running. For specific naming instructions if this master is to be swappable, see "Swapping a Master KDC and a Slave KDC" on page 395.

1 Become superuser on the master KDC.

2 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.

```
kdcl # cat /etc/krb5/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM
[realms]
    EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
    }
```
Configuring KDC Servers

```
}

[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
dc = FILE:/var/krb5/kdc.log

[appdefaults]
gkadmin = {
}
```

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

---

**Note** – If you want to restrict the encryption types, you can set the `default_tkt_enctypes` or `default_tgs_enctypes` lines. Refer to “Using Kerberos Encryption Types” on page 497 for a description of the issues involved with restricting the encryption types.

---

### 3 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
[kdcrealms]
    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
    sunw_dbprop_enable = true
    sunw_dbprop_master_uloysize = 1000
}
```

In this example, the realm name definition in the `realms` section was changed. Also, in the `realms` section, lines to enable incremental propagation and to select the number of updates the KDC master keeps in the log were added.
4 Create the KDC database by using the `kdb5_util` command.

   The `kdb5_util` command creates the KDC database. Also, when used with the `-s` option, this command 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: <Typethekey>
   Re-enter KDC database master key to verify: <Typethekeyagain>
   ```

   The `-r` option followed by the realm name is not required if the realm name is equivalent to the domain name in the server's namespace.

5 Edit the Kerberos access control list file (`kadm5.acl`).

   Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC.

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

   The 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 asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `kadm5.acl(4)` man page for more information.

6 Start the `kadmind.local` command and add principals.

   The next substeps create principals that are used by the Kerberos service.

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

   a. Add administration principals to the database.

       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: <Typethepassword>
       Re-enter password for principal kws/admin@EXAMPLE.COM: <Typethepasswordagain>
       Principal "kws/admin@EXAMPLE.COM" created.
       ```

   ```
b. **Create the kprop principals.**

The kprop principal is used to authorize updates from the master KDC.

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

c. **Create a keytab file for the kadmind service.**

This command sequence creates a special keytab file with principal entries for `kadmin` and `changepw`. These principals are needed for the `kadmind` service. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 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 AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

d. **Add the kprop principal for the master KDC server to the `kadmind` keytab file.**

Adding the kprop principal to the `kadm5.keytab` file allows the `kadmind` command to authenticate itself when incremental propagation is started.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kprop/kdc1.example.com
Entry for principal kprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.

kadmin.local:

7 Start the Kerberos daemons.

    kdc1 # svcadm enable -r network/security/krb5kdc
    kdc1 # svcadm enable -r network/security/kadmin

8 Start kadmin and add more principals.

At this point, you can add principals by using the SEAM Administration Tool. To do so, you must log in with one of the admin principal names that you created earlier in this procedure. However, the following command-line example is shown for simplicity.

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

    a. Create the master KDC host principal.

    The host principal is used by Kerberized applications, such as klist and kprop. Solaris 10 clients use this principal when mounting an authenticated NFS file system. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

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

    b. (Optional) Create the kclient principal.

    This principal is used by the kclient utility during the installation of a Kerberos client. If you do not plan on using this utility, then you do not need to add the principal. The users of the kclient utility need to use this password.

    kadmin: addprinc clntconfig/admin
    Enter password for principal clntconfig/admin@example.com: <Type the password>
    Re-enter password for principal clntconfig/admin@example.com: <Type it again>
    Principal "clntconfig/admin@example.com" created.
    kadmin:
c. Add the master KDC’s host principal to the master KDC’s keytab file.
   Adding the host principal to the keytab file allows this principal to be used automatically.

   kadmin: `ktadd host/kdc1.example.com`
   Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
   with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
   mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/kdc1.example.com with kvno 3, encryption type ARCFOUR
   with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode
   with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   kadmin:

   d. Quit kadmin.
      kadmin: `quit`

9  (Optional) Synchronize the master KDCs clock by using NTP or another clock synchronization
    mechanism.
    Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be
    within the default time that is defined in the libdefaults section of the krb5.conf file for
    authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page
    394 for information about NTP.

10 Configure Slave KDCs.
    To provide redundancy, make sure to install at least one slave KDC. See “How to Configure a Slave
    KDC” on page 368 for specific instructions.

How to Configure a Slave KDC

In this procedure, a new slave KDC named kdc2 is configured. Also, incremental propagation 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
- admin principal = kws/admin

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in the
“Online Help URL in the SEAM Administration Tool” on page 359.
The master KDC must be configured. For specific instructions if this slave is to be swappable, see “Swapping a Master KDC and a Slave KDC” on page 395.

1. **On the master KDC, become superuser.**

2. **On the master KDC, start** `kadmin`.

   You must log in with one of the `admin` principal names that you created when you configured the master KDC.

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

   a. **On the master KDC, add slave host principals to the database, if not already done.**

      For the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

      ```
      kadmin: addprinc -randkey host/kdc2.example.com
      Principal "host/kdc2@EXAMPLE.COM" created.
      kadmin:
      ```

   b. **On the master KDC, create the kprop principal.**

      The kprop principal is used to authorize incremental propagation from the master KDC.

      ```
      kadmin: addprinc -randkey kprop/kdc2.example.com
      Principal "kprop/kdc2.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   c. **Quit** `kadmin`.

      ```
      kadmin: quit
      ```

3. **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
   ...
   [realms]
   EXAMPLE.COM = {
         kdc = kdc1.example.com
         kdc = kdc2.example.com
         admin_server = kdc1.example.com
   }
   ```
4 On the master KDC, add an kprop entry to kadm5.acl.
   This entry allows the master KDC to receive requests for incremental propagation for the kdc2 server.

   kdc1 # cat /etc/krb5/kadm5.acl
   */admin@EXAMPLE.COM *
   kprop/kdc2.example.com@EXAMPLE.COM p

5 On the master KDC, restart kadmind to use the new entries in the kadm5.acl file.

   kdc1 # svcadm restart network/security/kadmin

6 On all slave KDCs, copy the KDC administration files from the master KDC server.
   This step needs to be followed on all slave KDCs, because 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 KDC:
   - /etc/krb5/krb5.conf
   - /etc/krb5/kdc.conf

7 On all slave KDCs, add an entry for the master KDC and each slave KDC into the database propagation configuration file, kpropd.acl.
   This information needs to be updated on all slave KDC servers.

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

8 On all slave KDCs, make sure that the Kerberos access control list file, kadm5.acl, is not populated.
   An unmodified kadm5.acl file would look like:

   kdc2 # cat /etc/krb5/kadm5.acl
   */admin@___default_realm___ *

   If the file has kprop entries, remove them.

9 On the new slave, change an entry in kdc.conf.
   Replace the sunw_dbprop_master_ulogsize entry with an entry defining sunw_dbprop_slave_poll. The entry sets the poll time to 2 minutes.

   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
On the new slave, start the `kadmin` command.

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

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
```

kadmin:

**a. Add the slave’s host principal to the slave’s keytab file by using kadmin.**

This entry allows kprop and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: ktadd host/kdc2.example.com
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

kadmin:

**b. Add the kiprop principal to the slave KDC’s keytab file.**

Adding the kiprop principal to the krb5.keytab file allows the kpropd command to authenticate itself when incremental propagation is started.

```
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

kadmin:

**c. Quit kadmin.**

```
kadmin: quit
```
11 On the new slave, start the Kerberos propagation daemon.
   kdc2 # /usr/lib/krb5/kpropd

12 On the new slave, create a stash file by using kdb5_util.
   kdc2 # /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>

13 Kill the Kerberos propagation daemon.
   kdc2 # pkill kpropd

14 (Optional) On the new slave KDC, synchronize the master KDCs clock by using NTP or another clock
    synchronization mechanism.
    Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be
    within the default time that is defined in the libdefaults section of the krb5.conf file for
    authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page
    394 for information about NTP.

15 On the new slave, start the KDC daemon (krb5kdc).
   When the krb5kdc service is enabled, kpropd also starts if the system is configured as a slave.
   kdc2 # svcadm enable network/security/krb5kdc

### Configuring Cross-Realm Authentication

You have several ways of linking realms together so that users in one realm can be authenticated in
another realm. Normally, cross-realm authentication is accomplished by establishing a secret key
that is shared between the two realms. The relationship of the realms can be either hierarchal or
directional (see “Realm Hierarchy” on page 354).

#### How to Establish Hierarchical Cross-Realm Authentication

The example in this procedure uses 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.

**Before You Begin**

The master KDC for each realm must be configured. To fully test the authentication process, several
clients or slave KDCs must be installed.

1 Become superuser on the first master KDC.
2 Create ticket-granting ticket service principals for the two realms.

You must log in with one of the admin principal names that was created when you configured the master KDC.

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

Note – The password that is specified for each service principal must be identical in both KDCs. Thus, the password for the service principal krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM must be the same in both realms.

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

```bash
# 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, because the file is searched top down.

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

For 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.

5 Repeat all of these steps in the second realm.

## How to Establish Direct Cross-Realm Authentication

The example in this procedure 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.

**Before You Begin**

The master KDC for each realm must be configured. To fully test the authentication process, several clients or slave KDCs must be installed.

1 Become superuser on one of the master KDC servers.
2 Create ticket-granting ticket service principals for the two realms.

You must log in with one of the admin principal names that was created when you configured the master KDC.

```bash
#/usr/sbin/kadmin -p kws/admin
Enter password: <Type 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 that is specified for each service principal must be identical in both KDCs. Thus, the password for the service principal krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM must be the same in both realms.

3 Add entries in the Kerberos configuration file to define the direct path to the remote realm.

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

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

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

For 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.

5 Repeat all of these steps for the second realm.
Configuring Kerberos Network Application Servers

Network application servers are hosts that provide access using one or more of the following network applications: ftp, rcp, rlogin, rsh, and telnet. Only a few steps are required to enable the Kerberos version of these commands on a server.

▼ How to Configure a Kerberos Network Application Server

This procedure uses the following configuration parameters:

- Application server = boston
- admin principal = kws/admin
- DNS domain name = example.com
- Realm name = EXAMPLE.COM

Before You Begin

This procedure requires that the master KDC has been configured. To fully test the process, several clients must be installed.

1 Install the Kerberos client software.

2 (Optional) Install the NTP client or another clock synchronization mechanism.

   See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 394 for information about NTP.

3 Add principals for the new server and update the server’s keytab.

   The following command reports the existence of the host principal:

   ```
boston # klist -k |grep host
```

4 host/boston.example.com@EXAMPLE.COM
4 host/boston.example.com@EXAMPLE.COM
4 host/boston.example.com@EXAMPLE.COM
4 host/boston.example.com@EXAMPLE.COM

If the command does not return a principal, then create new principals using the following steps.
How to use the SEAM Administration Tool to add a principal is explained in “How to Create a New Kerberos Principal” on page 442. The example in the following steps shows how to add the required principals using the command line. You must log in with one of the admin principal names that you created when configuring the master KDC.

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

a. **Create the server’s host principal.**

```
kadmin: adadmin -randkey host/boston.example.com
Principal "host/boston.example.com" created.
kadmin:
```

b. **Add the server’s host principal to the server’s keytab.**

If the kadmin command is not running, restart it with a command similar to the following:

```
/usr/sbin/kadmin -p kws/admin
```

```
kadmin: ktadd host/boston.example.com
Entry for principal host/boston.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type Triple DES CBC mode with HMAC/SHA1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type ARCFOUR with HMAC/MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type DES CBC mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **Quit kadmin.**

```
kadmin: quit
```

**Configuring Kerberos NFS Servers**

NFS services use UNIX user IDs (UIDs) to identify a user and cannot directly use GSS credentials. To translate the credential to a UID, a credential table that maps user credentials to UNIX UIDs might need to be created. See “Mapping GSS Credentials to UNIX Credentials” on page 357 for more information on the default credential mapping. The procedures in this section focus on the tasks that are necessary to configure a Kerberos NFS server, to administer the credential table, and to initiate Kerberos security modes for NFS-mounted file systems. The following task map describes the tasks that are covered in this section.
### How to Configure Kerberos NFS Servers

In this procedure, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- NFS server = denver.example.com
- admin principal = kws/admin

1. **Complete the prerequisites for configuring a Kerberos NFS server.**
   The master KDC must be configured. To fully test the process, you need several clients.

2. **(Optional) Install the NTP client or another clock synchronization mechanism.**
   Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 394 for information about NTP.

3. **Start `kadmin`.**
   You can use the SEAM Administration Tool to add a principal, as explained in “How to Create a New Kerberos Principal” on page 442. To do so, you must log in with one of the `admin` principal names.
that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

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

a. **Create the server's NFS service principal.**

Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

Repeat this step for each unique interface on the system that might be used to access NFS data. If a host has multiple interfaces with unique names, each unique name must have its own NFS service principal.

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

kadmin:

b. **Add the server's NFS service principal to the server's keytab file.**

Repeat this step for each unique service principal created in Step a.

kadmin: `ktadd nfs/denver.example.com`
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

kadmin:

c. **Quit kadmin.**

kadmin: `quit`

4 **(Optional) Create special GSS credential maps, if needed.**

Normally, the Kerberos service generates appropriate maps between the GSS credentials and the UNIX UIDs. The default mapping is described in "Mapping GSS Credentials to UNIX Credentials" on page 357. If the default mapping is not sufficient, see "How to Create a Credential Table" on page 379 for more information.

5 **Share the NFS file system with Kerberos security modes.**

See "How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 380 for more information.
How to Create a Credential Table

The `gsscred` credential table is used by an NFS server to map Kerberos credentials to a UID. For NFS clients to mount file systems from an NFS server with Kerberos authentication, this table must be created if the default mapping is not sufficient.

1. **Edit** `/etc/gss/gsscred.conf` and **change the security mechanism**.
   Change the mechanism to `files`.

2. **Create the credential table by using the `gsscred` command**.
   ```
   # gsscred -m kerberos_v5 -a
   ```

   The `gsscred` command gathers information from all sources that are listed with the `passwd` entry in the `/etc/nsswitch.conf` file. 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.

How to Add a Single Entry to the Credential Table

**Before You Begin**
This procedure requires that the `gsscred` table has already been created on the NFS server. See “How to Create a Credential Table” on page 379 for instructions.

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

2. **Add an entry to the credential table by using the `gsscred` command**.
   ```
   # gsscred -m mech [ -n name [ -u uid ] ] -a
   ```

   - **mech** Defines the security mechanism to be used.
   - **name** Defines the principal name for the user, as defined in the KDC.
   - **uid** Defines the UID for the user, as defined in the password database.
   - **-a** Adds the UID to principal name mapping.

**Example 22–1** Adding a Multiple Component Principal to the Credential Table
In the following example, an entry is added for a principal named `sandy/admin`, which is mapped to UID 3736.

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

**Example 22–2** Adding a Principal in a Different Domain to the Credential Table
In the following example, an entry is added for a principal named `sandy/admin@EXAMPLE.COM`, which is mapped to UID 3736.
How to Provide Credential Mapping Between Realms

This procedure provides appropriate credential mapping between realms that use the same password file. In this example, the realms CORP.EXAMPLE.COM and SALES.EXAMPLE.COM use the same password file. The credentials for bob@CORP.EXAMPLE.COM and bob@SALES.EXAMPLE.COM are mapped to the same UID.

1 Become superuser.

2 On the client system, add entries to the krb5.conf file.
   
   ```shell
   # cat /etc/krb5/krb5.conf
   [libdefaults]
   .
   [realms]
   CORP.EXAMPLE.COM = {
   .
   auth_to_local_realm = SALES.EXAMPLE.COM
   .
   }
   ```

Troubleshooting See “Observing Mapping from GSS Credentials to UNIX Credentials” on page 430 to help with the process of troubleshooting credential mapping problems.

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

This procedure enables a NFS server to provide secure NFS access using different security modes or flavors. When a client negotiates a security flavor with the NFS server, the first flavor that is offered by the server that the client has access to is used. This flavor is used for all subsequent client requests of the file system shared by the NFS server.

1 Become superuser on the NFS server.
2 Verify that there is an NFS service principal in the keytab file.

The `klist` command reports if there is a keytab file and displays the principals. If the results show that no keytab file exists or that no NFS service principal exists, you need to verify the completion of all the steps in “How to Configure Kerberos NFS Servers” on page 377.

`# klist -k`

```
Keytab name: FILE:/etc/krb5/krb5.keytab
KVNO Principal
-----------------------------------------------
  3 nfs/denver.example.com@EXAMPLE.COM
  3 nfs/denver.example.com@EXAMPLE.COM
  3 nfs/denver.example.com@EXAMPLE.COM
  3 nfs/denver.example.com@EXAMPLE.COM
```

3 Enable Kerberos security modes in the `/etc/nfssec.conf` file.

Edit the `/etc/nfssec.conf` file and remove the “#” that is placed in front of the Kerberos security modes.

`# cat /etc/nfssec.conf`

```
.
.
# # Uncomment the following lines to use Kerberos V5 with NFS
# krb5 390003 kerberos_v5 default - # RPCSEC_GSS
krb5i 390004 kerberos_v5 default integrity # RPCSEC_GSS
krb5p 390005 kerberos_v5 default privacy # RPCSEC_GSS
```

4 Edit the `/etc/dfs/dfstab` file and add the `sec=` option with the required security modes to the appropriate entries.

```
share -F nfs -o sec=mode file_system
```

- `mode` Specifies the security modes to be used when sharing the file system. When using multiple security modes, the first mode in the list is used as the default.
- `file_system` Defines the path to the file system to be shared.

All clients that attempt to access files from the named file system require Kerberos authentication. To access files, the user principal on the NFS client should be authenticated.

5 Make sure that the NFS service is running on the server.

If this command is the first `share` command or set of `share` commands that you have initiated, the NFS daemons are likely not running. The following command restarts the daemons:

`# svcadm restart network/nfs/server`
6 (Optional) If the automounter is being used, edit the auto_master database to select a security mode other than the default.

You need not follow this procedure if you are not using the automounter to access the file system or if the default selection for the security mode is acceptable.

```
file_system  auto_home  -nosuid, sec=mode
```

7 (Optional) Manually issue the mount command to access the file system by using a non-default mode.

Alternatively, you could use the mount command to specify the security mode, but this alternative does not take advantage of the automounter.

```
# mount -F nfs -o sec=mode file_system
```

Example 22–3  Sharing a File System With One Kerberos Security Mode

In this example, the dfstab file line means that Kerberos authentication must succeed before any files can be accessed through the NFS service.

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5 /export/home
```

Example 22–4  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 that is listed is used on all NFS V3 clients (in this case, krb5). See the nfssec(5) man page for more information.

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5:krb5i:krb5p /export/home
```

Configuring Kerberos Clients

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

Configuring Kerberos Clients (Task Map)

The following task map includes all of the procedures associated with setting up Kerberos clients. Each row includes a task identifier, a description of why you would want to do that task, followed by a link to the task.
**Establish a Kerberos client installation profile.**
Generates a client installation profile that can be used to automatically install a Kerberos client.

**Configure a Kerberos client.**
Manually installs a Kerberos client. Use this procedure if each client installation requires unique installation parameters.

Automatically installs a Kerberos client. Use this procedure if the installation parameters for each client are the same.

Interactively installs a Kerberos client. Use this procedure if only a few of the installation parameters need to change.

**Allow a client to access a NFS file system as the root user.**
Creates a root principal on the client, so that the client can mount a NFS file system shared with root access. Also, allows for the client to set up non-interactive root access to the NFS file system, so that cron jobs can run.

### How to Create a Kerberos Client Installation Profile

This procedure creates a kclient profile that can be used when you install a Kerberos client. By using the kclient profile, you reduce the likelihood of typing errors. Also, using the profile reduces user intervention as compared to the interactive process.

1. **Become superuser.**

2. **Create a kclient installation profile.**
   A sample kclient profile could look similar to the following:
   ```
   client# cat /net/kdcl.example.com/export/install/profile
   REALM EXAMPLE.COM
   KDC kdcl.example.com
   ADMIN clntconfig
   FILEPATH /net/kdcl.example.com/export/install/krb5.conf
   NFS 1
   DNSLOOKUP none
   ```

### How to Automatically Configure a Kerberos Client

**Before You Begin**
This procedure uses an installation profile. See “How to Create a Kerberos Client Installation Profile” on page 383.

1. **Become superuser.**
2 Run the kclient installation script.

You need to provide the password for the clntconfig principal to complete the process.

```
client# /usr/sbin/kclient -p /net/kdc1.example.com/export/install/krb5.conf
```

Starting client setup

```
kdc1.example.com
```

Setting up /etc/krb5/krb5.conf.

```
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:   <Type the password>
nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.
host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.
```

Copied /net/kdc1.example.com/export/clientinstall/krb5.conf.

```
Setup COMPLETE.
```

```
client#
```

Example 22-5 Automatically Configuring a Kerberos Client With Command-Line Overrides

The following example overrides the DNSARG and the KDC parameters that are set in the installation profile.

```
# /usr/sbin/kclient -p /net/kdc1.example.com/export/install/krb5.conf
   -d dns_fallback -k kdc2.example.com
```

Starting client setup

```
kdc1.example.com
```

Setting up /etc/krb5/krb5.conf.

```
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:   <Type the password>
nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.
```
host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/kdcl.example.com/export/install/krb5.conf.

---------------------------------------------------
Setup COMPLETE.

client#

▼ How to Interactively Configure a Kerberos Client

This procedure uses the kclient installation utility without an installation profile.

1 Become superuser.

2 Run the kclient installation script.
   You need to provide the following information:
   - Kerberos realm name
   - KDC master host name
   - Administrative principal name
   - Password for the administrative principal

Example 22–6 Running the kclient Installation Utility

The following output shows the results of running the kclient command.

client# /usr/sbin/kclient
Starting client setup
---------------------------------------------------
Do you want to use DNS for kerberos lookups ? [y/n]: n
   No action performed.
Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC hostname for the above realm: kdcl.example.com

Setting up /etc/krb5/krb5.conf.

Enter the krb5 administrative principal to be used: clntconfig/admin
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM: <Type the password>
Do you plan on doing Kerberized nfs ? [y/n]: n

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Chapter 22 • Configuring the Kerberos Service (Tasks) 385
How to Manually Configure a Kerberos Client

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
- Client = client.example.com
- admin principal = kws/admin
- User principal = mre

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in the “Online Help URL in the SEAM Administration Tool” on page 359.

1 Become superuser.

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

   To change the file from the Kerberos default version, you need to change the realm names and the server names. You also need to identify 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
      }
Configuring Kerberos Clients

3 (Optional) Change the process used to locate the KDCs.

By default, the mapping of host and domain name to kerberos realm is used to locate the KDCs. You can change this behavior by adding `dns_lookup_kdc`, `dns_lookup_realm`, or `dns_fallback` to the `libdefaults` section of the `krb5.conf` file. See the `krb5.conf(4)` man page for more information.

4 (Optional) Synchronize the client’s clock with the master KDC’s clock by using NTP or another clock synchronization mechanism.

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 394 for information about NTP.

5 Start `kadmin`.

You can use the SEAM Administration Tool to add a principal, as explained in “How to Create a New Kerberos Principal” on page 442. To do so, you must log in with one of the `admin` principal names.
that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

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

a. (Optional) Create a user principal if a user principal does not already exist.
   You need to create a user principal only if the user associated with this host does not already have a principal assigned to him or her.
   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:

b. (Optional) Create a root principal.
   If the client does not require root access to a remote file system which is mounted using the NFS service, then you can skip this step. The root principal should be a two component principal with the second component the host name of the Kerberos client system to avoid the creation of a realm wide root principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.
   kadmin: addprinc -randkey root/client.example.com
   Principal "root/client.example.com" created.
   kadmin:

c. Create a host principal.
   The host principal is used to authenticate applications.
   kadmin: addprinc -randkey host/denver.example.com
   Principal "host/denver.example.com@EXAMPLE.COM" created.
   kadmin:

d. (Optional) Add the server’s NFS service principal to the server’s keytab file.
   This step is only required if the client needs to access NFS file systems using Kerberos authentication.
   kadmin: ktadd nfs/denver.example.com
   Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal nfs/denver.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   kadmin:
e. **(Optional) Add the root principal to the server’s keytab file.**

   This step is required if you added a root principal so that the client can have root access to file systems mounted using the NFS service. This step is also required if non-interactive root access is needed, such as running cron jobs as root.

   

   kadmin: `ktadd root/client.example.com`

   Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.

   Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   Entry for principal root/client.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   kadmin:

   f. **Add the host principal to the server’s keytab file.**

   

   kadmin: `ktadd host/denver.example.com`

   Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.

   Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   Entry for principal host/denver.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

   kadmin:

   g. **Quit kadmin.**

   kadmin: `quit`

6 **(Optional) To use Kerberos with NFS, enable Kerberos security modes in the /etc/nfssec.conf file.**

   Edit the /etc/nfssec.conf file and remove the “#” that is placed in front of the Kerberos security modes.

   

   # cat /etc/nfssec.conf

   .

   .

   #

   # Uncomment the following lines to use Kerberos V5 with NFS

   #

   krb5 390003 kerberos_v5 default - # RPCSEC_GSS

   krb5i 390004 kerberos_v5 default integrity # RPCSEC_GSS

   krb5p 390005 kerberos_v5 default privacy # RPCSEC_GSS

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

   See the warn.conf(4) man page for more information.
### Example 22-7  Setting Up a Kerberos Client Using a Non-Kerberos KDC

A Kerberos client can be set up to work with a non-Kerberos KDC. In this case, a line must be included in the `/etc/krb5/krb5.conf` file in the `realms` section. This line changes the protocol that is used when the client is communicating with the Kerberos password-changing server. The format of this line follows.

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

### Example 22-8  DNS TXT Records for the Mapping of Host and Domain Name to Kerberos Realm

```plaintext
@ IN SOA kdc1.example.com root.kdc1.example.com
    1989020501 ;serial
    10800 ;refresh
    3600 ;retry
    3600000 ;expire
    86400 ;minimum

IN NS kdc1.example.com.
kdc1 IN A 192.146.86.20
kdc2 IN A 192.146.86.21

_kerberos.example.com. IN TXT "EXAMPLE.COM"
_kerberos.kdc1.example.com. IN TXT "EXAMPLE.COM"
_kerberos.kdc2.example.com. IN TXT "EXAMPLE.COM"
```

### Example 22-9  DNS SRV Records for Kerberos Server Locations

This example defines the records for the location of the master KDC, the `admin` server, and the `kpasswd` servers.

```plaintext
@ IN SOA kdc1.example.com root.kdc1.example.com
    1989020501 ;serial
    10800 ;refresh
    3600 ;retry
    3600000 ;expire
    86400 ;minimum

IN NS kdc1.example.com.
kdc1 IN A 192.146.86.20
kdc2 IN A 192.146.86.21
```
How to Access a Kerberos Protected NFS File System as the root User

This procedure allows a client to access a NFS file system that requires Kerberos authentication with the root ID privilege. In particular, if the NFS file system is shared with options like: `-o sec=krb5,root=client1.sun.com`.

1. Become superuser.

2. Start `kadmin`.

   You can use the SEAM Administration Tool to add a principal, as explained in "How to Create a New Kerberos Principal" on page 442. To do so, you must log in with one of the admin principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

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

   1. Create a root principal for the NFS client.
      
      This principal is used to provide root equivalent access to NFS mounted file systems that require Kerberos authentication. The root principal should be a two component principal with the second component the host name of the Kerberos client system to avoid the creation of a realm wide root principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

      ```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
```

   2. Add the root principal to the server’s keytab file.
      
      This step is required if you added a root principal so that the client can have root access to file systems mounted using the NFS service. This step is also required if non-interactive root access is needed, such as running cron jobs as root.

      ```
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ARCFOUR
```
with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode
with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

kadmin:

c. Quit kadmin.
   
kadmin: quit

▼ Configuring Automatic Migration of Users in a Kerberos Realm

Users, who do not have a Kerberos principal, can be automatically migrated to an existing Kerberos realm. The migration is achieved by using the PAM framework for the service in use by stacking the pam_krb5_migrate module in the service's authentication stack in /etc/pam.conf.

In this example, the rlogin and other PAM service names are configured to use the automatic migration. The following configuration parameters are used:

- Realm name = EXAMPLE.COM
- Master KDC = kdc1.example.com
- Machine hosting the migration service = server1.example.com
- Migration service principal = host/server1.example.com

Before You Begin

Setup server1 as a Kerberos client of the realm EXAMPLE.COM. See “Configuring Kerberos Clients” on page 382 for more information.

1 Check to see if a host service principal for server1 exists.
   The host service principal in the keytab file of server1 is used to authenticate the server to the master KDC.
   
server1 # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   
   KVNO Principal
   --- ------------------------------------------------
   3 host/server1.example.com@EXAMPLE.COM
   3 host/server1.example.com@EXAMPLE.COM
   3 host/server1.example.com@EXAMPLE.COM
   3 host/server1.example.com@EXAMPLE.COM

2 Make changes to the PAM configuration file.
   Add the pam_krb5_migrate PAM module to the authentication stack for the rlogin and other service names. Any user using rlogin, telnet, or ssh, without a Kerberos principal, would automatically have a principal created for them.
   
   # cat /etc/pam.conf
   
   .
   .
   .
   .
# rlogin service (explicit because of pam_rhost_auth)
#
# rlogin auth sufficient pam_rhosts_auth.so.1
# rlogin auth requisite pam_authtok_get.so.1
# rlogin auth required pam_dhkeys.so.1
# rlogin auth required pam_unix_cred.so.1
# rlogin auth required pam_unix_auth.so.1
# rlogin auth sufficient pam_krb5.so.1
# rlogin auth optional pam_krb5_migrate.so.1
#
# Other auth definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
# other auth requisite pam_authtok_get.so.1
# other auth required pam_dhkeys.so.1
# other auth required pam_unix_cred.so.1
# other auth required pam_unix_auth.so.1
# other auth sufficient pam_krb5.so.1
# other auth optional pam_krb5_migrate.so.1

3 (Optional) Force an immediate password change, if needed.

The newly created Kerberos accounts can have their password expiration time set to the current time (now), in order to force an immediate Kerberos password change. To set the expiration time to now add the expire_pw option to the lines which use the pam_krb5_migrate module. See the pam_krb5_migrate(5) man page for more information.

```bash
# cat /etc/pam.conf

rlogin auth optional pam_krb5_migrate.so.1 expire_pw
#

other auth optional pam_krb5_migrate.so.1 expire_pw
```

4 On the master KDC, update the access control file.

The following entries grant migrate and inquire privileges to the host/server1.example.com service principal for all users, excepting the root user. It is important that users who should not be migrated are listed in the kadm5.acl file using the U privilege. These entries need to be before the permit all or ui entry. See the kadm5.acl(4) man page for more information.

```bash
kdc1 # cat /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```
5 On the master KDC, restart the Kerberos administration daemon.

This step allows the kadm5 daemon to use the new kadm5.acl entries.

```
kdc1 # svcadm restart network/security/kadmind
```

6 On the master KDC, add entries to the `pam.conf` file.

The following entries enable the kadm5 daemon to use the k5migrate PAM service, to validate UNIX user password for accounts that require migration.

```
# grep k5migrate /etc/pam.conf
k5migrate auth required pam_unix_auth.so.1
k5migrate account required pam_unix_account.so.1
```

## Synchronizing Clocks Between KDCs and Kerberos Clients

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as clock skew). This requirement provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, client requests are 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). You can change this default in the `libdefaults` section of the `krb5.conf` file.

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

Because maintaining synchronized clocks between the KDCs and Kerberos clients is important, you should use the Network Time Protocol (NTP) software to synchronize them. 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, a process that can be less involved than using NTP. However, this section focuses 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 or network clock synchronization, or both, in a network environment. NTP is basically a server-client implementation. You pick one system to be the master clock (the NTP server). Then, you set up all your other systems (the NTP clients) to synchronize their clocks with the master clock.
To synchronize the clocks, NTP uses the xntpd daemon, which sets and maintains a UNIX system time-of-day in agreement with Internet standard time servers. The following shows an example of this server-client NTP implementation.

Ensuring that the KDCs and Kerberos clients maintain synchronized clocks involves implementing the following steps:

1. Setting up an NTP server on your network. This server can be any system, except the master KDC. See “Managing Network Time Protocol (Tasks)” in System Administration Guide: Network Services to find the NTP server task.

2. As you configure the KDCs and Kerberos clients on the network, setting them up to be NTP clients of the NTP server. See “Managing Network Time Protocol (Tasks)” in System Administration Guide: Network Services to find the NTP client task.

Swapping a Master KDC and a Slave KDC

You should use the procedures in this section to make the swap of a master KDC with a slave KDC easier. You should swap the master KDC with a slave KDC only if the master KDC server fails for some reason, or if the master KDC needs to be re-installed (for example, because new hardware is installed).
Swapping a Master KDC and a Slave KDC

How to Configure a Swappable Slave KDC

Perform this procedure on the slave KDC server that you want to have available to become the master KDC. This procedure assumes that you are using incremental propagation.

1 Use alias names for the master KDC and the swappable slave KDC during the KDC installation.
When you define the host names for the KDCs, make sure that each system has an alias included in DNS. Also, use the alias names when you define the hosts in the /etc/krb5/krb5.conf file.

2 Follow the steps to install a slave KDC.
Prior to any swap, this server should function as any other slave KDC in the realm. See "How to Configure a Slave KDC" on page 368 for instructions.

3 Move the master KDC commands.
To prevent the master KDC commands from being run from this slave KDC, move the kprop, kadmind, and kadmin.local commands 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

How to Swap a Master KDC and a Slave KDC

In this procedure, the master KDC server that is being swapped out is named kdc1. The slave KDC that will become the new master KDC is named kdc4. This procedure assumes that you are using incremental propagation.

Before You Begin
This procedure requires that the slave KDC server has been set up as a swappable slave. For more information, see "How to Configure a Swappable Slave KDC" on page 396.

1 On the new master KDC, start kadmin.

   kdc4 # /usr/sbin/kadmin -p kws/admin

   Enter password: <Type kws/admin password>

   kadmin:

   a. Create new principals for the kadmind service.

   The following example shows the first addprinc command on two lines, but it should be typed on one line.

   kadmin: addprinc -randkey -allow_tgs_req +password_changing_service -clearpolicy \
   changepw/kdc4.example.com

   Principal "changepw/kdc4.example.com@ENG.SUN.COM" created.

   kadmin: addprinc -randkey -allow_tgs_req -clearpolicy kadmin/kdc4.example.com

   Principal "kadmin/kdc4.example.com@EXAMPLE.COM" created.

   kadmin:
b. Create a keytab file.

kadmin: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc4.example.com
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

kadmin: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc4.example.com
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

kadmin: quit

2 On the new master KDC, force synchronization.

The following steps force a full KDC update on the slave server.
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
kdc4 # svcadm enable network/security/krb5kdc

3 On the new master KDC, clear the update log.

These steps reinitialize the update log for the new master KDC server.
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog

4 On the old master KDC, kill the kadmin and krb5kdc processes.

When you kill the kadmin process, you prevent any changes from being made to the KDC database.
kdc1 # svcadm disable network/security/kadmin
kdc1 # svcadm disable network/security/krb5kdc

5 On the old master KDC, specify the poll time for requesting propagations.

Replace the sunw_dbprop_master ulogsize entry in /etc/krb5/kdc.conf with an entry defining sunw_dbprop_slave_poll. The entry sets the poll time to 2 minutes.
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
Swapping a Master KDC and a Slave KDC

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
    sunw_dbprop_enable = true
    sunw_dbprop_slave_poll = 2m
}

6 On the old master KDC, move the master KDC commands and the `kadm5.acl` file.
   To prevent the master KDC commands from being run, move the kprop, kadmind, and kadmin.local commands to a reserved place.
   kdc1 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
   kdc1 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
   kdc1 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
   kdc1 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save

7 On the DNS server, change the alias names for the master KDC.
   To change the servers, edit the example.com zone file and change the entry for masterkdc.
   masterkdc IN CNAME kdc4

8 On the DNS server, restart the Internet domain name server.
   Run the following command to reload the new alias information:
   # svcadm refresh network/dns/server

9 On the new master KDC, move the master KDC commands and the slave `kpropd.acl` file.
   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
   kdc4 # mv /etc/krb5/kpropd.acl /etc/krb5/kpropd.acl.save

10 On the new master KDC, create the Kerberos access control list file (`kadm5.acl`).
    Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC. The file should also list all of the slaves that make requests for incremental propagation. See the `kadm5.acl(4)` man page for more information.
    kdc4 # cat /etc/krb5/krb5.krb
    kws/admin@EXAMPLE.COM *
    kprop/kdc1.example.com@EXAMPLE.COM p
11. On the new master KDC, specify the update log size in the kdc.conf file.

   Replace the sunw_dbprop_slave_poll entry with an entry defining
   sunw_dbprop_master_uologsize. The entry sets the log size to 1000 entries.
   
   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
   sunw_dbprop_enable = true
   sunw_dbprop_master_uologsize = 1000
   }

12. On the new master KDC, add the kprop principal to the kadmind keytab file.

   kdc4 # kadmin.local
   kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kprop/kdc4.example.com
   Entry for principal kprop/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
   with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
   Entry for principal kprop/kdc4.example.com with kvno 3, encryption type Triple DES cbc
   mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
   Entry for principal kprop/kdc4.example.com with kvno 3, encryption type ARCFOUR
   with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
   Entry for principal kprop/kdc4.example.com with kvno 3, encryption type DES cbc mode
   with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
   kadmin.local: quit

13. On the new master KDC, start kadmind and krb5kdc.

   kdc4 # svcadm enable network/security/krb5kdc
   kdc4 # svcadm enable network/security/kadmin

14. On the old master KDC, add the kprop service principal.

   Adding the kprop principal to the krb5.keytab file allows the kpropd daemon to authenticate itself
   for the incremental propagation service.

   kdc1 # /usr/sbin/kadmin -p kws/admin
   Authenticating as principal kws/admin@EXAMPLE.COM with password.
   Enter password: <Type kws/admin password>
   kadmin: ktadd kprop/kdc1.example.com
   Entry for principal kprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
   with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Administering the Kerberos Database

Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab "WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab "WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab "WRFILE:/etc/krb5/krb5.keytab.

kadmin: quit

15 On the old master KDC, add an entry for each KDC listed in krb5.conf to the propagation configuration file, kpropd.acl.

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
host/kdc4.example.com@EXAMPLE.COM
```

16 On the old master KDC, start kpropd and krb5kdc.

When the krb5kdc daemon is started, kpropd also starts if the system is configured as a slave.

```
kdc1 # svcadm enable network/security/krb5kdc
```

Administering the Kerberos Database

The Kerberos database is the backbone of Kerberos and must be maintained properly. This section provides some procedures on how to administer the Kerberos database, such as backing up and restoring the database, setting up incremental or parallel propagation, and administering the stash file. The steps to initially set up the database are in "How to Configure a Master KDC" on page 363.

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 the slave KDCs will lose synchronization. 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 KDC for purposes of load balancing, the clients that use that slave KDC as a master KDC will not have the latest information. Therefore, you must make sure that propagation occurs often enough or else configure the servers for incremental propagation, based on how often you change the Kerberos database. Incremental propagation is preferred over manual propagation because there is more administrative overhead when you manually propagate the database. Also, there are inefficiencies when you do full propagation of the database.

When you configure the master KDC, you set up the kprop_script command in a cron job to automatically back up the Kerberos database to the /var/krb5/slave_data/trans dump file and propagate it to the slave KDCs. But, as with any file, the Kerberos database can become corrupted. If
data corruption occurs on a slave KDC, you might never notice, because the next automatic propagation of the database installs a fresh copy. However, if corruption occurs on the master KDC, the corrupted database is propagated to all of the slave KDCs 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: 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 and write permissions only as root. To protect against unauthorized access, use only the kprop command to propagate the database dump file, which encrypts the data that is being transferred. Also, kprop propagates the data only to the slave KDCs, which minimizes the chance of accidentally sending the database dump file 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 KDC slaves will not contain the updates. The updates will be lost. For this reason, if you add significant updates to the Kerberos database before a regularly scheduled propagation, you should manually propagate the database to avoid data loss.

---

**The kpropd.acl File**

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

However, the Kerberos installation and subsequent configuration steps in this book instruct you to add the same kpropd.acl file to the master KDC and the slave KDCs. This file contains all the KDC host principal names. This configuration enables you to propagate from any KDC, in case the propagating KDCs become temporarily unavailable. And, by keeping an identical copy on all KDCs, you make the configuration easy to maintain.

**The kprop_script Command**

The kprop_script command uses the kprop command to propagate the Kerberos database to other KDCs. If the kprop_script command is run on a slave KDC, it propagates the slave KDC’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 kprop_script is run, it creates a backup of the Kerberos database to the /var/krb5/slave_datatrans 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 the `kdb5_util` command.

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

   `- verbose` Prints the name of each principal and policy that is being backed up.

   `-d dbname` Defines the name of the database to back up. Note that you can specify an absolute path for the file. If the `-d` option is not specified, the default database name is `/var/krb5/principal`.

   `filename` Defines the file that is used to back up the database. You can specify an absolute path for the file. If you don't specify a file, the database is dumped to standard output.

   `principals` Defines 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 any principals, the entire database is backed up.

Example 22–10  Backing Up the Kerberos Database

In the following example, the Kerberos database is backed up 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
```

In the following example, the `pak` and `pak/admin` principals from the Kerberos database are backed up.

```
# kdb5_util dump -verbose dumpfile 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 the `kdb5_util` command.

   ```
   # /usr/sbin/kdb5_util load [-verbose] [-d dbname] [-update] [filename]
   ```
Administering the Kerberos Database

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

dbname
Defines the name of the database to restore. Note you can specify an absolute path for
the file. If the -d option is not specified, the default database name is
/var/krb5/principal.

-update
Updates the existing database. Otherwise, a new database is created or the existing
database is overwritten.

filename
Defines the file from which to restore the database. You can specify an absolute path
for the file.

Example 22–11 Restoring the Kerberos Database

In the following example, the database called database1 is restored into the current directory from
the dumpfile file. Because the -update option isn’t specified, a new database is created by the restore.

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

How to Reload a Kerberos Database

If your KDC database was not created on a server running the Solaris 10 release, reloading the
database allows you to take advantage of the improved database format.

Before You Begin

Make sure that the database is using an older format. See for specific instructions.

1 On the master, stop the KDC daemons.

```
kdc1 # svcadm disable network/security/krb5kdc
kdc1 # svcadm disable network/security/kadmin
```

2 Dump the KDC database.

```
kdc1 # kdb5_util dump /tmp/prdb.txt
```

3 Save copies of the current database files.

```
kdc1 # cd /var/krb5
kdc1 # mkdir old
kdc1 # mv princ* old/
```

4 Load the database.

```
kdc1 # kdb5_util load /tmp/prdb.txt
```

5 Start the KDC daemons.

```
kdc1 # svcadm enable network/security/krb5kdc
kdc1 # svcadm enable network/security/kadmin
```
How to Reconfigure a Master KDC to Use Incremental Propagation

The steps in this procedure can be used to reconfigure an existing master KDC to use incremental propagation. 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 principal = kws/admin

1 Add entries to kdc.conf.
You need to enable incremental propagation and select the number of updates the KDC master keeps in the log. See the kdc.conf(4) man page for more information.

```
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 
    sunw_dbprop_enable = true 
    sunw_dbprop_master_uologsize = 1000 
}
```

2 Create the kiprop principal.
The kiprop principal is used to authenticate the master KDC server and to authorize updates from the master KDC.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
kadmin: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```
3 **Add the kprop principal to the kadmind keytab file**

Adding the kprop principal to the kadmind keytab file allows the kadmind command to authenticate itself when it is started.

```
admin: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc1.example.com
```

Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab

```
WRFILE:/etc/krb5/kadm5.keytab.
```

Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab

```
WRFILE:/etc/krb5/kadm5.keytab.
```

Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab

```
WRFILE:/etc/krb5/kadm5.keytab.
```

Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab

```
WRFILE:/etc/krb5/kadm5.keytab.
```

```
admin: quit
```

4 **(Optional) On the master KDC, add a kprop entry to kpropd.acl**

This entry allows the master KDC to receive requests for incremental propagation for the kdc2 server.

```
kdc1 # cat /etc/krb5/kpropd.acl
```

```
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

5 **Comment out the kprop line in the root crontab file.**

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

```
kdc1 # crontab -e
```

```
#ident "@(#)root 1.20 01/11/06 SMI"
#
# 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 * * * /usr/sbin/logadm
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 kdc2.example.sun.com #SUNWkr5ma
```

6 **Restart kadmind.**

```
kdc1 # svcadm restart network/security/kadmin
```

7 **Reconfigure all slave KDC servers that use incremental propagation.**
How to Reconfigure a Slave KDC to Use Incremental Propagation

1. **Add entries to krb5.conf.**
   The new entries enable incremental propagation and set the poll time to 2 minutes.
   ```
   kdc2 # 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
   sunw_dbprop_enable = true
   sunw_dbprop_slave_poll = 2m
   }
   ```

2. **Add the kprop principal to the krb5.keytab file.**
   ```
   kdc2 # /usr/sbin/kadmin -p kws/admin
   Enter password:  
   <Type kws/admin password>

   kadmin: ktadd kprop/kdc2.example.com

   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   
   kadmin: quit
   ```

3. **Disable kpropd.**
   ```
   kdc2 # svcadm disable network/security/krb5_prop
   ```

4. **Restart the KDC server.**
   ```
   kdc2 # svcadm restart network/security/krb5kdc
   ```
**How to Configure a Slave KDC to Use Full Propagation**

This procedure shows how to reconfigure a slave KDC server running the Solaris 10 release to use full propagation. Normally, the procedure would only need to be used if the master KDC server is running either the Solaris 9 release or an earlier release. In this case, the master KDC server cannot support incremental propagation, so the slave needs to be configured to allow propagation to work.

In this procedure, a 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 principal = kws/admin`

Note – Adjust the URL to point to the “SEAM Administration Tool” section, as described in the “Online Help URL in the SEAM Administration Tool” on page 359.

---

**Before You Begin**

The master KDC must be configured. For specific instructions if this slave is to be swappable, see “Swapping a Master KDC and a Slave KDC” on page 395.

1. **On the master KDC, become superuser.**

2. **On the master KDC, start `kadmin`.**

   You must log in with one of the `admin principal` names that you created when you configured the master KDC.

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

3. **On the master KDC, add slave host principals to the database, if not already done.**

   For the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

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

kadmin: quit

3 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

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

4 On the master KDC, add an entry for the master KDC and each slave KDC into the kpropd.acl file.

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
```

5 On all slave KDCs, copy the KDC administration files from the master KDC server.

This step needs to be followed on all slave KDCs, because 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 KDC:

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

6 On all slave KDCs, make sure that the Kerberos access control list file, `kadm5.acl`, is not populated.

An unmodified `kadm5.acl` file would look like:

kdc2 # cat /etc/krb5/kadm5.acl

```
*/admin@___default_realm___ *
```

If the file has kprop entries, remove them.
7 On the new slave, start the \texttt{kadmin} command.
You must log in with one of the \texttt{admin} principal names that you created when you configured the master KDC.

\texttt{kdc2 \# /usr/sbin/kadmin -p kws/admin}

Enter password: \texttt{Type kws/admin password}

\texttt{kadmin:}

a. Add the slave's host principal to the slave's keytab file by using \texttt{kadmin}.
This entry allows \texttt{kprop} and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the \texttt{/etc/resolv.conf} file.

\texttt{kadmin: ktadd host/kdc3.example.com}

Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab \texttt{WRFILE:/etc/krb5/krb5.keytab}.
Entry for principal host/kdc3.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab \texttt{WRFILE:/etc/krb5/krb5.keytab}.
Entry for principal host/kdc3.example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab \texttt{WRFILE:/etc/krb5/krb5.keytab}.
Entry for principal host/kdc3.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab \texttt{WRFILE:/etc/krb5/krb5.keytab}.

\texttt{kadmin: Quit kadmin.}

\texttt{kadmin: quit}

8 On the master KDC, add the slave KDC name to the \texttt{cron} job, which automatically runs the backups, by running \texttt{crontab -e}.
Add the name of each slave KDC server at the end of the \texttt{kprop_script} line.

\texttt{10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com \textit{kdc3.example.com}}

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

9 On the new slave, start the Kerberos propagation daemon.

\texttt{kdc3 \# svcadm enable network/security/krb5_prop}

10 On the master KDC, back up and propagate the database by using \texttt{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 411 for further instructions.

\texttt{kdc1 \# /usr/lib/krb5/kprop_script kdc3.example.com}

Database propagation to kdc3.example.com: SUCCEEDED
11 On the new slave, create a stash file by using `kdb5_util`.
   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>

12 (Optional) On the new slave KDC, synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.
   Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 394 for information about NTP.

13 On the new slave, start the KDC daemon (`krb5kdc`).
   `svcadm enable network/security/krb5kdc`


\section*{How to Verify That the KDC Servers Are Synchronized}

If incremental propagation has been configured, this procedure ensures that the information on the slave KDC has been updated.

1 On the KDC master server, run the `kproplog` command.
   `kdc1 # /usr/sbin/kproplog -h`

2 On a KDC slave server, run the `kproplog` command.
   `kdc2 # /usr/sbin/kproplog -h`

3 Check that the last serial # and the last timestamp values match.

\begin{example}

Verifying That the KDC Servers Are Synchronized

The following is a sample of results from running the `kproplog` command on the master KDC server.

`kdc1 # /usr/sbin/kproplog -h`

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
   Log version #: 1
   Log state: Stable
   Entry block size: 2048
   Number of entries: 2500
   First serial #: 137966
   Last serial #: 140465
\end{example}
The following is a sample of results from running the \texttt{kproplog} command on a slave KDC server.

\begin{verbatim}
kdc2 # /usr/sbin/kproplog -h
Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
  Log version #: 1
  Log state: Stable
  Entry block size: 2048
  Number of entries: 0
  First serial #: None
  Last serial #: 140465
  First time stamp: None
  Last time stamp: Fri Nov 28 01:06:13 2004
\end{verbatim}

Notice that the values for the last serial number and the last timestamp are identical, which indicates that the slave is synchronized with the master KDC server.

In the slave KDC server output, notice that no update entries exist in the slave KDC server’s update log. No entries exist because the slave KDC server does not keep a set of updates, unlike the master KDC server. Also, the KDC slave server does not include information on the first serial number or the first timestamp because this is not relevant information.

\section*{How to Manually Propagate the Kerberos Database to the Slave KDCs}

This procedure shows you how to propagate the Kerberos database by using the \texttt{kprop} command. Use this procedure if you need to synchronize a slave KDC with the master KDC outside the periodic \texttt{cron} job. Unlike the \texttt{kprop\_script}, you can use \texttt{kprop} to propagate just the current database backup without first making a new backup of the Kerberos database.

\textbf{Note} – Do not use this procedure if you are using incremental propagation.

1. Become superuser on the master KDC.

2. (Optional) Back up the database by using the \texttt{kdb5\_util} command.
\begin{verbatim}
# /usr/sbin/kdb5\_util dump /var/krb5/slave_datatrans
\end{verbatim}

3. Propagate the database to a slave KDC by using the \texttt{kprop} command.
\begin{verbatim}
# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
\end{verbatim}
Manually Propagating the Kerberos Database to the Slave KDCs Using `kprop_script`

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 Kerberos database to the slave KDCs. However, if your site has many slave KDCs, you might consider load-sharing the propagation process, known as *parallel propagation*.

---

**Note** – Do not use this procedure if you are using incremental propagation.

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

For example, say your site has one master KDC and six slave KDCs (shown in Figure 22–2), where `slave-1` through `slave-3` consist of one logical grouping and `slave-4` through `slave-6` consist of another logical grouping. To set up parallel propagation, you could have the master KDC propagate the database to `slave-1` and `slave-4`. In turn, those KDC slaves could propagate the database to the KDC slaves in their group.

```
master
  └──-propagation
    ├── slave-1
    │     └── Propagation Slaves
    │          └── slave-2
    │          └── slave-3
    └── slave-4
         └── Propagation Slaves
              └── slave-5
              └── slave-6
```

**Figure 22–2** Example of Parallel Propagation Configuration

Configuration Steps for Setting Up Parallel
Propagation

The following is not a detailed step-by-step procedure, but a high-level list of configuration steps to enable parallel propagation. These steps involve the following:

1. On the master KDC, changing the `kprop_script` entry in its `cron` job to include arguments for only the KDC slaves that will perform the succeeding propagation (the propagation slaves).

2. On each propagation slave, adding 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 Kerberos database.

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

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

**EXAMPLE 22–14 Setting Up Parallel Propagation**

Using the example in Figure 22–2, the master KDC’s `kprop_script` entry would look similar to the following:

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

The slave-1’s `kprop_script` entry would look similar to the following:

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

Note that the propagation on the slave starts an hour after it is propagated by the master.

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

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

The `kpropd.acl` file on the KDC slaves being propagated by slave-1 would 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 the `kdb5_util` utility 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 is not automatically removed with the database, you have to remove the stash file 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
   ```

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

   **Note** – If you need to re-create the stash file, you can use the `-f` option of the kdb5_util command.

Increasing Security on Kerberos Servers

Follow these steps to increase security on Kerberos application servers and on KDC servers.

How to Enable Only Kerberized Applications

This procedure restricts network access to the server that is running telnet, ftp, rcp, rsh, and rlogin to use Kerberos authenticated transactions only.

1. **Change the exec property for the telnet service.**
   Add the `-a user` option to the exec property for telnet to restrict access to those users who can provide valid authentication information.
   
   ```
   # inetadm -m svc:/network/telnet:default exec="/usr/sbin/in.telnetd -a user"
   ```

2. **(Optional) If not already configured, change the exec property for the telnet service.**
   Add the `-a` option to the exec property for ftp to permit only Kerberos authenticated connections.
   
   ```
   # inetadm -m svc:/network/ftp:default exec="/usr/sbin/in.ftpd -a"
   ```

3. **Disable other services.**
   The `in.rshd` and `in.rlogin` daemons should be disabled.
   
   ```
   # svcadm disable network/shell
   # svcadm disable network/login:rlogin
   ```
How to Restrict Access to KDC Servers

Both master KDC servers 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 Kerberos installation.

1. **Disable remote services, as needed.**
   
   To provide a secure KDC server, all nonessential network services should be disabled. Depending on your configuration, some of these services may already be disabled. Check the service status with the `svcs` command. 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.

   ```
   # svcadm disable network/comsat
   # svcadm disable network/dtspc/tcp
   # svcadm disable network/finger
   # svcadm disable network/login:rlogin
   # svcadm disable network/rexec
   # svcadm disable network/shell
   # svcadm disable network/talk
   # svcadm disable network/tname
   # svcadm disable network/uucp
   # svcadm disable network/rpc_100068_2-5/rpc_udp
   ```

2. **Restrict access to the hardware that supports the KDC.**
   
   To restrict physical access, make sure that the KDC server and its monitor are located in a secure facility. Users should not be able to access this server in any way.

3. **Store KDC database backups on local disks or on the KDC slaves.**
   
   Make tape backups of your KDC only if the tapes are stored securely. Follow the same practice for copies of keytab files. It would be best to store these files on a local file system that is not shared with other systems. The storage file system can be on either the master KDC server or any of the slave KDCs.
This chapter provides resolutions for error messages that you might receive when you use the Kerberos service. This chapter also provides some troubleshooting tips for various problems. This is a list of the error message and troubleshooting information in this chapter.

- “SEAM Administration Tool Error Messages” on page 417
- “Common Kerberos Error Messages (A-M)” on page 418
- “Common Kerberos Error Messages (N-Z)” on page 425
- “Problems With the Format of the krb5.conf File” on page 429
- “Problems Propagating the Kerberos Database” on page 429
- “Problems Mounting a Kerberized NFS File System” on page 429
- “Problems Authenticating as root” on page 430
- “Observing Mapping from GSS Credentials to UNIX Credentials” on page 430

### Kerberos Error Messages

This section provides information about Kerberos error messages, including why each error occurs 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 in with does not have the list privilege (l) in the Kerberos ACL file (kadm5.acl). So, you cannot view the principal list or policy list.
Solution: You must type the principal and policy names in the Name field to work on them, or you need to log in with a principal that has the appropriate privileges.

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: A serious problem exists with the Java Native Interface that is used by the SEAM Administration Tool (gkadmin).

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

Common Kerberos Error Messages (A-M)

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

All authentication systems disabled; connection refused

Cause: This version of rlogin does not support any authentication mechanism.

Solution: Make sure that rlogin is invoked with the -k option.

Another authentication mechanism must be used to access this host

Cause: Authentication could not be done.

Solution: Make sure that the client is using Kerberos V5 mechanism for authentication.

Authentication negotiation has failed, which is required for encryption. Good bye.

Cause: Authentication could not be negotiated with the server.

Solution: Start authentication debugging by invoking the telnet command with the toggle authdebug command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

Bad krb5 admin server hostname while initializing kadmin interface

Cause: An invalid host name is configured for admin_server in the krb5.conf file.
Solution: Make sure that the correct host name for the master KDC is specified on the admin_server line in the krb5.conf file.

Bad lifetime value
Cause: The lifetime value provided is not valid or incorrectly formatted.
Solution: Make sure that the value provided is consistent with the Time Formats section in the kinit(1) man page.

Bad start time value
Cause: The start time value provided is not valid or incorrectly formatted.
Solution: Make sure that the value provided is consistent with the Time Formats section in the kinit(1) man page.

Cannot contact any KDC for requested realm
Cause: No KDC responded in the requested realm.
Solution: Make sure that at least one KDC (either the master or a slave) is reachable or that the krb5kdc daemon is running on the KDCs. Check the /etc/krb5/krb5.conf file 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 that there is a default realm name, or that 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 that the Kerberos configuration file (krb5.conf) specifies a KDC in the realm section.

cannot initialize realm realm_name
Cause: The KDC might not have a stash file.
Solution: Make sure that the KDC has a stash file. If not, create a stash file by using the kdb5_util command, and try restarting the krb5kdc command.

Cannot resolve KDC for requested realm
Cause: Kerberos cannot determine any KDC for the realm.
Solution: Make sure that the Kerberos configuration file (krb5.conf) specifies a KDC in the realm section.

Cannot reuse password
Cause: The password that you specified 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 that are kept in the KDC database for each principal. This policy is enforced by the principal’s policy.

Can’t get forwarded credentials
Cause: Credential forwarding could not be established.
Solution: Make sure that the principal has forwardable credentials.

Can’t open/find Kerberos configuration file
Cause: The Kerberos configuration file (krb5.conf) was unavailable.
Solution: Make sure that the krb5.conf file is available in the correct location and has the correct permissions. This file should be writable by root and readable by everyone else.

Client did not supply required checksum—connection rejected
Cause: Authentication with checksum was not negotiated with the client. The client might be using an old Kerberos V5 protocol that does not support initial connection support.
Solution: Make sure that the client is using a Kerberos V5 protocol that supports initial connection support.

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 that 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.

Communication failure with server while initializing kadmin interface
Cause: The host that was specified for the admin server, also called the master KDC, did not have the kadmin daemon running.
Solution: Make sure that you specified the correct host name for the master KDC. If you specified the correct host name, make sure that kadmin is running on the master KDC that 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 that 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 that the credentials cache has not been removed, and that there is space left on the device by using the `df` command.

**Decrypt integrity check failed**

**Cause:** You might have an invalid ticket.

**Solution:** Verify both of these conditions:
- Make sure that your credentials are valid. Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.
- Make sure that the target host has a keytab file with the correct version of the service key. Use `kadmin` to view the key version number of the service principal (for example, `host/FQDN-hostname`) in the Kerberos database. Also, use `klist -k` on the target host to make sure that it has the same key version number.

**Encryption could not be enabled. Goodbye.**

**Cause:** Encryption could not be negotiated with the server.

**Solution:** Start authentication debugging by invoking the `telnet` command with the `toggle encdebug` command and look at the debug messages for further clues.

**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 that you used the correct principal and password when you executed `kadmin`.

**Field is too long for this implementation**

**Cause:** The message size that was being sent by a Kerberized application was too long. This error could be generated if the transport protocol is UDP, which has a default maximum message size 65535 bytes. In addition, there are limits on individual fields within a protocol message that is sent by the Kerberos service.

**Solution:** Verify that you have not restricted the transport to UDP in the KDC server’s `/etc/krb5/kdc.conf` file.

**GSS-API (or Kerberos) error**

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

**Solution:** Check the `/var/www logs` file to find the more specific 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 that the host name is defined in DNS and that 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 might not have the correct trust relationships set up.
Solution: Make sure that the realms you are using have the correct trust relationships.

Improper format of Kerberos configuration file
Cause: The Kerberos configuration file has invalid entries.
Solution: Make sure that all the relations in the \texttt{krb5.conf} file are followed by the “=” sign and a value. Also, 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 \texttt{krb5.conf} and \texttt{kdc.conf} files.

Incorrect net address
Cause: There was a mismatch in the network address. The network address in the ticket that was being forwarded was different from the network address where the ticket was processed. This message might occur when tickets are being forwarded.
Solution: Make sure that the network addresses are correct. Destroy your tickets with \texttt{kdestroy}, and create new tickets with \texttt{kinit}.

Invalid credential was supplied
Service key not available
Cause: The service ticket in the credentials cache may be incorrect.
Solution: Destroy current credential cache and rerun \texttt{kinit} before trying to use this service.

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

Invalid number of character classes
Cause: The password that you specified for the principal does not contain enough password classes, as enforced by the principal’s policy.
Solution: Make sure that you specify a password with the minimum number of password classes that the policy requires.

KADM err: Memory allocation failure
Cause: There is insufficient 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 might be that postdating or forwardable options were being requested, and the KDC did not allow them. Another problem might be that you requested the renewal of a TGT, but you didn’t have a renewable TGT.
Solution: Determine if you are either requesting an option that the KDC does not allow or a type of ticket that is not available.

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 that you are using kinit with the correct options. If necessary, modify the policy that is associated with the principal or change the principal’s attributes to allow the request. You can modify the policy or principal by using kadmin.

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 that the KDC you are communicating with complies with RFC1510, that the request you are sending is a Kerberos V5 request, or that the KDC is available.

kdestroy: Could not obtain principal name from cache
Cause: The credentials cache is missing or corrupted.
Solution: Check that the cache location provided is correct. Remove and obtain a new TGT using kinit, if necessary.

kdestroy: No credentials cache file found while destroying cache
Cause: The credentials cache (/tmp/krb5c_uid) is missing or corrupted.
Solution: Check that the cache location provided is correct. Remove and obtain a new TGT using kinit, if necessary.

kdestroy: TGT expire warning NOT deleted
Cause: The credentials cache is missing or corrupted.
Solution: Check that the cache location provided is correct. Remove and obtain a new TGT using kinit, if necessary.

Kerberos authentication failed
Cause: The Kerberos password is either incorrect or the password might not be synchronized with the UNIX password.

Solution: If the password are not synchronized, then you must specify a different password to complete Kerberos authentication. It is possible that the user has forgotten their original password.

Kerberos V5 refuses authentication
Cause: Authentication could not be negotiated with the server.

Solution: Start authentication debugging by invoking the telnet command with the toggle authdebug command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

Key table entry not found
Cause: No entry exists for the service principal in the network application server’s keytab file.

Solution: Add the appropriate service principal to the server’s keytab file so that it can provide the Kerberized service.

Key version number for principal in key table is incorrect
Cause: A principal’s key version in the keytab file is different from the version in the Kerberos database. Either a service’s key has been changed, or you might 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 file where the service is running.

Alternately, you might be using an old service ticket that has an older key. You might want to run the kdestroy command and then the kinit command again.

kinit: gethostname failed
Cause: An error in the local network configuration is causing kinit to fail.

Solution: Make sure that the host is configured correctly.

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 that the Kerberos PAM module is in the `/usr/lib/security` directory and that it is a valid executable binary. Also, make sure that the `/etc/pam.conf` file 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 that 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 that contains the master key. The master key is located in `/var/krb5/.k5.REALM`.

Solution: Make sure that the master key in the loaded database dump matches the master key that is located in `/var/krb5/.k5.REALM`.

Matching credential not found

Cause: The matching credential for your request was not found. Your request requires credentials that are unavailable in the credentials cache.

Solution: Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

Message out of order

Cause: Messages that were sent using sequential-order privacy arrived out of order. Some messages might have been lost in transit.

Solution: You should reinitialize the Kerberos session.

Message stream modified

Cause: There was a mismatch between the computed checksum and the message checksum. The message might have been modified while in transit, which can indicate a security leak.

Solution: Make sure that the messages are being sent across the network correctly. Because this message can also indicate the possible tampering of messages while they are being sent, destroy your tickets using `kdestroy` and reinitialize the Kerberos services that you are using.

**Common Kerberos Error Messages (N-Z)**

This section provides an alphabetical list (N-Z) of common error messages for the Kerberos commands, Kerberos daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

No credentials cache file found

Cause: Kerberos could not find the credentials cache (/tmp/krb5cc_uid).
Solution: Make sure that the credential file exists and is readable. If it isn't, try performing `kinit` again.

No credentials were supplied, or the credentials were unavailable or inaccessible

No credential cache found

Cause: The user's credential cache is incorrect or does not exist.

Solution: The user should run `kinit` before trying to start the service.

No credentials were supplied, or the credentials were unavailable or inaccessible

No principal in keytab matches desired name

Cause: An error occurred while trying to authenticate the server.

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

Operation requires "privilege" privilege

Cause: The `admin` principal that was being used does not have the appropriate privilege configured in the `kadmind.acl` file.

Solution: Use a principal that has the appropriate privileges. Or, configure the principal that was being used to have the appropriate privileges by modifying the `kadmind.acl` file. Usually, a principal with `/admin` as part of its name has the appropriate privileges.

PAM-KRB5 (auth): `krb5_verify_init_creds` failed: Key table entry not found

Cause: The remote application tried to read the host's service principal in the local `/etc/krb5/krb5.keytab` file, but one does not exist.

Solution: Add the host's service principal to the host's keytab file.

Password is in the password dictionary

Cause: The password that you specified is in a password dictionary that is being used. Your password 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. Your server might have been first run under a user ID different than your current user ID.

Solution: Make sure that the replay cache has the appropriate permissions. The replay cache is stored on the host where the Kerberized server application is running. The replay cache file is called `/var/krb5/rcache/rc_service_name_uid` for non-root users. For root users the replay cache file is called `/var/krb5/rcache/root/rc_service_name`.

Protocol version mismatch

Cause: Most likely, a Kerberos V4 request was sent to the KDC. The Kerberos service supports only the Kerberos V5 protocol.
Solution: Make sure that 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 might have been stolen, and someone else is trying to reuse the tickets.

Solution: Wait for a few minutes, and reissue the request.

Requested principal and ticket don’t match

Cause: The service principal that you are connecting to and the service ticket that you have do not match.

Solution: Make sure that DNS is functioning properly. If you are using another vendor’s software, make sure that the software is using principal names correctly.

Requested protocol version not supported

Cause: Most likely, a Kerberos V4 request was sent to the KDC. The Kerberos service supports only the Kerberos V5 protocol.

Solution: Make sure that your applications are using the Kerberos V5 protocol.

Server refused to negotiate authentication, which is required for encryption. Good bye.

Cause: The remote application is not capable or has been configured not to accept Kerberos authentication from the client.

Solution: Provide a remote application that can negotiate authentication or configure the application to use the appropriate flags to turn on authentication.

Server refused to negotiate encryption. Good bye.

Cause: Encryption could not be negotiated with the server.

Solution: Start authentication debugging by invoking the telnet command with the toggle encdebug command and look at the debug messages for further clues.

Server rejected authentication (during sendauth exchange)

Cause: The server that you are trying to communicate with rejected the authentication. Most often, this error occurs during Kerberos database propagation. Some common causes might be problems with the kpropd.acl file, DNS, or the keytab file.
**Solution:** If you get this error when you are running applications other than kprop, investigate whether the server’s keytab file is correct.

**The ticket isn’t for us**

**Ticket/authenticator don’t match**

*Cause:* There was a mismatch between the ticket and the authenticator. The principal name in the request might 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 a non-FQDN name, or vice versa.

**Solution:** If you get this error when you are running applications other than kprop, investigate whether the server’s keytab file 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` to allow postdating.

**Ticket not yet valid**

*Cause:* The postdated ticket is not valid yet.

**Solution:** Create a new ticket with the correct date, or wait until the current ticket is valid.

**Truncated input file detected**

*Cause:* The database dump file that was being used in the operation is not a complete dump file.

**Solution:** Create the dump file again, or use a different database dump file.

**Unable to securely authenticate user ... exit**

*Cause:* Authentication could not be negotiated with the server.

**Solution:** Start authentication debugging by invoking the `telnet` command with the `toggle authdebug` command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

**Wrong principal in request**

*Cause:* There was an invalid principal name in the ticket. This error might indicate a DNS or FQDN problem.

**Solution:** Make sure that the principal of the service matches the principal in the ticket.
Kerberos Troubleshooting

This section provides troubleshooting information for the Kerberos 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 problem occurs, the following error message is displayed:

```
Error initializing krb5: Improper format of Kerberos configuration
```

In addition, an incorrectly formatted krb5.conf file, prevents the applications that use the GSSAPI from using the krb5 mechanisms.

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 you allow Kerberos features to be used.

**Problems Propagating the Kerberos Database**

If propagating the Kerberos database fails, try /usr/bin/rlogin -x between the slave KDC and master KDC, and vice versa.

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 enable the eklogin service.

```
# svcadm enable svc:/network/login:eklogin
```

After you finish troubleshooting the problem, you need to disable the eklogin service.

If rlogin does not work, problems are likely because of the keytab files on the KDCs. If rlogin does work, the problem is not in the keytab file or the name service, because rlogin and the propagation software use the same host/host-name principal. In this case, make sure that the kpropd.acl file is correct.

**Problems Mounting a Kerberized NFS File System**

- If mounting a Kerberized NFS file system fails, make sure that the /var/rcache/root file exists on the NFS server. If the file system 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 that the gssd service is enabled on your system and the NFS server.
- If you see either the invalid argument or bad directory error message when you are trying to access a Kerberized NFS file system, the problem might be that you are not using a fully qualified DNS name when you are trying to mount the NFS file system. The host that is being mounted is not the same as the host name part of the service principal in the server’s keytab file.
This problem might 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 the Kerberos service, you should set up multiple address records per host as follows:\(^1\):

- `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.`,  
  `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 a single service principal instead of three service principals in the server's keytab file.

**Problems Authenticating as root**

If authentication fails when you try to become superuser on your system and you have already added the root principal to your host's keytab file, there are two potential problems to check. First, make sure that the root principal in the keytab file has a fully qualified hostname as its instance. If it does, check the `/etc/resolv.conf` file to make sure that the system is correctly set up as a DNS client.

**Observing Mapping from GSS Credentials to UNIX Credentials**

To be able to monitor the credential mappings, first uncomment this line from the `/etc/gss/gsscred.conf` file.

```
SYSLOG_UID_MAPPING=yes
```

Next instruct the gssd service to get information from the `/etc/gss/gsscred.conf` file.

```
# pkill -HUP gssd
```

Now you should be able to monitor the credential mappings as gssd requests them. The mappings are recorded by syslogd, if the syslog.conf file is configured for the auth system facility with the debug severity level.

---


---

**Kerberos Troubleshooting**

System Administration Guide: Security Services • May 2006

430
Administering Kerberos Principals and Policies (Tasks)

This chapter provides procedures for administering principals and the policies that are associated with them. This chapter also shows how to administer a host’s keytab file.

This chapter should be used by anyone who needs to administer principals and policies. Before you use this chapter, you should be familiar with principals and policies, including any planning considerations. Refer to Chapter 20 and Chapter 21, respectively.

This is a list of the information in this chapter.

- "Ways to Administer Kerberos Principals and Policies" on page 431
- "SEAM Administration Tool" on page 432
- "Administering Kerberos Principals" on page 436
- "Administering Kerberos Policies" on page 449
- "SEAM Tool Reference" on page 457
- "Administering Keytab Files" on page 461

Ways to Administer Kerberos 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 to modify their attributes, you can use either the kadmin or gkadmin command.

The kadmin command provides an interactive command-line interface that enables you to maintain Kerberos principals, policies, and keytab files. There are two versions of the kadmin command:

- kadmin - Uses Kerberos authentication to operate securely from anywhere on the network
- kadmin.local – Must be run directly on the master KDC

Other than kadmin using Kerberos to authenticate the user, the capabilities of the two versions are identical. The local version is necessary to enable you to set up enough of the database so that you can use the remote version.
Also, the Solaris release provides the SEAM Administration Tool, \texttt{gkadmin}, which is an interactive graphical user interface (GUI) that provides essentially the same capabilities as the \texttt{kadmin} command. See “SEAM Administration Tool” on page 432 for more information.

**SEAM Administration Tool**

The SEAM Administration Tool (SEAM Tool) is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. This tool provides much the same capabilities as the \texttt{kadmin} command. However, this tool does not support the management of keytab files. You must use the \texttt{kadmin} command to administer keytab files, which is described in “Administering Keytab Files” on page 461.

Similar to the \texttt{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 do the following:

- Create new principals that are based on default values or existing principals.
- Create new policies that are 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 lists and policy lists.
- View and search principal lists and policy lists.

The SEAM Tool also provides context-sensitive help and general online help.

The following task maps provide pointers to the various tasks that you can do with the SEAM Tool:

- “Administering Kerberos Principals (Task Map)” on page 437
- “Administering Kerberos Policies (Task Map)” on page 449

Also, go to “SEAM Tool Panel Descriptions” on page 457 for descriptions of all the principal attributes and policy attributes that you can either specify or view in the SEAM Tool.

**Command-Line Equivalents of the SEAM Tool**

This section lists the \texttt{kadmin} commands that provide the same capabilities as the SEAM Tool. These commands can be used without running an X Window system. Even though most procedures in this chapter use the SEAM Tool, many procedures also provide corresponding examples that use the command-line equivalents.

<table>
<thead>
<tr>
<th>SEAM Tool Procedure</th>
<th>Equivalent \texttt{kadmin} Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the list of principals.</td>
<td>\texttt{list_principals} or \texttt{get_principals}</td>
</tr>
</tbody>
</table>
TABLE 24–1 Command-Line Equivalents of the SEAM Tool  (Continued)

<table>
<thead>
<tr>
<th>SEAM Tool Procedure</th>
<th>Equivalent kadmin Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>View a principal’s attributes.</td>
<td>get_principal</td>
</tr>
<tr>
<td>Create a new principal.</td>
<td>add_principal</td>
</tr>
<tr>
<td>Duplicate a principal.</td>
<td>No command-line equivalent</td>
</tr>
<tr>
<td>Modify a principal.</td>
<td>modify_principal or change_password</td>
</tr>
<tr>
<td>Delete a principal.</td>
<td>delete_principal</td>
</tr>
<tr>
<td>Set up defaults for creating new principals.</td>
<td>No command-line equivalent</td>
</tr>
<tr>
<td>View the list of policies.</td>
<td>list_policies or get_policies</td>
</tr>
<tr>
<td>View a policy’s attributes.</td>
<td>get_policy</td>
</tr>
<tr>
<td>Create a new policy.</td>
<td>add_policy</td>
</tr>
<tr>
<td>Duplicate a policy.</td>
<td>No command-line equivalent</td>
</tr>
<tr>
<td>Modify a policy.</td>
<td>modify_policy</td>
</tr>
<tr>
<td>Delete a policy.</td>
<td>delete_policy</td>
</tr>
</tbody>
</table>

The Only File Modified by the SEAM Tool

The only file that the SEAM Tool modifies is the $HOME/.gkadmin file. This file contains the default values for creating new principals. You can update this file by choosing Properties from the Edit menu.

Print and Online Help Features of the SEAM Tool

The SEAM Tool provides both print features 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 principal or the loaded principal
- The currently selected policy or the loaded policy

From the Help menu, you can access 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 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 to accumulate 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. Thus, your productivity with the tool will decrease. There are several ways to work around this problem.

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 type a principal or policy name in the new Name field that is provided, then select the operation that you want to perform on it. In effect, typing a name is 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 retrieve the lists again. This option eliminates the need to keep loading the lists from the server, which is what takes so long.

You can set list caching by 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 as the first option does. Also, even though caching doesn’t enable you to see the changes of other users, you can still see the latest list information based on your changes, because your changes update the lists both on the server and in the cache. And, if you want to update the cache to see other changes 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 SEAM Administration Login window is displayed.
2 **If you don't want to use the default values, specify new default values.**

The 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 retrieve the default values, click Start Over.

---

**Note** – The administration operations that each Principal Name can perform are dictated by the Kerberos ACL file, `/etc/krb5/kadm5.acl`. For information about limited privileges, see “Using the SEAM Tool With Limited Kerberos Administration Privileges” on page 460.

---

3 **Type a password for the specified principal name.**

4 **Click OK.**

The following window is displayed.
Administering Kerberos Principals

This section provides the step-by-step instructions used to administer principals with the SEAM Tool. This section also provides examples of command-line equivalents, when available.
Administering Kerberos Principals (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the list of principals.</td>
<td>View the list of principals by clicking the Principals tab.</td>
<td>&quot;How to View the List of Kerberos Principals&quot; on page 438</td>
</tr>
<tr>
<td>View a principal’s attributes.</td>
<td>View a principal’s attributes by selecting the Principal in the Principal List, then clicking the Modify button.</td>
<td>&quot;How to View a Kerberos Principal’s Attributes&quot; on page 440</td>
</tr>
<tr>
<td>Create a new principal.</td>
<td>Create a new principal by clicking the Create New button in the Principal List panel.</td>
<td>&quot;How to Create a New Kerberos Principal&quot; on page 442</td>
</tr>
<tr>
<td>Duplicate a principal.</td>
<td>Duplicate a principal by selecting the principal to duplicate in the Principal List, then clicking the Duplicate button.</td>
<td>&quot;How to Duplicate a Kerberos Principal&quot; on page 444</td>
</tr>
<tr>
<td>Modify a principal.</td>
<td>Modify a principal by selecting the principal to modify in the Principal List, then clicking the Modify button. 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.</td>
<td>&quot;How to Modify a Kerberos Principal&quot; on page 444</td>
</tr>
<tr>
<td>Delete a principal.</td>
<td>Delete a principal by selecting the principal to delete in the Principal List, then clicking the Delete button.</td>
<td>&quot;How to Delete a Kerberos Principal&quot; on page 445</td>
</tr>
<tr>
<td>Set up defaults for creating new principals.</td>
<td>Set up defaults for creating new principals by choosing Properties from the Edit menu.</td>
<td>&quot;How to Set Up Defaults for Creating New Kerberos Principals&quot; on page 446</td>
</tr>
<tr>
<td>Modify the Kerberos administration privileges (kadm5.acl file).</td>
<td>Command-line only. 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 /etc/krb5/kadm5.acl file to modify the Kerberos administration privileges for each principal.</td>
<td>&quot;How to Modify the Kerberos Administration Privileges&quot; on page 447</td>
</tr>
</tbody>
</table>

Automating the Creation of New Kerberos 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 10 or even 100 new principals in a short 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 automate the creation of new principals:
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, which contains a principal name and its password on each line, separated by one or more spaces. The `+needchange` option configures the principal so that the user is prompted for a new password during login with the principal for the first time. This practice helps to ensure that the passwords in the `princnames` file are not a security risk.

You can build more elaborate scripts. For example, your script could use 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 determined by your site’s needs and your scripting expertise.

**How to View the List of Kerberos Principals**

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   
   See “How to Start the SEAM Tool” on page 434 for more information.

   $ /usr/sbin/gkadmin

2. **Click the Principals tab.**
   
   The list of principals is displayed.
3 Display a specific principal or a sublist of principals.

Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of principals that match 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 type the filter string ge, the filter mechanism displays 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 24–1  Viewing the List of Kerberos Principals (Command Line)

In the following example, the list_principals command of kadmin is used to list all the principals that match test*. Wildcards can be used with the list_principals command.

```bash
kadmin: list_principals test*
test1@EXAMPLE.COM
test2@EXAMPLE.COM
kadmin: quit
```

How to View a Kerberos Principal’s Attributes

An example of the command-line equivalent follows this procedure.

1 If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 434 for more information.
   $ /usr/sbin/gkadmin

2 Click the Principals tab.

3 Select the principal in the list that you want to view, then click Modify.
   The Principal Basics panel that contains some of the principal’s attributes is displayed.

4 Continue to click Next to view 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, for all the principal attribute descriptions, go to “SEAM Tool Panel Descriptions” on page 457.

5 When you are finished viewing, click Cancel.

Example 24–2  Viewing a Kerberos Principal’s Attributes

The following example shows the first window when you are viewing the jdb/admin principal.
In the following example, the `get_principal` command of `kadmin` is used to view the attributes of the `jdb/admin` principal.

```
krb5-admin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM
Expiration date: Fri Aug 25 17:19:05 PDT 2004
Last password change: [never]
Password expiration date: Wed Apr 14 11:53:10 PDT 2003
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
```
How to Create a New Kerberos Principal

An example of the command-line equivalent follows this procedure.

1 **If necessary, start the SEAM Tool.**
   
   See “How to Start the SEAM Tool” on page 434 for more information.

   ![Note](https://example.com/note.png)
   
   **Note** – If you are creating a new principal that might need a new policy, you should create the new policy before you create the new principal. Go to “How to Create a New Kerberos Policy” on page 453.

   $ /usr/sbin/gkadmin

2 **Click the Principals tab.**

3 **Click New.**
   
   The Principal Basics panel that contains some attributes for a principal is displayed.

4 **Specify a principal name and a password.**
   
   Both the principal name and the 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, for all the principal attribute descriptions, go to “SEAM Tool Panel Descriptions” on page 457.

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 447 for more details.
Example 24–4  Creating a New Kerberos Principal

The following example shows the Principal Basics panel when a new principal called pak is created. The policy is set to testuser.

Example 24–5  Creating a New Kerberos Principal (Command Line)

In the following example, the add_principal command of kadmin is used to create a new principal called pak. The principal’s policy is set to testuser.

```bash
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 Kerberos Principal

This procedure explains how to use all or some of the attributes of an existing principal to create a new principal. No command-line equivalent exists for this procedure.

1 If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 434 for more information.
   $ /usr/sbin/gkadmin

2 Click the Principal tab.

3 Select the principal in the list that you want to duplicate, then 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 a password.
   Both the principal name and the password are mandatory. To make an exact duplicate of the principal you selected, click Save and skip to Step 7.

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, for all the principal attribute descriptions, go to “SEAM Tool Panel Descriptions” on page 457.

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 447 for more details.

How to Modify a Kerberos Principal

An example of the command-line equivalent follows this procedure.

1 If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 434 for more information.
   $ /usr/sbin/gkadmin
2 Click the Principals tab.

3 Select the principal in the list that you want to modify, then click Modify.
The Principal Basics panel that contains 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, for all the principal attribute
descriptions, go to “SEAM Tool Panel Descriptions” on page 457.

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 447 for more details.

Example 24-6  Modifying a Kerberos Principal’s Password (Command Line)

In the following example, the change_password command of kadmin is used to modify the password
for the jdb principal. The change_password command does not let you change the password to a
password 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 Kerberos Principal

An example of the command-line equivalent follows this procedure.

1 If necessary, start the SEAM Tool.
See “How to Start the SEAM Tool” on page 434 for more information.

$ /usr/sbin/gkadmin

2 Click the Principals tab.
Select the principal in the list that you want to delete, then click Delete.
After you confirm the deletion, the principal is deleted.

Remove the principal from the Kerberos access control list (ACL) file, /etc/krb5/kadm5.acl.
See “How to Modify the Kerberos Administration Privileges” on page 447 for more details.

Example 24–7 Deleting a Kerberos Principal (Command Line)

In the following example, the delete_principal command of kadmin is used 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 Kerberos Principals

No command-line equivalent exists for this procedure.

1 If necessary, start the SEAM Tool.
See “How to Start the SEAM Tool” on page 434 for more information.

2 Choose Properties from the Edit Menu.
The Properties window is displayed.
3 Select the defaults that you want to use when you create new principals.

Choose Context-Sensitive Help from the Help menu for information about the various attributes in each window.

4 Click Save.

**How to Modify the Kerberos Administration Privileges**

Even though your site probably has many 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.acl*. The *kadm5.acl* 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** Specifies 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 want 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, the user `jdb` obtains `jdb/admin` tickets only when he or she actually needs to use those privileges.

**privileges** Specifies which operations can or cannot be performed by the principal. This field consists of 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 polices.
c [Dis]allows the changing of passwords for principals.
i [Dis]allows inquiries to the Kerberos database.
l [Dis]allows the listing of principals or policies in the Kerberos database.
x or * Allows all privileges (admcil).
```

**principal-target** When a principal is specified in this field, the privileges apply to the principal only when the principal operates on the principal-target. Any part of the principal name can include the '*' wildcard, which is useful to group principals.

---

**Example 24–8** Modifying the Kerberos Administration Privileges

The following entry in the `kadm5.acl` file gives any principal in the `EXAMPLE.COM` realm with the `admin` instance all the privileges on the Kerberos database:

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

The following entry in the `kadm5.acl` file gives the `jdb@EXAMPLE.COM` principal the privileges to add, list, and inquire about any principal that has the `root` instance.

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```
Administering Kerberos Policies

This section provides step-by-step instructions used to administer policies with the SEAM Tool. This section also provides examples of command-line equivalents, when available.

Administering Kerberos Policies (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the list of policies.</td>
<td>View the list of policies by clicking the Policies tab.</td>
<td>&quot;How to View the List of Kerberos Policies&quot; on page 449</td>
</tr>
<tr>
<td>View a policy's attributes.</td>
<td>View a policy's attributes by selecting the policy in the Policy List, then clicking the Modify button.</td>
<td>&quot;How to View a Kerberos Policy's Attributes&quot; on page 451</td>
</tr>
<tr>
<td>Create a new policy.</td>
<td>Create a new policy by clicking the Create New button in the Policy List panel.</td>
<td>&quot;How to Create a New Kerberos Policy&quot; on page 453</td>
</tr>
<tr>
<td>Duplicate a policy.</td>
<td>Duplicate a policy by selecting the policy to duplicate in the Policy List, then clicking the Duplicate button.</td>
<td>&quot;How to Duplicate a Kerberos Policy&quot; on page 455</td>
</tr>
<tr>
<td>Modify a policy.</td>
<td>Modify a policy by selecting the policy to modify in the Policy List, then 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.</td>
<td>&quot;How to Modify a Kerberos Policy&quot; on page 455</td>
</tr>
<tr>
<td>Delete a policy.</td>
<td>Delete a policy by selecting the policy to delete in the Policy List, then clicking the Delete button.</td>
<td>&quot;How to Delete a Kerberos Policy&quot; on page 456</td>
</tr>
</tbody>
</table>

How to View the List of Kerberos Policies

An example of the command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.
   See "How to Start the SEAM Tool" on page 434 for more information.
   
   $ /usr/sbin/gkadmin

2. Click the Policies tab.
   The list of policies is displayed.
3 Display a specific policy or a sublist of policies.

Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of policies that match 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 type the filter string ge, the filter mechanism displays 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.
Viewing the List of Kerberos Policies (Command Line)

In the following example, the `list_policies` command of `kadmin` is used 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 Kerberos Policy’s Attributes

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See “How to Start the SEAM Tool” on page 434 for more information.
   `$ /usr/sbin/gkadmin`

2. **Click the Policies tab.**

3. **Select the policy in the list that you want to view, then click Modify.**
   The Policy Details panel is displayed.

4. **When you are finished viewing, click Cancel.**

Viewing a Kerberos Policy’s Attributes

The following example shows the Policy Details panel when you are viewing the `test` policy.
Example 24–11  Viewing a Kerberos Policy’s Attributes (Command Line)

In the following example, the `get_policy` command of `kadmin` is used to view the attributes of the enguser policy.

```plaintext
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
```
The Reference count is the number of principals that use this policy.

**How to Create a New Kerberos Policy**

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See “How to Start the SEAM Tool” on page 434 for more information.

   ```bash
   /usr/sbin/gkadmin
   ```

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 for information about the various attributes in this window. Or, go to Table 24–5 for all the policy attribute descriptions.

6. **Click Save to save the policy, or click Done.**

**Example 24–12 Creating a New Kerberos Policy**

In the following example, a new policy called build11 is created. The Minimum Password Classes is set to 3.
Example 24–13  Creating a New Kerberos Policy (Command Line)

In the following example, the add_policy command of kadmin is used to create the build11 policy. This policy requires at least 3 character classes in a password.

```
$ kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```
**How to Duplicate a Kerberos Policy**

This procedure explains how to use all or some of the attributes of an existing policy to create a new policy. No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**
   
   See “How to Start the SEAM Tool” on page 434 for more information.

   $ /usr/sbin/gkadmin

2. **Click the Policies tab.**

3. **Select the policy in the list that you want to duplicate, then 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. To make an exact duplicate of the policy you selected, skip to Step 6.

5. **Specify different values for the policy’s attributes.**
   
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 24–5 for all the policy attribute descriptions.

6. **Click Save to save the policy, or click Done.**

**How to Modify a Kerberos Policy**

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   
   See “How to Start the SEAM Tool” on page 434 for details.

   $ /usr/sbin/gkadmin

2. **Click the Policies tab.**

3. **Select the policy in the list that you want to modify, then click Modify.**
   
   The Policy Details panel is displayed.

4. **Modify the policy’s attributes.**
   
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 24–5 for all the policy attribute descriptions.
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 24–14 Modifying a Kerberos Policy (Command Line)
In the following example, the modify_policy command of kadmin is used 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 Kerberos Policy
An example of the command-line equivalent follows this procedure.

Note – Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to modify the principals' Policy attribute. The policy cannot be deleted if any principal is using it.

1 If necessary, start the SEAM Tool. See “How to Start the SEAM Tool” on page 434 for more information.

$ /usr/sbin/gkadmin

2 Click the Policies tab.

3 Select the policy in the list that you want to delete, then click Delete.
After you confirm the deletion, the policy is deleted.

Example 24–15 Deleting a Kerberos Policy (Command Line)
In the following example, the delete_policy command of the kadmin command is used 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 you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to use the modify_principal -policy command of kadmin on the affected principals. The delete_policy command fails if the policy is in use by a principal.
SEAM Tool Reference

This section provides descriptions of each panel in the SEAM Tool. Also, information about using limited privileges with SEAM Tool are provided.

SEAM Tool Panel Descriptions

This section provides descriptions for each principal and policy attribute that you can either specify or view in the SEAM Tool. The attributes are organized by the panel in which they are displayed.

**TABLE 24–2 Attributes for the Principal Basics Panel of the SEAM Tool**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal Name</td>
<td>The name of the principal (which is the primary/instance 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 its name.</td>
</tr>
<tr>
<td>Password</td>
<td>The password for the principal. You can use the Generate Random Password button to create a random password for the principal.</td>
</tr>
<tr>
<td>Policy</td>
<td>A menu of available policies for the principal.</td>
</tr>
<tr>
<td>Account Expires</td>
<td>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 might be unable to log in.</td>
</tr>
<tr>
<td>Last Principal Change</td>
<td>The date on which information for the principal was last modified. (Read only)</td>
</tr>
<tr>
<td>Last Changed By</td>
<td>The name of the principal that last modified the account for this principal. (Read only)</td>
</tr>
<tr>
<td>Comments</td>
<td>Comments that are related to the principal (for example, “Temporary Account”).</td>
</tr>
</tbody>
</table>

**TABLE 24–3 Attributes for the Principal Details Panel of the SEAM Tool**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Success</td>
<td>The date and time when the principal last logged in successfully. (Read only)</td>
</tr>
<tr>
<td>Last Failure</td>
<td>The date and time when the last login failure for the principal occurred. (Read only)</td>
</tr>
<tr>
<td>Failure Count</td>
<td>The number of times a login failure has occurred for the principal. (Read only)</td>
</tr>
<tr>
<td>Last Password Change</td>
<td>The date and time when the principal’s password was last changed. (Read only)</td>
</tr>
<tr>
<td>Password Expires</td>
<td>The date and time when the principal’s current password expires.</td>
</tr>
<tr>
<td>Key Version</td>
<td>The key version number for the principal. This attribute is normally changed only when a password has been compromised.</td>
</tr>
</tbody>
</table>
### TABLE 24-3 Attributes for the Principal Details Panel of the SEAM Tool (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Lifetime (seconds)</td>
<td>The maximum length of time for which a ticket can be granted for the principal (without renewal).</td>
</tr>
<tr>
<td>Maximum Renewal (seconds)</td>
<td>The maximum length of time for which an existing ticket can be renewed for the principal.</td>
</tr>
</tbody>
</table>

### TABLE 24-4 Attributes of the Principal Flags Panel of the SEAM Tool

<table>
<thead>
<tr>
<th>Attribute (Radio Buttons)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable Account</td>
<td>When checked, prevents the principal from logging in. This attribute provides an easy way to temporarily freeze a principal account.</td>
</tr>
<tr>
<td>Require Password Change</td>
<td>When checked, expires the principal’s current password, which forces the user to use the <code>kpasswd</code> command to create a new password. This attribute is useful if a security breach occurs, and you need to make sure that old passwords are replaced.</td>
</tr>
<tr>
<td>Allow Postdated Tickets</td>
<td>When checked, allows the principal to obtain postdated tickets. For example, you might need to use postdated tickets for <code>cron</code> jobs that must run after hours, but you cannot obtain tickets in advance because of short ticket lifetimes.</td>
</tr>
<tr>
<td>Allow Forwardable Tickets</td>
<td>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>, then other services, such as NFS services, are available without your being prompted for another password.</td>
</tr>
<tr>
<td>Allow Renewable Tickets</td>
<td>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 ticket expires). Currently, the NFS service is the ticket service that can renew tickets.</td>
</tr>
<tr>
<td>Allow Proxiable Tickets</td>
<td>When checked, allows the principal to obtain proxiable tickets. 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. However, the service cannot obtain a ticket-granting ticket (TGT).</td>
</tr>
<tr>
<td>Allow Service Tickets</td>
<td>When checked, allows service tickets to be issued for the principal. You should not allow service tickets to be issued for the <code>kadmin</code>/<code>hostname</code> and <code>changepw</code>/<code>hostname</code> principals. This practice ensures that only these principals can update the KDC database.</td>
</tr>
</tbody>
</table>
### TABLE 24–4 Attributes of the Principal Flags Panel of the SEAM Tool (Continued)

<table>
<thead>
<tr>
<th>Attribute (Radio Buttons)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow TGT-Based Authentication</td>
<td>When checked, allows the service principal to provide services to another principal. More specifically, this attribute allows the KDC to issue a service ticket for the service principal. This attribute is valid only for service principals. When unchecked, service tickets cannot be issued for the service principal.</td>
</tr>
<tr>
<td>Allow Duplicate Authentication</td>
<td>When checked, allows the user principal to obtain service tickets for other user principals. This attribute is valid only for user principals. When unchecked, the user principal can still obtain service tickets for service principals, but not for other user principals.</td>
</tr>
<tr>
<td>Required Preauthentication</td>
<td>When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through software) that the principal is really the principal that is requesting the TGT. This preauthentication is usually done through an extra password, for example, from a DES card. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.</td>
</tr>
<tr>
<td>Required Hardware Authentication</td>
<td>When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through hardware) that the principal is really the principal that is requesting the TGT. Hardware preauthentication can occur, for example, on a Java ring reader. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.</td>
</tr>
</tbody>
</table>

### TABLE 24–5 Attributes for the Policy Basics Pane of the SEAM Tool

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>The name of the policy. A policy is a set of rules that govern a principal’s password and tickets. If you are modifying a policy, you cannot edit its name.</td>
</tr>
<tr>
<td>Minimum Password Length</td>
<td>The minimum length for the principal’s password.</td>
</tr>
<tr>
<td>Minimum Password Classes</td>
<td>The minimum number of different character types that are required in the principal’s password.</td>
</tr>
<tr>
<td></td>
<td>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 sets no restriction on the number of password character types.</td>
</tr>
<tr>
<td>Saved Password History</td>
<td>The number of previous passwords that have been used by the principal, and a list of the previous passwords that cannot be reused.</td>
</tr>
<tr>
<td>Minimum Password Lifetime</td>
<td>The minimum length of time that the password must be used before it can be changed.</td>
</tr>
</tbody>
</table>
Using the SEAM Tool With Limited Kerberos Administration Privileges

All features of the SEAM Administration Tool are available if your admin principal has all the privileges to administer the Kerberos database. However, you might have limited privileges, such as only being allowed to view the list of principals or to change a principal’s password. With limited Kerberos administration privileges, you can still use the SEAM Tool. However, various parts of the SEAM Tool change based on the Kerberos administration privileges that you do not have. Table 24–6 shows how the SEAM Tool changes based on your Kerberos administration privileges.

The most visual change to the SEAM Tool occurs 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 a policy that you want to manipulate.

If you log in to the SEAM Tool, and you do not have sufficient privileges to perform tasks with it, the following message displays and you are sent back to the SEAM Administration Login window:

Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.

To change the privileges for a principal so that it can administer the Kerberos database, go to “How to Modify the Kerberos Administration Privileges” on page 447.

<table>
<thead>
<tr>
<th>Disallowed Privilege</th>
<th>How the SEAM Tool Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (add)</td>
<td>The Create New and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the add privilege, you cannot create new principals or policies, or duplicate them.</td>
</tr>
<tr>
<td>d (delete)</td>
<td>The Delete button is unavailable in the Principal List and Policy List panels. Without the delete privilege, you cannot delete principals or policies.</td>
</tr>
</tbody>
</table>
TABLE 24–6 Using the SEAM Tool With Limited Kerberos Administration Privileges

<table>
<thead>
<tr>
<th>Disallowed Privilege</th>
<th>How the SEAM Tool Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>m (modify)</td>
<td>The Modify button is unavailable in the Principal List and Policy List panels. Without the modify privilege, you cannot modify principals or policies. Also, with the Modify button unavailable, you cannot modify a principal’s password, even if you have the change password privilege.</td>
</tr>
<tr>
<td>c (change password)</td>
<td>The Password field in the Principal Basics panel is read only and cannot be changed. Without the change password privilege, you cannot modify a principal’s password. Note that even if you have the change password privilege, you must also have the modify privilege to change a principal’s password.</td>
</tr>
<tr>
<td>i (inquiry to database)</td>
<td>The Modify and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the inquiry privilege, you cannot modify or duplicate a principal or a policy. Also, with the Modify button unavailable, you cannot modify a principal’s password, even if you have the change password privilege.</td>
</tr>
<tr>
<td>l (list)</td>
<td>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 the policy that you want to manipulate.</td>
</tr>
</tbody>
</table>

Administering Keytab Files

Every host that provides a service must have a local file, called a keytab (short for “key table”). The keytab contains 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 file that contains its nfs service principal.

To add a service key to a keytab file, you add the appropriate service principal to a host’s keytab file by using the ktadd command of kadmin. Because you are adding a service principal to a keytab file, the principal must already exist in the Kerberos database so that kadmin can verify its existence. On the master KDC, the keytab file is located at /etc/krb5/kadm5.keytab, by default. On application servers that provide 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 keytab files. You should always store keytab files on a local disk, and make them readable only by the root user. Also, you should never send a keytab file over an unsecured network.
There is also a special instance in which to add a root principal to a host's keytab file. If you want a user on the Kerberos client to mount Kerberized NFS file systems that require root-equivalent access, you must add the client's root principal to the client's keytab file. 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 with root access, even when they are using the automounter.

**Note** – When you set up a master KDC, you need to add the `kadmind` and `changepw` principals to the `kadm5.keytab` file.

Another command that you can use to administer keytab files is the `ktutil` command. This interactive command enables you to manage a local host's keytab file without having Kerberos administration privileges, because `ktutil` doesn't interact with the Kerberos database as `kadmind` does. So, after a principal is added to a keytab file, you can use `ktutil` to view the keylist in a keytab file or to temporarily disable authentication for a service.

**Note** – When you change a principal in a keytab file using the `ktadd` command in `kadmind`, a new key is generated and added to the keytab file.

## Administering Keytab Files (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a service principal to a keytab file.</td>
<td>Use the <code>ktadd</code> command of <code>kadmind</code> to add a service principal to a keytab file.</td>
<td>&quot;How to Add a Kerberos Service Principal to a Keytab File&quot; on page 463</td>
</tr>
<tr>
<td>Remove a service principal from a keytab file.</td>
<td>Use the <code>ktremove</code> command of <code>kadmind</code> to remove a service from a keytab file.</td>
<td>&quot;How to Remove a Service Principal From a Keytab File&quot; on page 464</td>
</tr>
<tr>
<td>Display the keylist (list of principals) in a keytab file.</td>
<td>Use the <code>ktutil</code> command to display the keylist in a keytab file.</td>
<td>&quot;How to Display the Keylist (Principals) in a Keytab File&quot; on page 465</td>
</tr>
<tr>
<td>Temporarily disable authentication for a service on a host.</td>
<td>This procedure is a quick way to temporarily disable authentication for a service on a host without requiring <code>kadmind</code> privileges. Before you use <code>ktutil</code> to delete the service principal from the server's keytab file, copy the original keytab file to a temporary location. When you want to enable the service again, copy the original keytab file back to its proper location.</td>
<td>&quot;How to Temporarily Disable Authentication for a Service on a Host&quot; on page 466</td>
</tr>
</tbody>
</table>
How to Add a Kerberos Service Principal to a Keytab File

1. Make sure that the principal already exists in the Kerberos database. See “How to View the List of Kerberos Principals” on page 438 for more information.

2. Become superuser on the host that needs a principal added to its keytab file.

3. Start the kadmin command.
   
   `#/usr/sbin/kadmin`

4. Add a principal to a keytab file by using the ktadd command.

   ```
   kadmin: ktadd [-e enctype] [-k keytab] [-q] [principal | -glob principal-exp]
   ```

   - `-e enctype` Overrides the list of encryption types defined in the krb5.conf file.
   - `-k keytab` Specifies the keytab file. By default, `/etc/krb5/krb5.keytab` is used.
   - `-q` Displays less verbose information.
   - `principal` Specifies the principal to be added to the keytab file. You can add the following service principals: host, root, nfs, and ftp.
   - `-glob principal-exp` Specifies the principal expressions. All principals that match the `principal-exp` are added to the keytab file. The rules for principal expression are the same as for the `list_principals` command of kadmin.

5. Quit the kadmin command.

   `kadmin: quit`

Example 24–16  Adding a Service Principal to a Keytab File

In the following example, the kadmin/admin and kadmin/changepw principals are added to a master KDC’s keytab file. For this example, the keytab file must be the file that is specified in the kdc.conf file.

```bash
kdc1 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/admin kadmin/admin/changepw
```

EnEntry for principal kadmin/admin@example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.

Entry for principal kadmin/admin@example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.

Entry for principal kadmin/admin@example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.

Entry for principal kadmin/admin@example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.

In the following example, denver’s host principal is added to denver’s keytab file, so that the KDC can authenticate denver’s network services.

denver # /usr/sbin/kadmin
kadmin: ktadd host/denver@example.com@EXAMPLE.COM
Entry for principal host/denver@example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver@example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver@example.com with kvno 3, encryption type ARCFOUR with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver@example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit

How to Remove a Service Principal From a Keytab File

1. **Become superuser on the host with a service principal that must be removed from its keytab file.**

2. **Start the kadmin command.**
   ```
   # /usr/sbin/kadmin
   ```

3. **(Optional) To display the current list of principals (keys) in the keytab file, use the ktutil command.**
   See “How to Display the Keylist (Principals) in a Keytab File” on page 465 for detailed instructions.

4. **Remove a principal from the keytab file by using the ktremove command.**
   ```
   kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
   ```
   
   `-k keytab` Specifies the keytab file. By default, /etc/krb5/krb5.keytab is used.
   `-q` Displays less verbose information.
   `principal` Specifies the principal to be removed from the keytab file.
   `kvno` Removes all entries for the specified principal whose key version number matches `kvno`. 
**Removing a Service Principal From a Keytab File**

In the following example, denver’s host principal is removed 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 File**

1. Become superuser on the host with the keytab file.

   **Note** – Although you can create keytab files that are owned by other users, using the default location for the keytab file requires root ownership.

2. Start the `ktutil` command.

   ```
   # /usr/bin/ktutil
   
   ktutil: read_kt keytab
   
   ktutil: list
   
   The current keylist buffer is displayed.
   
   ktutil: quit
   ```

3. Display the keylist buffer by using the `read_kt` command.

4. Display the keylist buffer by using the `list` command.

5. Quit the `ktutil` command.
How to Temporarily Disable Authentication for a Service on a Host

At times, you might need to temporarily disable the authentication mechanism for a service, such as rlogin or ftp, on a network application server. For example, you might want to stop users from logging in to a system while you are performing maintenance procedures. The `ktutil` command enables you to accomplish this task by removing the service principal from the server’s keytab file, without requiring `kadmin` privileges. To enable authentication again, you just need to copy the original keytab file that you saved back to its original location.

**Note** – By default, most services are set up to require authentication. If a service is not set up to require authentication, then the service still works, even if you disable authentication for the service.

1. **Become superuser on the host with the keytab file.**

   **Note** – Although you can create keytab files that are owned by other users, using the default location for the keytab file requires root ownership.

2. **Save the current keytab file to a temporary file.**

3. **Start the `ktutil` command.**

   ```
   # /usr/bin/ktutil
   ```

4. **Read the keytab file 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 that 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
```

Where `slot-number` specifies the slot number of the service principal to be deleted, which is displayed by the `list` command.

7 Write the keylist buffer to a new keytab file by using the `write_kt` command.

```
ktutil: write_kt new-keytab
```

8 Quit the `ktutil` command.

```
ktutil: quit
```

9 Move the new keytab file.

```
# mv new-keytab keytab
```

10 When you want to re-enable the service, copy the temporary (original) keytab file back to its original location.

**Example 24–19 Temporarily Disabling a Service on a Host**

In the following example, the service on the `denver` host is temporarily disabled. To re-enable

the host service on `denver`, you would copy the `krb5.keytab.temp` file to the

```
/etc/krb5/krb5.keytab
```

```
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/new.krb5.keytab
    ktutil: quit

denver # cp /etc/krb5/new.krb5.keytab /etc/krb5/krb5.keytab
```
Using Kerberos Applications (Tasks)

This chapter is intended for anyone on a system with the Kerberos service configured on it. This chapter explains how to use the “Kerberized” commands and services that are provided. You should already be familiar with these commands (in their non-Kerberized versions) before you read about them here.

Because this chapter is intended for the general reader, it includes information on tickets: obtaining, viewing, and destroying them. This chapter also includes information on choosing or changing a Kerberos password.

This is a list of the information in this chapter:
- “Kerberos Ticket Management” on page 469
- “Kerberos Password Management” on page 473
- “Kerberos User Commands” on page 477

For an overview of the Solaris Kerberos product, see Chapter 20.

Kerberos Ticket Management

This section explains how to obtain, view, and destroy tickets. For an introduction to tickets, see “How the Kerberos Service Works” on page 340.

Do You Need to Worry About Tickets?

With any of the SEAM releases or the Solaris 10 release installed, Kerberos is built into the login command, and you will obtain tickets automatically when you log in. The Kerberized commands rsh, rcp, rdist, telnet, and rlogin are usually set up to forward copies of your tickets to the other machines, so you don’t have to explicitly ask for tickets to get access to those machines. Your configuration might not include this automatic forwarding, but it is the default behavior. See “Overview of Kerberized Commands” on page 477 and “Forwarding Kerberos Tickets” on page 480 for more information on forwarding tickets.
For information on ticket lifetimes, see “Ticket Lifetimes” on page 491.

Creating a Kerberos Ticket

Normally, if PAM is configured properly, a ticket is created automatically when you log in, and you need not do anything special to obtain a ticket. However, you might need to create a ticket if your ticket expires. Also, you might need to use a different principal besides your default principal, for example, if you use `rlogin -l` to log in to a machine as someone else.

To create a ticket, use the `kinit` command.

```
% /usr/bin/kinit
```

The `kinit` command prompts you for your password. For the full syntax of the `kinit` command, see the `kinit(1)` man page.

Examples—Creating a Kerberos Ticket

This example shows a user, jennifer, creating a ticket on her own system.

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM: <Type password>
```

Here, the user david creates a ticket that is valid for three hours with the `-l` option.

```
% kinit -l 3h david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG: <Type password>
```

This example shows the user david creating a forwardable ticket (with the `-f` option) 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: <Type password>
```

For more information on how forwarding tickets works, see “Forwarding Kerberos Tickets” on page 480 and “Types of Tickets” on page 489.

Viewing Kerberos Tickets

Not all tickets are alike. One ticket might, for example, be *forwardable*. Another ticket might be *postdated*. While a third ticket might be both forwardable and postdated. You can see which tickets you have, and what their attributes are, by using the `klist` command with the `-f` option:
The following symbols indicate the attributes that are associated with each ticket, as displayed by klist:

A  Preauthenticated
D  Postdatable
d  Postdated
F  Forwardable
f  Forwarded
I  Initial
i  Invalid
P  Proxiable
P  Proxy
R  Renewable

“Types of Tickets” on page 489 describes the various attributes that a ticket can have.

Example—Viewing Kerberos 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 04 15:09:51 09 Mar 04 21:09:51 nfs/EXAMPLE.SUN.COM@EXAMPLE.SUN.COM
renew until 10 Mar 04 15:12:51, Flags: Fdi

The following example shows that the user david has two tickets that were forwarded (f) to his host from another host. The tickets are also forwardable (F).
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting        Expires        Service principal
07 Mar 04 06:09:51   09 Mar 04 23:33:51   host/EXAMPLE.COM@EXAMPLE.COM
          renew until 10 Mar 04 17:09:51, Flags: fF

Valid starting        Expires        Service principal
08 Mar 04 08:09:51   09 Mar 04 12:54:51   nfs/EXAMPLE.COM@EXAMPLE.COM
          renew until 10 Mar 04 15:22:51, Flags: fF

The following example shows how to display the encryption types of the session key and the ticket by using the -e option. The -a option is used to map the host address to a hostname if the name service can do the conversion.

% klist -fea
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting        Expires        Service principal
07 Mar 04 06:09:51   09 Mar 04 23:33:51   krbtgt/EXAMPLE.COM@EXAMPLE.COM
          renew until 10 Mar 04 17:09:51, Flags: FRIA
Etype(skey, tkt): DES cbc mode with RSA-MD5, DES cbc mode with CRC-32
Addresses: client.example.com

Destroying Kerberos Tickets

If you want to destroy all Kerberos tickets acquired during your current session, use the kdestroy command. The command destroys your credential cache, which destroys all your credentials and tickets. While this is not usually necessary, running kdestroy reduces the chance of the credential cache being compromised during times that you are not logged in.

To destroy your tickets, use the kdestroy command.

% /usr/bin/kdestroy

The kdestroy command destroys all your tickets. You cannot use this command 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 use either kdestroy or a screen saver that locks the screen.
Kerberos Password Management

With the Kerberos service configured, 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.

Advice on Choosing a Password

Your password can include almost any character that you can type. The main exceptions are the Control keys and the Return key. A good password is a password that you can remember readily, but no one else can easily guess. Examples of bad passwords include the following:

- Words that can be found in a dictionary
- Any common or popular name
- The name of a famous person or character
- Your name or user name in any form (for example: your name spelled backward, repeated twice, and so forth)
- A spouse’s name, child’s name, or pet’s name
- Your birth date or a relative’s birth date
- Your social security number, driver’s license number, passport number, or other similar identifying number
- Any sample password that appears in this manual or any other manual

A good password is at least eight characters long. Moreover, a password should include a mix of characters, such as uppercase and lowercase letters, numbers, and punctuation marks. Examples of passwords that would be good if they didn’t appear in this manual include the following:

- Acronyms, such as “I2LMLminSF” (which is recalled as “I too left my heart in San Francisco”)
- Easy-to-pronounce nonsense words, such as “WumpaBun” or “WangDangdoodle!”
- Deliberately misspelled phrases, such as “6o’cluck” or “RriotGrrrlsRrrule!”

Caution – Don’t use these examples. Passwords that appear in manuals are the first passwords that an intruder will try.

Changing Your Password

If PAM is properly configured, you can change your Kerberos password in two ways:

- With the usual UNIX passwd command. With the Kerberos service configured, 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 UNIX and Kerberos passwords at the same time. However, you generally 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 might be required to change both passwords in some configurations. For some sites, the UNIX password must be changed, while other sites 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 delay might take anywhere from a few minutes to an hour or more. If you need to get new Kerberos tickets shortly after you change your password, try the new password first. If the new password doesn’t work, try again using the old password.

Kerberos V5 protocol enables 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 Kerberos Policies” on page 449 for more on policies.

For example, suppose that user jennifer’s policy (call it jenpol) mandates that passwords be at least eight letters long and include a mix of at least two types of characters. `kpasswd` will therefore reject an attempt to use “sloth” as a password.

```bash
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password: <Jennifer types 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 types 'sloth'>
New password (again): <Jennifer re-types '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 types of characters (numbers and lowercase letters).

```bash
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
```
Old password: <Jennifer types 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 types ‘slothrop49’>
New password (again): <Jennifer re-types ‘slothrop49’>
Kerberos password changed.

Examples—Changing Your Password

In the following example, user david changes both his UNIX password and Kerberos password with
passwd.

% passwd
  passwd: Changing password for david
    Enter login (NIS+) password:  <Type the current UNIX password>
    New password:  <Type the new UNIX password>
    Re-enter password:  <Confirm the new UNIX password>
    Old KRB5 password:  <Type the current Kerberos password>
    New KRB5 password:  <Type the new Kerberos password>
    Re-enter new KRB5 password:  <Confirm the new Kerberos password>

Note that passwd asks for both the UNIX password and the Kerberos password. This behavior is
established by the default configuration. In that case, user david must use kpasswd to set his
Kerberos password to something else, as shown next.

This example shows user david changing only his Kerberos password with kpasswd.

% kpasswd
  kpasswd: Changing password for david@ENG.EXAMPLE.COM.
    Old password:  <Type the current Kerberos password>
    New password:  <Type the new Kerberos password>
    New password (again):  <Confirm the new Kerberos password>
  Kerberos password changed.

In this example, user david changes the password for the Kerberos principal david/admin (which is
not a valid UNIX user). He must use kpasswd.

% kpasswd david/admin
  kpasswd: Changing password for david/admin.
    Old password:  <Type the current Kerberos password>
    New password:  <Type the new Kerberos password>
    New password (again):  <Type the new Kerberos password>
  Kerberos password changed.
Granting Access to Your Account

If you need to give someone access to log in to your account (as you), you can do so through Kerberos, without revealing your password, by putting a `.k5login` file in your home directory. A `.k5login` file is a list of one or more Kerberos principals corresponding to each person for whom you want to grant access. Each principal must be on a separate line.

Suppose that the user david keeps a `.k5login` file in his home directory that looks like the following:

```
jennifer@ENG.EXAMPLE.COM
joe@EXAMPLE.ORG
```

This file allows the users jennifer and joe to assume david’s identity, provided that they already have Kerberos tickets in their respective realms. For example, jennifer can remotely log in to david’s machine (boston), as him, without having to give his password.

```
rlogin boston -l david
```

**FIGURE 25–1 Using the `.k5login` File to Grant Access to Your Account**

In the case where david’s home directory is NFS-mounted, using Kerberos V5 protocols, from another (third) machine, jennifer must have a forwardable ticket in order to access his home directory. See “Creating a Kerberos Ticket” on page 470 for an example of using a forwardable ticket.

If you will be logging into other machines across a network, you’ll want to include your own Kerberos principal in `.k5login` files on those machines.

Using a `.k5login` file is much safer than giving out your password for these reasons:

- You can take access away any time by removing the principal from your `.k5login` file.
- Although users principals named in the `.k5login` file in your home directory have full access to your account on that machine (or sets of machines, if the `.k5login` file is shared, for example, over NFS). However, any Kerberized services will authorize access based on that user’s identity, not yours. So jennifer can log in to joe’s machine and perform tasks there. However, if she uses a Kerberized program such as `ftp` or `rlogin`, she does so as herself.
Kerberos keeps a log of who obtains tickets, so a system administrator can find out, if necessary, who is capable of using your user identity at a particular time.

One common way to use the .k5login file is to put it in root's home directory, giving root access for that machine to the Kerberos principals listed. This configuration allows system administrators to become root locally, or to log in remotely as root, without having to give out the root password, and without requiring anyone to type the root password over the network.

Example — Using the .k5login File to Grant Access to Your Account

Suppose jennifer decides to log in to the machine boston.example.com as root. Because she has an entry for her principal name in the .k5login file in root's home directory on boston.example.com, she again does not have to type in her password.

```
% rlogin boston.example.com -l root -x
This rlogin session is using DES encryption for all data transmissions.
Last login: Thu Jun 20 16:20:50 from daffodil
SunOS Release 5.7 (GENERIC) #2: Tue Nov 14 18:09:31 EST 1998
boston[root]$
```

Kerberos User Commands

Kerberos V5 product is a single-sign-on system, which means that you only have to type your password once. The Kerberos V5 programs do the authenticating (and optional encrypting) for you, because Kerberos has been built into each of a suite of existing, familiar network programs. The Kerberos V5 applications are versions of existing UNIX network programs with Kerberos features added.

For example, when you use a Kerberized program to connect to a remote host, the program, the KDC, and the remote host perform a set of rapid negotiations. When these negotiations are completed, your program has proven your identity on your behalf to the remote host, and the remote host has granted you access.

Note that Kerberized commands try to authenticate with Kerberos first. If Kerberos authentication fails, an error occurs or UNIX authentication is attempted, depending on what options were used with the command. Refer to the Kerberos Security section in each Kerberos command man page for more detailed information.

Overview of Kerberized Commands

The Kerberized network services are programs that connect to another machine somewhere on the Internet. These programs are the following:

- ftp
- rcp
Kerberos User Commands

- `rdist`
- `rlogin`
- `rsh`
- `ssh`
- `telnet`

These programs have features that transparently use your Kerberos tickets for negotiating authentication and optional encryption with the remote host. In most cases, you’ll notice only that you no longer have to type your password to use them, because Kerberos will provide proof of your identity for you.

The Kerberos V5 network programs include options that enable you to do the following:

- Forward your tickets to the another host (if you initially obtained forwardable tickets).
- Encrypt data transmitted between you and the remote host.

**Note** – This section assumes you are already familiar with the non-Kerberos versions of these programs, and highlights the Kerberos functionality added by the Kerberos V5 package. For detailed descriptions of the commands described here, see their respective man pages.

The following Kerberos options have been added to `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`:

- `-a` Attempts automatic login using your existing tickets. Uses the username as returned by `getlogin()`, unless the name is different from the current user ID. See the `telnet(1)` man page for details.

- `-f` Forwards a *non-forwardable* ticket to a remote host. This option is mutually exclusive with the `-F` option. They cannot be used together in the same command.

You’ll want to forward a ticket if you have reason to believe you’ll need to authenticate yourself to other Kerberos-based services on a third host. For example, you might want to remotely log in to another machine and then remotely log in from it to a third machine.

You should definitely use a forwardable ticket if your home directory on the remote host is NFS-mounted using the Kerberos V5 mechanism. Otherwise, you won’t be able to access your home directory. That is, suppose you initially log in to System 1. From System 1, you remotely log in to your home machine, System 2, which mounts your home directory from System 3. Unless you’ve used the `-f` or `-F` option with `rlogin`, you won’t be able to get to your home directory because your ticket can’t be forwarded to System 3.

By default, `kinit` obtains forwardable ticket-granting tickets (TGTs). However, your configuration might differ in this respect.
For more information on forwarding tickets, see “Forwarding Kerberos Tickets” on page 480.

- **F**
  Forwards a reforwardable copy of your TGT to a remote system. It is similar to - f, but it allows for access to a further (say, fourth or fifth) machine. The - F option can therefore be regarded as being a superset of the - f option. The - F option is mutually exclusive with the - f option. They cannot be used together in the same command.

For more information on forwarding tickets, see “Forwarding Kerberos Tickets” on page 480.

- **k** realm
  Requests tickets for the remote host in the specified realm, instead of determining the realm itself using the krb5.conf file.

- **K**
  Uses your tickets to authenticate to the remote host, but does not automatically log in.

- **m** mechanism
  Specifies the GSS-API security mechanism to use, as listed in the /etc/gss/mech file. Defaults to kerberos_v5.

- **x**
  Encrypts this session.

- **X auth_type**
  Disables the auth-type type of authentication.

The following table shows which commands have specific options. An “X” indicates that the command has that option.

### TABLE 25-1 Kerberos Options for Network Commands

<table>
<thead>
<tr>
<th>Options</th>
<th>ftp</th>
<th>rcp</th>
<th>rlogin</th>
<th>rsh</th>
<th>telnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>- a</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- f</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>- F</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>- k</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- m</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Additionally, `ftp` allows the protection level for a session to be set at its prompt:

clear
  Sets the protection level to “clear” (no protection). This protection level is the default.
private  Sets the protection level to “private.” Data transmissions are confidentiality-protected and integrity-protected by encryption. The privacy service might not be available to all Kerberos users, however.

safe  Sets the protection level to “safe.” Data transmissions are integrity-protected by cryptographic checksum.

You can also set the protection level at the ftp prompt by typing protect followed by any of the protection levels shown above (clear, private, or safe).

### Forwarding Kerberos Tickets

As described in “Overview of Kerberized Commands” on page 477, some commands allow you to forward tickets with either the -f or -F option. Forwarding tickets allows you to “chain” your network transactions. You can, for example, remotely log in to one machine and then remotely log in from it to another machine. The -f option allows you to forward a ticket, while the -F option allows you to reforward a forwarded ticket.

In Figure 25–2, the user david obtains a non-forwardable ticket-granting ticket (TGT) with kinit. The ticket is non-forwardable because he did not specify the -f option. In scenario 1, he is able to remotely log in to machine B, but he can go no further. In scenario 2, the rlogin -f command fails because he is attempting to forward a ticket that is non-forwardable.

1. (On A): kinit david@ACME.ORG
   ```
   rlogin B
   ```

2. (On A): kinit david@ACME.ORG
   ```
   rlogin -f B
   ```

**FIGURE 25–2 Using Non-Forwardable Tickets**

In actuality, Kerberos configuration files are set up so that kinit obtains forwardable tickets by default. However, your configuration might differ. For the sake of explanation, we have assumed that kinit does not obtain forwardable TGTs unless it is invoked with kinit -f. Notice, by the way, that kinit does not have a -F option. TGTs are either forwardable or not.
In Figure 25–3, the user david obtains forwardable TGTs with \texttt{kinit -f}. In scenario 3, he is able to reach machine C because he uses a forwardable ticket with \texttt{rlogin}. In scenario 4, the second \texttt{rlogin} fails because the ticket is not reforwardable. By using the \texttt{-F} option instead, as in scenario 5, the second \texttt{rlogin} succeeds and the ticket can be reforwarded on to machine D.

3. (On A): \texttt{kinit -f david@ACME.ORG}

\begin{center}
\begin{tabular}{c|c|c|c}
A & \texttt{rlogin -f} & B & C
\end{tabular}
\end{center}

4. (On A): \texttt{kinit -f david@ACME.ORG}

\begin{center}
\begin{tabular}{c|c|c|c}
A & \texttt{rlogin -f} & B & \texttt{rlogin} & C
\end{tabular}
\end{center}

5. (On A): \texttt{kinit -f david@ACME.ORG}

\begin{center}
\begin{tabular}{c|c|c|c}
A & \texttt{rlogin -F} & B & \texttt{rlogin -f} & C & \texttt{rlogin} & D
\end{tabular}
\end{center}

\textbf{FIGURE 25–3 Using Forwardable Tickets}

\section*{Examples — Using Kerberized Commands}

The following examples show how the options to the Kerberized commands work.

\textbf{Example — Using the \texttt{-a, -f, and -x} Options With \texttt{telnet}}

In this example, the user david has already logged in, and wants to \texttt{telnet} to the machine \texttt{denver.example.com}. He uses the \texttt{-f} option to forward his existing tickets, the \texttt{-x} option to encrypt the session, and the \texttt{-a} option to perform the login automatically. Because he does not plan to use the services of a third host, he can use \texttt{-f} instead of \texttt{-F}.

\begin{verbatim}
% telnet -a -f -x denver.example.com
Trying 128.0.0.5... Connected to denver.example.com. Escape character is '^['. [ Kerberos V5 accepts you as "david@eng.example.com" ] [ Kerberos V5 accepted forwarded credentials ] SunOS 5.9: Tue May 21 00:31:42 EDT 2004 Welcome to SunOS %
\end{verbatim}
Notice that David's machine used Kerberos to authenticate him to denver.example.com, and logged him in automatically as himself. He had an encrypted session, a copy of his tickets already waiting for him, and he never had to type his password. If he had used a non-Kerberos version of telnet, he would have been prompted for his password, and it would have been sent over the network unencrypted. If an intruder had been watching network traffic at the time, the intruder would have known David's password.

If you forward your Kerberos tickets, telnet (as well as the other commands discussed here) destroys them when it exits.

Example — Using rlogin With the -F Option

Here, the user Jennifer wants to log in to her own machine, boston.example.com. She forwards her existing tickets with the -F option, and encrypts the session with the -x option. She chooses -F rather than -f because after she is logged in to boston, she might want to perform other network transactions requiring tickets to be reforwarded. Also, because she is forwarding her existing tickets, she does not have to type her password.

```
% rlogin boston.example.com -F -x
This rlogin session is using encryption for all transmissions.
Last login Mon May 19 15:19:49 from daffodil
SunOS Release 5.9 (GENERIC) #2 Tue Nov 14 18:09:3 EST 2003
```

Example — Setting the Protection Level in ftp

Suppose that Joe wants to use ftp to get his mail from the directory ~joe/MAIL from the machine denver.example.com, encrypting the session. The exchange would look like the following:

```
% ftp -f denver.example.com
Connected to denver.example.com
220 denver.example.org FTP server (Version 6.0) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded Name (daffodil.example.org:joe)
232 GSSAPI user joe@MELPOMENE.EXAMPLE.COM is authorized as joe
230 User joe logged in.
Remote system type is UNIX.
Using BINARY mode to transfer files.
ftp> protect private
200 Protection level set to Private
ftp> cd ~joe/MAIL
250 CWD command successful.
ftp> get RMAIL
227 Entering Passive Mode (128,0,0,5,16,49)
150 Opening BINARY mode data connection for RMAIL (158336 bytes).
226 Transfer complete. 158336 bytes received in 1.9 seconds (1.4e+02 Kbytes/s)
```
ftp> quit
%

To encrypt the session, joe sets the protection level to private.
This chapter lists many of the files, commands, and daemons that are part of the Kerberos product. In addition, this chapter provides detailed information about how Kerberos authentication works.

This is a list of the reference information in this chapter.

- “Kerberos Files” on page 485
- “Kerberos Commands” on page 487
- “Kerberos Daemons” on page 487
- “Kerberos Terminology” on page 488
- “How the Kerberos Authentication System Works” on page 493
- “Gaining Access to a Service Using Kerberos” on page 494
- “Using Kerberos Encryption Types” on page 497
- “Using the gsscred Table” on page 499
- “Notable Differences Between Solaris Kerberos and MIT Kerberos” on page 499

### Kerberos Files

#### TABLE 26–1 Kerberos Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~/.gkadmin</td>
<td>Default values for creating new principals in the SEAM Administration Tool</td>
</tr>
<tr>
<td>~/.k5login</td>
<td>List of principals that grant access to a Kerberos account</td>
</tr>
<tr>
<td>/etc/krb5/kadm5.acl</td>
<td>Kerberos access control list file, which includes principal names of KDC administrators and their Kerberos administration privileges</td>
</tr>
<tr>
<td>/etc/krb5/kadm5.keytab</td>
<td>Keytab file for the kadmin service on the master KDC</td>
</tr>
</tbody>
</table>
### Kerberos Files

#### TABLE 26–1 Kerberos Files  (Continued)

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/etc/krb5/kdc.conf</code></td>
<td>KDC configuration file</td>
</tr>
<tr>
<td><code>/etc/krb5/kpropd.acl</code></td>
<td>Kerberos database propagation configuration file</td>
</tr>
<tr>
<td><code>/etc/krb5/krb5.conf</code></td>
<td>Kerberos realm configuration file</td>
</tr>
<tr>
<td><code>/etc/krb5/krb5.keytab</code></td>
<td>Keytab file for network application servers</td>
</tr>
<tr>
<td><code>/etc/krb5/warn.conf</code></td>
<td>Kerberos ticket expiration warning and automatic renewal configuration file</td>
</tr>
<tr>
<td><code>/etc/pam.conf</code></td>
<td>PAM configuration file</td>
</tr>
<tr>
<td><code>/tmp/krb5cc_uid</code></td>
<td>Default credentials cache, where <code>uid</code> is the decimal UID of the user</td>
</tr>
<tr>
<td><code>/tmp/ovsec_adm.xxxxxx</code></td>
<td>Temporary credentials cache for the lifetime of the password changing operation, where <code>xxxxxx</code> is a random string</td>
</tr>
<tr>
<td><code>/var/krb5/.k5.REALM</code></td>
<td>KDC stash file, which contains an encrypted copy of the KDC master key</td>
</tr>
<tr>
<td><code>/var/krb5/kadmin.log</code></td>
<td>Log file for <code>kadmin</code></td>
</tr>
<tr>
<td><code>/var/krb5/kdc.log</code></td>
<td>Log file for the KDC</td>
</tr>
<tr>
<td><code>/var/krb5/principal</code></td>
<td>Kerberos principal database</td>
</tr>
<tr>
<td><code>/var/krb5/principal.kadm5</code></td>
<td>Kerberos administrative database, which contains policy information</td>
</tr>
<tr>
<td><code>/var/krb5/principal.kadm5.lock</code></td>
<td>Kerberos administrative database lock file</td>
</tr>
<tr>
<td><code>/var/krb5/principal.ok</code></td>
<td>Kerberos principal database initialization file that is created when the Kerberos database is initialized successfully</td>
</tr>
<tr>
<td><code>/var/krb5/principal.ulog</code></td>
<td>Kerberos update log, which contains updates for incremental propagation</td>
</tr>
<tr>
<td><code>/var/krb5/slave_datatrans</code></td>
<td>Backup file of the KDC that the <code>kprop_script</code> script uses for propagation</td>
</tr>
<tr>
<td><code>/var/krb5/slave_datatrans_slave</code></td>
<td>Temporary dump file that is created when full updates are made to the specified slave</td>
</tr>
</tbody>
</table>
Kerberos Commands

This section lists some commands that are included in the Kerberos product.

TABLE 26–2 Kerberos Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/bin/ftp</td>
<td>File Transfer Protocol program</td>
</tr>
<tr>
<td>/usr/bin/rcp</td>
<td>Remote file copy program</td>
</tr>
<tr>
<td>/usr/bin/rdist</td>
<td>Remote file distribution program</td>
</tr>
<tr>
<td>/usr/bin/rlogin</td>
<td>Remote login program</td>
</tr>
<tr>
<td>/usr/bin/rsh</td>
<td>Remote shell program</td>
</tr>
<tr>
<td>/usr/bin/telnet</td>
<td>Kerberized telnet program</td>
</tr>
<tr>
<td>/usr/lib/krb5/kprop</td>
<td>Kerberos database propagation program</td>
</tr>
<tr>
<td>/usr/sbin/gkadmin</td>
<td>Kerberos database administration GUI program, which is used to manage principals and policies</td>
</tr>
<tr>
<td>/usr/sbin/kadmin</td>
<td>Remote Kerberos database administration program (run with Kerberos authentication), which is used to manage principals, policies, and keytab files</td>
</tr>
<tr>
<td>/usr/sbin/kadmin.local</td>
<td>Local Kerberos database administration program (run without Kerberos authentication and must be run on master KDC), which is used to manage principals, policies, and keytab files</td>
</tr>
<tr>
<td>/usr/sbin/kclient</td>
<td>Kerberos client installation script which is used with or without an installation profile</td>
</tr>
<tr>
<td>/usr/sbin/kdb5_util</td>
<td>Creates Kerberos databases and stash files</td>
</tr>
<tr>
<td>/usr/sbin/kproplog</td>
<td>Lists a summary of update entries in the update log</td>
</tr>
</tbody>
</table>

Kerberos Daemons

The following table lists the daemons that the Kerberos product uses.

TABLE 26–3 Kerberos Daemons

<table>
<thead>
<tr>
<th>Daemon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/sbin/in.ftpd</td>
<td>File Transfer Protocol daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/kadmind</td>
<td>Kerberos database administration daemon</td>
</tr>
</tbody>
</table>
Kerberos Terminology

The following section presents Kerberos terms and their definitions. These terms are used throughout the Kerberos documentation. To grasp Kerberos concepts, an understanding of these terms is essential.

**Kerberos-Specific Terminology**

You need to understand the terms in this section in order to administer KDCs.

The Key Distribution Center or KDC is the component of Kerberos that is responsible for issuing credentials. These credentials are created by using information that is stored in the KDC database. Each realm needs at least two KDCs, a master and at least one slave. All KDCs generate credentials, but only the master KDC handles any changes to the KDC database.

A stash file contains the master key for the KDC. This key is used when a server is rebooted to automatically authenticate the KDC before starting the kadmin and krb5kdc commands. Because this file includes the master key, the file and any backups of the file should be kept secure. The file is created with read-only permissions for root. To keep the file secure, do not change the permissions. If the file is compromised, then the key could be used to access or modify the KDC database.

**Authentication-Specific Terminology**

You need to know the terms in this section to understand the authentication process. Programmers and system administrators should be familiar with these terms.

A client is the software that runs on a user’s workstation. The Kerberos software that runs on the client makes many requests during this process. So, differentiating the actions of this software from the user is important.

---

**TABLE 26–3 Kerberos Daemons (Continued)**

<table>
<thead>
<tr>
<th>Daemon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/lib/krb5/kpropd</td>
<td>Kerberos database propagation daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/krb5kdc</td>
<td>Kerberos ticket processing daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/ktkt_warn</td>
<td>Kerberos ticket expiration warning and automatic renewal daemon</td>
</tr>
<tr>
<td>/usr/sbin/in.rlogind</td>
<td>Remote login daemon</td>
</tr>
<tr>
<td>/usr/sbin/in.rshd</td>
<td>Remote shell daemon</td>
</tr>
<tr>
<td>/usr/sbin/in.telnetd</td>
<td>telnet daemon</td>
</tr>
</tbody>
</table>
The terms server and service are often used interchangeably. To clarify, the term server is used to define the physical system that Kerberos software is running on. The term service corresponds to a particular function that is being supported on a server (for example, ftp or nfs). Documentation often mentions servers as part of a service, but this definition clouds the meaning of the terms. Therefore, the term server refers to the physical system. The term service refers to the software.

The Kerberos product uses two types of keys. One type of key is a password derived key. The password derived key is given to each user principal and is known only to the user and to the KDC. The other type of key used by the Kerberos product is a random key that is not associated with a password and so is not suitable for use by user principals. Random keys are typically used for service principals that have entries in a keytab and session keys generated by the KDC. Service principals can use random keys since the service can access the key in the keytab which allows it to run non-interactively. Session keys are generated by the KDC (and shared between the client and service) to provide secure transactions between a client and a service.

A ticket is an information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains:

- Principal name of the service
- Principal name of the user
- IP address of the user’s host
- Timestamp
- Value which defines the lifetime of the ticket
- Copy of the session key

All of this data is encrypted in the server’s service key. Note, the KDC issues the ticket embedded in a credential described below. After a ticket has been issued, it can be reused until the ticket expires.

A credential is a packet of information that includes a ticket and a matching session key. The credential is encrypted with the requesting principal’s key. Typically, the KDC generates a credential in response to a ticket request from a client.

An authenticator is information used by the server to authenticate the client user principal. An authenticator includes the principal name of the user, a timestamp, and other data. Unlike a ticket, an authenticator can be used once only, usually when access to a service is requested. An authenticator is encrypted by using the session key shared by the client and server. Typically, the client creates the authenticator and sends it with the server's or service's ticket in order to authenticate to the server or service.

## 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 being forwardable to being forwarded. You can view ticket properties with the klist command. See “Viewing Kerberos Tickets” on page 470.

Tickets can be described by one or more of the following terms:
### Kerberos Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwardable/forwarded</td>
<td>A forwardable ticket can be sent from one host to another host, obviating the need for a client to reauthenticate itself. For example, if the user <strong>David</strong> obtains a forwardable ticket while on user <strong>Jennifer</strong>'s machine, he can log in to his own machine without having to get a new ticket (and thus authenticate himself again). See “Examples—Creating a Kerberos Ticket” on page 470 for an example of a forwardable ticket.</td>
</tr>
<tr>
<td>Initial</td>
<td>An initial ticket is a ticket 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 initial in order to assure themselves that the client can demonstrate a knowledge of its secret key. 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.</td>
</tr>
<tr>
<td>Invalid</td>
<td>An invalid ticket is a postdated ticket that has not yet become usable. An invalid ticket will be rejected by an application server until it becomes validated. To be validated, a ticket must be presented to the KDC by the client in a ticket-granting service request, with the <code>VALIDATE</code> flag set, after its start time has passed.</td>
</tr>
<tr>
<td>Postdatable/postdated</td>
<td>A postdated ticket is a ticket that does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to be run late at night, because 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. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the ticket is marked renewable, its lifetime is normally set to be equal to the duration of the full life of the ticket-granting ticket.</td>
</tr>
<tr>
<td>Proxiable/proxy</td>
<td>At times, it is necessary for a principal to allow a service to perform an operation on its behalf. The principal name of the proxy must be specified when the ticket is created. The Solaris release does not support proxiable or proxy tickets. A proxiable 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.</td>
</tr>
</tbody>
</table>
| 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, which is one week. If a client wants to continue to use a ticket, the client 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 10 hours. If the client that is 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 the attributes of tickets, see “Viewing Kerberos Tickets” on page 470.

**Ticket Lifetimes**

Any time a principal obtains a ticket, including a ticket-granting ticket (TGT), the ticket’s lifetime is set as the smallest of the following lifetime values:

- The lifetime value that is specified by the `-l` option of `kinit`, if `kinit` is used to get the ticket. By default, `kinit` used the maximum lifetime value.
- The maximum lifetime value (`max_life`) that is specified in the `kdc.conf` file.
- The maximum lifetime value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/realm`.
- The maximum lifetime value that is specified in the Kerberos database for the user principal that requests the ticket.

Figure 26–1 shows how a TGT’s lifetime is determined and where the four lifetime values come from. Even though this figure 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 that provides the ticket provides a maximum lifetime value (instead of the `krbtgt/realm` principal).
Ticket lifetime = Minimum value of L1, L2, L3, and L4

The renewable ticket lifetime is also determined from the minimum of four values, but renewable lifetime values are used instead, as follows:

- The renewable lifetime value that is 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) that is specified in the kdc.conf file.
- The maximum lifetime renewable value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of kinit, the service principal is krbtgt/realm.
- The maximum lifetime renewable value that is specified in the Kerberos database for the user principal that requests the ticket.

**Kerberos 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 principal names.

### TABLE 26–4: Examples of Kerberos Principal Names

<table>
<thead>
<tr>
<th>Principal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>changepw/kdc1.example.com@EXAMPLE.COM</td>
<td>A principal for the master KDC server that allows access to the KDC when you are changing passwords.</td>
</tr>
</tbody>
</table>
### How the Kerberos Authentication System Works

Applications allow you to log in 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 that is being accessed. A ticket and session key are created by the KDC for all users when they first log in. The ticket and the 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 that runs on a particular server. For example, access to the ftp service on a server named boston requires one credential. 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 requester. When received, the credential is stored in a credential cache.

**Gaining Access to a Service Using Kerberos**

To access a specific service on a specific server, the user must obtain two credentials. The first credential is for the ticket-granting ticket (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. The following sections describe this process in more detail.

**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. No secure information is 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 entry in the database, 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 by 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, by using the private key for the user principal. Because 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.

During this process, a user is normally prompted for a password. If the password the user specifies is the same as the password that was used to build the private key stored in the KDC database, then the client can successfully decrypt the information that is 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.
1. Client requests credential for server.

2. Authentication service sends back credential, which includes Session Key 1 and TGT.

3. Client decrypts credential with entered password.

4. Client decrypts credential with entered password.

TGT = Ticket-granting ticket  
KDC = Key Distribution Center

**FIGURE 26–2 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 494. The client then sends a request to the ticket-granting service, which includes the service principal name, Ticket 1, and an authenticator that was encrypted with Session Key 1. Ticket 1 was originally encrypted by the authentication service by 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 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 by using Session Key 1. Because Session Key 1 is known only to the client and the ticket-granting service, this information is secure and can be safely sent over the network.

4. When the client receives this information packet, the client decrypts the information by 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.
1. Client sends TGT and authenticator encrypted with session key 1 to KDC.

2. Ticket-granting service decrypts TGT and authenticator.

3. Ticket-granting service sends back a credential which includes Session Key 2 and Ticket 2.

4. Client decrypts credential with Session Key 1

**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 494 and “Obtaining a Credential for a Server” on page 495. The client can then send a request to the server including Ticket 2 and another authenticator. The authenticator is encrypted by using Session Key 2.

2. Ticket 2 was encrypted by the ticket-granting service with the service key for the service. Because 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.
Using Kerberos Encryption Types

Encryption types identify which cryptographic algorithms and mode to use when cryptographic operations are performed. The aes, des3-cbc-sha1 and rc4-hmac encryption types enable the creation of keys that can be used for higher strength cryptographic operations. These higher strength operations enhance the overall security of the Kerberos service.

Note – The aes256-cts-hmac-sha1-96 encryption type can be used with the Kerberos service if the unbundled Strong Cryptographic packages are installed.

When a client requests a ticket from the KDC, the KDC must use keys whose encryption type is compatible with both the client and the server. While the Kerberos protocol allows the client to request that the KDC use particular encryption types for the client’s part of the ticket reply, the protocol does not allow the server to specify encryption types to the KDC.

Note – If you have a master KDC installed that is not running the Solaris 10 release, the slave KDCs must be upgraded to the Solaris 10 release before you upgrade the master KDC. A Solaris 10 master KDC will use the new encryption types, which an older slave will not be able to handle.

The following lists some of the issues that must be considered before you change the encryption types.

- The KDC assumes that the first key/enctype associated with the server principal entry in the principal database is supported by the server.
- On the KDC, you should make sure that the keys generated for the principal are compatible with the systems on which the principal will be authenticated. By default, the kadmin command creates keys for all supported encryption types. If the systems that the principal is used on do not
Using Kerberos Encryption Types

support this default set of encryption types, then you should restrict the encryption types when creating a principal. You can restrict the encryption types through use of the -e flag in kadmin addprinc or by setting the supported_enctypes parameter in the kdc.conf file to this subset. The supported_enctypes parameter should be used when most of the systems in a Kerberos realm support a subset of the default set of encryption types. Setting supported_enctypes specifies the default set of encryption types kadmin addprinc uses when it creates a principal for a particular realm. As a general rule, it is best to control the encryption types used by Kerberos using one of these two methods.

- When determining the encryption types a system supports, consider both the version of Kerberos running on the system as well as the cryptographic algorithms supported by the server application for which a server principal is being created. For example, when creating an nfs/hostname service principal, you should restrict the encryption types to the types supported by the NFS server on that host. Note that in the Solaris 10 release, all supported Kerberos encryption types are also supported by the NFS server.

- The master_key_enctype parameter in the kdc.conf file can be used to control the encryption type of the master key that encrypts the entries in the principal database. Do not use this parameter if the KDC principal database has already been created. The master_key_enctype parameter can be used at database creation time to change the default master key encryption type from des-cbc-crc to a stronger encryption type. Make sure that all slave KDCs support the chosen encryption type and that they have an identical master_key_enctype entry in their kdc.conf when configuring the slave KDCs. Also, make sure that the master_key_enctype is set to one of the encryption types in supported_enctypes, if supported_enctypes is set in kdc.conf. If either of these issues are not handled properly, then the master KDC might not be able to work with the slave KDCs.

- On the client, you can control which encryption types the client requests when getting tickets from the KDC through a couple of parameters in krb5.conf. The default_tkt_enctypes parameter specifies the encryption types the client is willing to use when the client requests a ticket-granting ticket (TGT) from the KDC. The TGT is used by the client to acquire other server tickets in a more efficient manner. The effect of setting default_tkt_enctypes is to give the client some control over the encryption types used to protect the communication between the client and KDC when the client requests a server ticket using the TGT (this is called a TGS request). Note, that the encryption types specified in default_tkt_enctypes must match at least one of the principal key encryption types in the principal database stored on the KDC. Otherwise, the TGT request will fail. In most situations, it is best not to set default_tkt_enctypes because this parameter can be a source of interoperability problems. By default, the client code requests that all supported encryption types and the KDC choose the encryption types based on the keys the KDC finds in the principal database.

- The default_tgs_enctypes parameter restricts the encryption types the client requests in its TGS requests, which are used to acquire server tickets. This parameter also restricts the encryption types the KDC uses when creating the session key that the client and server share. For example, if a client wants to only use 3DES encryption when doing secure NFS, you should set default_tgs_enctypes = des3-cbc-sha1. Make sure that the client and server principals have a des-3-cbc-sha1 key in the principal database. As with default_tkt_enctype, it is probably best in most cases not to set this because it can cause interoperability problems if the credentials are not setup properly both on the KDC and the server.
On the server, you can control the encryption types accepted by the server with the `permitted_enctypes` in `kdc.conf`. In addition, you can specify the encryption types used when creating keytab entries. Again, it is generally best not to use either of these methods to control encryption types and instead let the KDC determine the encryption types to use because the KDC does not communicate with the server application to determine which key or encryption type to use.

### Using the `gsscred` Table

The `gsscred` table is used by an NFS server when the server is trying to identify a Kerberos user, if the default mappings are not sufficient. The NFS service uses UNIX IDs to identify users. These IDs are not part of a user principal or a credential. The `gsscred` table provides additional mapping from GSS credentials to UNIX UIDs (from the password file). The table must be created and administered after the KDC database is populated. See “Mapping GSS Credentials to UNIX Credentials” on page 357 for more information.

When a client request comes in, the NFS service tries to map the credential name to a UNIX ID. If the mapping fails, the `gsscred` table is checked.

### Notable Differences Between Solaris Kerberos and MIT Kerberos

The Solaris 10 version of the Kerberos service is based on MIT Kerberos version 1.2.1. The following lists the enhancements included in the Solaris 10 release that are not included in the MIT 1.2.1 version:

- Kerberos support of Solaris remote applications
- Incremental propagation for the KDC database
- Client configuration script
- Localized error messages
- BSM audit record support
- Thread safe use of Kerberos using GSS-API
- Use of the Encryption Framework for cryptography

This version also includes some post MIT 1.2.1 bug fixes. In particular, 1.2.5 btree bug fixes and 1.3 TCP support have been added.
This section provides information on the configuration, management, and use of the Solaris auditing subsystem.
Solaris auditing keeps a record of how the system is being used. The auditing service includes tools to assist with the analysis of the auditing data.

This chapter introduces how auditing works in the Solaris Operating System. The following is a list of the information in this chapter.

- "What Is Auditing?" on page 503
- "How Does Auditing Work?" on page 504
- "How Is Auditing Related to Security?" on page 505
- "Audit Terminology and Concepts" on page 505
- "Solaris Auditing Enhancements in the Solaris 10 Release" on page 510

For planning suggestions, see Chapter 28. For procedures to configure auditing at your site, see Chapter 29. For reference information, see Chapter 30.

What Is Auditing?

Auditing is the collecting of data about the use of system resources. The audit data provides a record of security-related system events. This data can then be used to assign responsibility for actions that take place on a host. Successful auditing starts with two security features: identification and authentication. At each login, after a user supplies a user name and password, a unique audit session ID is generated and associated with the user’s process. The audit session ID is inherited by every process that is started during the login session. Even if a user changes identity within a single session, all user actions are tracked with the same audit session ID. For more details about changing identity, see the su(1M) man page.

The auditing service makes the following possible:

- Monitoring security-relevant events that take place on the host
- Recording the events in a network-wide audit trail
- Detecting misuse or unauthorized activity
- Reviewing patterns of access and the access histories of individuals and objects
Discovering attempts to bypass the protection mechanisms
Discovering extended use of privilege that occurs when a user changes identity

During system configuration, you preselect which classes of audit records to monitor. You can also fine-tune the degree of auditing that is done for individual users.

After audit data is collected, postselection tools enable you to reduce and examine interesting parts of the audit trail. For example, you can choose to review audit records for individual users or specific groups. You can examine all records for a certain type of event on a specific day. Or, you can select records that were generated at a certain time of day.

Systems that install non-global zones can audit all zones identically from the global zone. These systems can also be configured to collect different records in the non-global zones. For more information, see “Auditing and Solaris Zones” on page 564.

How Does Auditing Work?

Auditing generates audit records when specified events occur. Most commonly, events that generate audit records include the following:

- System startup and system shutdown
- Login and logout
- Process creation or process destruction, or thread creation or thread destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of privilege capabilities or role-based access control (RBAC)
- Identification actions and authentication actions
- Permission changes by a process or user
- Administrative actions, such as installing a package
- Site-specific applications

Audit records are generated from three sources:

- By an application
- As a result of an asynchronous event
- As a result of a process system call

Once the relevant event information has been captured, the information is formatted into an audit record. The record is then written to audit files. Complete audit records are stored in binary format. With the Solaris 10 release, audit records can also be logged by the syslog utility.

Audit files that are stored in binary format can be stored in a local partition. The files can also be stored on NFS-mounted file servers. The location can include multiple partitions on the same system, partitions on different systems, or partitions on systems on different but linked networks. The collection of audit files that are linked together is considered an audit trail. Audit records
accumulate in audit files chronologically. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information.

Audit records can also be monitored by using the syslog utility. These audit logs can be stored locally. Or, the logs can be sent to a remote system over the UDP protocol. For more information, see “Audit Files” on page 508.

**How Is Auditing Related to Security?**

Solaris auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. Solaris auditing also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. Users who know that their activities are being audited are less likely to attempt malicious activities.

To protect a computer system, especially a system on a network, requires mechanisms that control activities before system processes or user processes begin. Security requires tools that monitor activities as the activities occur. Security also requires reports of activities after the activities have happened. Initial configuration of Solaris auditing requires that parameters be set before users log in or system processes begin. Most auditing activities involve monitoring current events and reporting those events that meet the specified parameters. How Solaris auditing monitors and reports these events is discussed in detail in Chapter 28 and Chapter 29.

Auditing cannot prevent hackers from unauthorized entry. However, the auditing service can report, for example, that a specific user performed specific actions at a specific time and date. The audit report can identify the user by entry path and user name. Such information can be reported immediately to your terminal and to a file for later analysis. Thus, the auditing service provides data that helps you determine the following:

- How system security was compromised
- What loopholes need to be closed to ensure the desired level of security

**Audit Terminology and Concepts**

The following terms are used to describe the auditing service. Some definitions include pointers to more complete descriptions.

**TABLE 27–1 Solaris Auditing Terms**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit class</td>
<td>A grouping of audit events. Audit classes provide a way to select a group of events to be audited. For more information, see “Audit Classes and Preselection” on page 507.</td>
</tr>
</tbody>
</table>
Table 27-1 Solaris Auditing Terms (Continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit directory</td>
<td>A repository of audit files in binary format. For a description of the types of audit directories, see “Audit Files” on page 508.</td>
</tr>
<tr>
<td>Audit event</td>
<td>A security-related system action that is audited. For ease of selection, events are grouped into audit classes. For a discussion of the system actions that can be audited, see “Audit Events” on page 506.</td>
</tr>
<tr>
<td>Audit policy</td>
<td>A set of auditing options that you can enable or disable at your site. These options include whether to record certain kinds of audit data. The options also include whether to suspend auditable actions when the audit trail is full. For more information, see “Determining Audit Policy” on page 517.</td>
</tr>
<tr>
<td>Audit record</td>
<td>Audit data that is stored in audit files. An audit record describes a single audit event. Each audit record is composed of audit tokens. For more information about audit records, see “Audit Records and Audit Tokens” on page 508.</td>
</tr>
<tr>
<td>Audit token</td>
<td>A field of an audit record or event. Each audit token describes an attribute of an audit event, such as a user, a program, or other object. For descriptions of all the audit tokens, see “Audit Token Formats” on page 571.</td>
</tr>
<tr>
<td>Audit trail</td>
<td>A collection of one or more audit files that store the audit data from all systems that run the auditing service. For more information, see “Audit Trail” on page 568.</td>
</tr>
<tr>
<td>Preselection</td>
<td>Preselection is the choice of which audit classes to monitor before you enable the auditing service. The audit events of preselected audit classes appear in the audit trail. Audit classes that are not preselected are not audited, so their events do not appear in the audit trail. A postselection tool, the audit reduce command, selects records from the audit trail. For more information, see “Audit Classes and Preselection” on page 507.</td>
</tr>
<tr>
<td>Public objects</td>
<td>A public object is a file that is owned by the root user and readable by the world. For example, files in the /etc directory and the /usr/bin directory are public objects. Public objects are not audited for read-only events. For example, even if the file_read (fr) audit class is preselected, the reading of public objects is not audited. You can override the default by changing the public audit policy option.</td>
</tr>
</tbody>
</table>

Audit Events

Security-relevant system actions can be audited. These auditable actions are defined as audit events. Audit events are listed in the /etc/security/audit_event file. Each audit event is defined in the file by an event number, a symbolic name, a short description, and the set of audit classes to which the event belongs. For more information on the audit_event file, see the audit_event(4) man page.

For example, the following entry defines the audit event for the exec() system call:

7: AUE_EXEC: exec (2): ps, ex

When you preselect for auditing either the audit class ps or the audit class ex, then exec() system calls are recorded in the audit trail.
Solaris auditing handles *attributable* and *nonattributable* events. The `exec()` system call can be attributed to a user, so the call is considered an attributable event. Events are nonattributable if the events occur at the kernel-interrupt level. Events that occur before a user is authenticated are also nonattributable. The `na` audit class handles audit events that are nonattributable. For example, booting the system is a nonattributable event.

When the class to which an audit event belongs is preselected for auditing, the event is recorded in the audit trail. For example, when you preselect the `ps` and `na` audit classes for auditing, the `exec()` system calls and system boot actions, among other events, are recorded in the audit trail.

In addition to the audit events that are defined by the Solaris auditing service, third-party applications can generate audit events. Audit event numbers from 32768 to 65535 are available for third-party applications.

## Audit Classes and Preselection

Each audit event belongs to an *audit class* or classes. Audit classes are convenient containers for large numbers of audit events. When you *preselect* a class to be audited, you specify that all the events in that class should be recorded in the audit trail. You can preselect for events on a system and for events initiated by a particular user. After the auditing service is running, you can dynamically add or remove audit classes from the preselected classes.

- **System-wide preselection** – Specify system-wide defaults for auditing in the flags, naflags, and plugin lines in the `audit_control` file. The `audit_control` file is described in “audit_control File” on page 559. See also the `audit_control(4)` man page.

- **User-specific preselection** – Specify additions to the system-wide auditing defaults for individual users in the `audit_user` database.

The audit preselection mask determines which classes of events are audited for a user. The user’s audit preselection mask is a combination of the system-wide defaults and the audit classes that are specified for the user. For a more detailed discussion, see “Process Audit Characteristics” on page 567.

The `audit_user` database can be administered locally or by a name service. The Solaris Management Console provides the graphical user interface (GUI) to administer the database. For details, see the `audit_user(4)` man page.

- **Dynamic preselection** – Specify audit classes as arguments to the `auditconfig` command to add or remove those audit classes from a process or session. For more information, see the `auditconfig(1M)` man page.

A postselection command, `auditreduce`, enables you to select records from the preselected audit records. For more information, see “Examining the Audit Trail” on page 510 and the `auditreduce(1M)` man page.

Audit classes are defined in the `/etc/security/audit_class` file. Each entry contains the audit mask for the class, the name for the class, and a descriptive name for the class. For example, the `ps` and `na` class definitions appear in the `audit_class` file as follows:
There are 32 possible audit classes. The classes include the two global classes: all and no. The audit classes are described in the audit_class(4) man page.

The mapping of audit events to classes is configurable. You can remove events from a class, add events to a class, and create a new class to contain selected events. For the procedure, see “How to Change an Audit Event’s Class Membership” on page 531.

Audit Records and Audit Tokens

Each audit record records the occurrence of a single audited event. The record includes information such as who did the action, which files were affected, what action was attempted, and where and when the action occurred. The following example shows a login audit record:

```
header,01,2,login - local,,2003-10-13 11:23:31.050 -07:00
subject,root,root,other,root,other,378,378,0 0 example_system
text,successful login
return,success,0
```

The type of information that is saved for each audit event is defined by a set of audit tokens. Each time an audit record is created for an event, the record contains some or all of the tokens that are defined for the event. The nature of the event determines which tokens are recorded. In the preceding example, each line begins with the name of the audit token. The content of the audit token follows the name. Together, the four audit tokens comprise the login audit record.

For a detailed description of the structure of each audit token with an example of praudit output, see “Audit Token Formats” on page 571. For a description of the binary stream of audit tokens, see the audit.log(4) man page.

Audit Files

Audit records are collected in audit logs. Solaris auditing provides two output modes for audit logs. Logs that are called audit files store audit records in binary format. The set of audit files from a system or site provide a complete audit record. The complete audit record is called the audit trail.

The syslog utility collects and stores text version summaries of the audit record. A syslog record is not complete. The following example shows a syslog entry for a login audit record:

```
Oct 13 11:24:11 example_system auditd: [ID 6472 audit.notice] login - login ok session 378 by root as root:other
```

A site can store audit records in both formats. You can configure the systems at your site to use binary mode, or to use both modes. The following table compares binary audit records with syslog audit records.
### TABLE 27–2 Comparison of Binary Audit Records With syslog Audit Records

<table>
<thead>
<tr>
<th>Feature</th>
<th>Binary Records</th>
<th>syslog Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Writes to the file system</td>
<td>Uses UDP for remote logging</td>
</tr>
<tr>
<td>Data type</td>
<td>Binary</td>
<td>Text</td>
</tr>
<tr>
<td>Record length</td>
<td>No limit</td>
<td>Up to 1024 characters per audit record</td>
</tr>
<tr>
<td>Location</td>
<td>Stored on local disk, and in directories that are mounted by using NFS</td>
<td>Stored in a location that is specified in the syslog.conf file</td>
</tr>
<tr>
<td>How to configure</td>
<td>Edit audit_control file, and protect and NFS-mount audit directories</td>
<td>Edit audit_control file, and edit syslog.conf file</td>
</tr>
<tr>
<td>How to read</td>
<td>Typically, in batch mode</td>
<td>In real time, or searched by scripts that you have created for syslog</td>
</tr>
<tr>
<td></td>
<td>Browser output in XML</td>
<td>Plain text output</td>
</tr>
<tr>
<td>Completeness</td>
<td>Guaranteed to be complete, and to appear in the correct order</td>
<td>Are not guaranteed to be complete</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Greenwich Mean Time (GMT)</td>
<td>Time on the system that is being audited</td>
</tr>
</tbody>
</table>

Binary records provide the greatest security and coverage. Binary output meets the requirements of security certifications, such as the Common Criteria Controlled Access Protection Profile (CAPP). The records are written to a file system that is protected from snooping. On a single system, all binary records are collected and are displayed in order. The GMT timestamp on binary logs enables accurate comparison when systems on one audit trail are distributed across time zones. The `praudit -x` command enables you to view the records in a browser in XML. You can also use scripts to parse the XML output.

In contrast, the syslog records provide greater convenience and flexibility. For example, you can collect the syslog data from a variety of sources. Also, when you monitor `audit notice` events in the syslog.conf file, the syslog utility logs an audit record summary with the current timestamp. You can use the same management and analysis tools that you have developed for syslog messages from a variety of sources, including workstations, servers, firewalls, and routers. The records can be viewed in real time, and can be stored on a remote system.

By using syslog.conf to store audit records remotely, you protect log data from alteration or deletion by an attacker. On the other hand, when audit records are stored remotely, the records are susceptible to network attacks such as denial of service and spoofed source addresses. Also, UDP can drop packets or can deliver packets out of order. The limit on syslog entries is 1024 characters, so some audit records could be truncated in the log. On a single system, not all audit records are collected. The records might not display in order. Because each audit record is stamped with the local system's date and time, you can not rely on the timestamp to construct an audit trail for several systems.

For more information on audit logs, refer to the following:
Audit Storage

An audit directory holds audit files in binary format. A typical installation uses many audit directories. The contents of all audit directories comprise the audit trail. Audit records are stored in audit directories in the following order:

- **Primary audit directory** – A directory where the audit files for a system are placed under normal conditions
- **Secondary audit directory** – A directory where the audit files for a system are placed if the primary audit directory is full or not available
- **Directory of last resort** – A local audit directory that is used if the primary audit directory and all secondary audit directories are not available

The directories are specified in the `audit_control` file. A directory is not used until a directory that is earlier in the list is full. For an annotated `audit_control` file with a list of directory entries, see Example 29–3.

Examining the Audit Trail

The auditing service provides commands to combine and reduce files from the audit trail. The `auditreduce` command can merge audit files from the audit trail. The command can also filter files to locate particular events. The `praudit` command reads the binary files. Options to the `praudit` command provide output that is suitable for scripting and for browser display.

Solaris Auditing Enhancements in the Solaris 10 Release

Since the Solaris 9 release, the following features have been introduced to Solaris auditing:

- Solaris auditing can use the syslog utility to store audit records in text format. For discussion, see “Audit Files” on page 508. To set up the `audit_control` file to use the syslog utility, see “How to Configure syslog Audit Logs” on page 526.
- The `praudit` command has an additional output format, XML. XML is a standard, portable, processable format. The XML format enables the output to be read in a browser, and provides source for XML scripting for reports. The -x option to the `praudit` command is described in “praudit Command” on page 556.
- The default set of audit classes has been restructured. Audit metaclasses provide an umbrella for finer-grained audit classes. For a list of the default set of classes, see “Definitions of Audit Classes” on page 565.
The `bsmconv` command no longer disables the use of the Stop-A key. The Stop-A event can be audited.

The timestamp in audit records is reported in ISO 8601 format. For information about the standard, see `http://www.iso.org`.

Three audit policy options have been added:

- **public** – Public objects are no longer audited for read-only events. By not auditing public files, the audit log size is greatly reduced. Attempts to read sensitive files are therefore easier to monitor. For more on public objects, see “Audit Terminology and Concepts” on page 505.

- **perzone** – The per zone policy has broad effects. A separate audit daemon runs in each zone. The daemon uses audit configuration files that are specific to the zone. Also, the audit queue is specific to the zone. For details, see the `audittd(1M)` and `auditconfig(1M)` man pages. For more on zones, see “Auditing and Solaris Zones” on page 564. For more on policy, see “How to Plan Auditing in Zones” on page 514.

- **zonename** – The name of the Solaris zone in which an audit event occurred can be included in audit records. For more on zones, see “Auditing and Solaris Zones” on page 564. For a discussion of when to use the option, see “Determining Audit Policy” on page 517.

Five audit tokens have been added:

- The `cmd` token records the list of arguments and the list of environment variables that are associated with a command. For more information, see “`cmd Token`” on page 574.

- The `path_attr` token records the sequence of attribute file objects that are below the `path` token object. For more information, see “`path_attr Token`” on page 580.

- The `privilege` token records the use of privilege on a process. For more information, see “`privilege Token`” on page 581.

- The `uauth` token records the use of authorization with a command or action. For more information, see “`uauth Token`” on page 586.

- The `zonename` token records the name of the non-global zone in which an audit event occurred. The `zonename` audit policy option determines whether the `zonename` token is included in the audit record. For more information, see “`zonename Token`” on page 586.

For overview information, see “Auditing and Solaris Zones” on page 564. To learn about zones, see Part II, “Zones,” in System Administration Guide: Solaris Containers-Resource Management and Solaris Zones.
This chapter describes how to set up the auditing service for your Solaris installation. In particular, the chapter covers issues that you need to consider before you enable the auditing service. The following is a list of the planning information in this chapter:

- “Planning Solaris Auditing (Task Map)” on page 513
- “Determining Audit Policy” on page 517
- “Controlling Auditing Costs” on page 519
- “Auditing Efficiently” on page 521

For an overview of auditing, see Chapter 27. For procedures to configure auditing at your site, see Chapter 29. For reference information, see Chapter 30.

### Planning Solaris Auditing (Task Map)

The following task map points to the major tasks that are required for planning disk space and what events to record.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine auditing strategy for non-global zones</td>
<td>“How to Plan Auditing in Zones” on page 514</td>
</tr>
<tr>
<td>Plan storage space for the audit trail</td>
<td>“How to Plan Storage for Audit Records” on page 515</td>
</tr>
<tr>
<td>Determine who and what to audit</td>
<td>“How to Plan Who and What to Audit” on page 516</td>
</tr>
</tbody>
</table>
Planning Solaris Auditing (Tasks)

You want to be selective about what kinds of activities are audited. At the same time, you want to collect useful audit information. Audit files can quickly grow to fill the available space, so you should allocate enough disk space. You also need to carefully plan who to audit and what to audit.

▼ How to Plan Auditing in Zones

If your system has implemented zones, you have two audit configuration possibilities:

- You can configure non-global zones individually.
- You can configure auditing in the global zone for all zones.

1 Determine if you want to customize auditing in non-global zones.

- If you do not want to customize auditing in non-global zones, go to Step 2.

- If you want to customize auditing in non-global zones, consider the following:
  - You must also configure the global zone.
  
  To collect audit records according to the audit configuration files in the non-global zones, you must set the `perzone` audit policy in the global zone.

  **Note** – If you implement non-global zones with customized name service files, you should set the `perzone` audit policy option. Name service files include `/etc/password`, `/etc/shadow`, and `nsswitch.conf`. For the implications of not setting the `perzone` option, see “Auditing and Solaris Zones” on page 564.

- The audit configuration files in a zone are read by the zone’s audit daemon.
  
  Each zone runs its own audit daemon, has its own audit queue, and collects its own audit log. These operations are computer-intensive.

- Each zone can set all policy options except for `perzone` and `ahlt`. These policy options are set in the global zone.

  If you customize audit configuration files in every zone, use “How to Plan Who and What to Audit” on page 516 to plan for every zone. You can skip the first step. You must also “How to Plan Storage for Audit Records” on page 515 for every zone.

2 Determine if you want a single-image audit trail.

A single-image audit trail treats the systems that are being audited as one machine. The global zone runs the only audit daemon on the system, and collects audit logs for every zone. You customize audit configuration files only in the global zone.
This configuration treats all zones as part of one system. To differentiate zone audit records, you can set the zonename policy. You can then use the `audit reduce` command to select audit events by zone. For an example, see the `audit reduce(1M)` man page.

To plan a single-image audit trail, use “How to Plan Who and What to Audit” on page 516 to plan. Start with the first step. You must also “How to Plan Storage for Audit Records” on page 515.

### How to Plan Storage for Audit Records

The audit trail requires dedicated file space. The dedicated file space for audit files must be available and secure. Each system should have several audit directories that are configured for audit files. You should decide how to configure the audit directories as one of the first tasks before you enable auditing on any systems. The following procedure covers the issues to be resolved when you plan for audit trail storage.

**Before You Begin**

If you are implementing non-global zones, complete “How to Plan Auditing in Zones” on page 514 before using this procedure.

1. **Determine how much auditing your site needs.**
   Balance your site’s security needs against the availability of disk space for the audit trail.
   For guidance on how to reduce space requirements while still maintaining site security, as well as how to design audit storage, see “Controlling Auditing Costs” on page 519 and “Auditing Efficiently” on page 521.

2. **Determine which systems are to be audited.**
   On those systems, allocate space for at least one local audit directory. To specify the audit directories, see Example 29–3.

3. **Determine which systems are to store audit files.**
   Decide which servers are to hold the primary and secondary audit directories. For examples of configuring disks for audit directories, see “How to Create Partitions for Audit Files” on page 533.

4. **Name the audit directories.**
   Create a list of all the audit directories that you plan to use. For the naming conventions, see “Conventions for Binary Audit File Names” on page 568.

5. **Determine which systems are to use which audit directories.**
   Create a map that shows which system should use which audit directory. The map helps you to balance the auditing activity. For an illustration, see Figure 30–1 and Figure 30–2.
How to Plan Who and What to Audit

Before You Begin
If you are implementing non-global zones, complete “How to Plan Auditing in Zones” on page 514 before using this procedure.

1. **Determine if you want a single-image audit trail.**
   
   If you plan to audit individual systems differently, start with the next step. You should complete the rest of the planning steps for every system.

   A single-image audit trail treats the systems that are being audited as one machine. To create a single-image audit trail for a site, every system in the installation should be configured as follows:
   
   - Use the same `audit_warn`, `audit_event`, `audit_class`, and `audit_startup` files as every other system.
   - Use the same `audit_user` database. The database can be in the name service.
   - Have identical `flags`, `naflags`, and `plugin` entries in the `audit_control` file.

2. **Determine the audit policy.**

   Use the `auditconfig -lspolicy` command to see a short description of available policy options. By default, only the `cnt` policy is turned on. For a fuller discussion, see Step 8.

   For the effects of the policy options, see “Determining Audit Policy” on page 517. To set audit policy, see “How to Configure Audit Policy” on page 536.

3. **Determine if you want to modify event-to-class mappings.**

   In many situations, the default mapping is sufficient. However, if you add new classes, change class definitions, or determine that a record of a specific system call is not useful, you might also need to move an event to a different class.

   For an example, see “How to Change an Audit Event’s Class Membership” on page 531.

4. **Determine which audit classes to preselect.**

   The best time to add audit classes or to change the default classes is before you start the auditing service.

   The audit class values of the `flags`, `naflags`, and `plugin` entries in the `audit_control` file apply to all users and processes. The preselected classes determine whether an audit class is audited for success, for failure, or for both.

   To preselect audit classes, see “How to Modify the audit_control File” on page 524.

5. **Determine user exceptions to the system-wide preselected audit classes.**

   If you decide that some users should be audited differently from the system-wide preselected audit classes, modify the individual users’ entries in the `audit_user` database.

   For an example, see “How to Change a User’s Audit Characteristics” on page 528.
6 Determine the minimum free disk space.
When disk space on an audit file system drops below the minfree percentage, the auditd daemon switches to the next available audit directory. The daemon then sends a warning that the soft limit has been exceeded.

To set the minimum free disk space, see Example 29–4.

7 Decide how to manage the audit_warn email alias.
The audit_warn script is run whenever the audit system needs to notify you of a situation that requires administrative attention. By default, the audit_warn script sends email to an audit_warn alias and sends a message to the console.

To set up the alias, see “How to Configure the audit_warn Email Alias” on page 536.

8 Decide what action to take when all the audit directories are full.
By default, when the audit trail overflows, the system continues to work. The system counts the audit records that are dropped, but does not record the events. For greater security, you can disable the cnt policy, and enable the ahtl policy. The ahtl policy stops the system when the audit audit trail overflows.

To configure these policy options, see Example 29–14.

9 Decide whether to collect audit records in syslog format as well as in binary logs.
For overall information, see “Audit Files” on page 508.
For an example, see “How to Configure syslog Audit Logs” on page 526.

Determining Audit Policy
Audit policy determines the characteristics of the audit records for the local system. The policy options are set by a startup script. The bsmconv script, which enables the auditing service, creates the /etc/security/audit_startup script. The audit_startup script executes the auditconfig command to establish audit policy. For details about the script, see the audit_startup(1M) man page.

Most audit policy options are disabled by default to minimize storage requirements and system processing demands. You can dynamically enable and disable audit policy options with the auditconfig command. You can permanently enable and disable the policy options with the audit_startup script.

Use the following table to determine if the needs of your site justify the additional overhead that results from enabling one or more audit policy options.
## TABLE 28–1 Effects of Audit Policy Options

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Description</th>
<th>Why Change the Policy Option?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ahlt</td>
<td>This policy applies to asynchronous events only. When disabled, this policy allows the event to complete without an audit record being generated. When enabled, this policy stops the system when the audit file systems are full. Administrative intervention is required to clean up the audit queue, make space available for audit records, and reboot. This policy can only be enabled in the global zone. The policy affects all zones.</td>
<td>The disabled option makes sense when system availability is more important than security. The enabled option makes sense in an environment where security is paramount.</td>
</tr>
<tr>
<td>argv</td>
<td>When disabled, this policy omits environment variables of an executed program from the exec audit record. When enabled, this policy adds the environment variables of an executed program to the exec audit record. The resulting audit records contain much more detail than when this policy is disabled.</td>
<td>The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have suspicions about the environment variables that are being used in exec programs.</td>
</tr>
<tr>
<td>argv</td>
<td>When disabled, this policy omits the arguments of an executed program from the exec audit record. When enabled, this policy adds the arguments of an executed program to the exec audit record. The resulting audit records contain much more detail than when this policy is disabled.</td>
<td>The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have reason to believe that unusual exec programs are being run.</td>
</tr>
<tr>
<td>cnt</td>
<td>When disabled, this policy blocks a user or application from running. The blocking happens when audit records cannot be added to the audit trail because no disk space is available. When enabled, this policy allows the event to complete without an audit record being generated. The policy maintains a count of audit records that are dropped.</td>
<td>The disabled option makes sense in an environment where security is paramount. The enabled option makes sense when system availability is more important than security.</td>
</tr>
<tr>
<td>group</td>
<td>When disabled, this policy does not add a groups list to audit records. When enabled, this policy adds a groups list to every audit record as a special token.</td>
<td>The disabled option usually satisfies requirements for site security. The enabled option makes sense when you need to audit which groups are generating audit events.</td>
</tr>
<tr>
<td>path</td>
<td>When disabled, this policy records in an audit record at most one path that is used during a system call. When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.</td>
<td>The disabled option places at most one path in an audit record. The enabled option enters each file name or path that is used during a system call in the audit record as a path token.</td>
</tr>
</tbody>
</table>
### TABLE 28–1 Effects of Audit Policy Options (Continued)

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Description</th>
<th>Why Change the Policy Option?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>perzone</strong></td>
<td>When disabled, this policy maintains a single audit configuration for a system. One audit daemon runs in the global zone. Audit events in non-global zones can be located in the audit record by preselecting the zonename audit token. When enabled, this policy maintains separate audit configuration, audit queue, and audit logs for each zone. A separate version of the audit daemon runs in each zone. This policy can be enabled in the global zone only.</td>
<td>The disabled option is useful when you have no special reason to maintain a separate audit log, queue, and daemon for each zone. The enabled option is useful when you cannot monitor your system effectively by simply preselecting the zonename audit token.</td>
</tr>
<tr>
<td><strong>public</strong></td>
<td>When disabled, this policy does not add read-only events of public objects to the audit trail when the reading of files is preselected. Audit classes that contain read-only events include fr, fa, and cl. When enabled, this policy records every read-only audit event of public objects if an appropriate audit class is preselected.</td>
<td>The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.</td>
</tr>
<tr>
<td><strong>seq</strong></td>
<td>When disabled, this policy does not add a sequence number to every audit record. When enabled, this policy adds a sequence number to every audit record. The sequence token holds the sequence number.</td>
<td>The disabled option is sufficient when auditing is running smoothly. The enabled option makes sense when the cnt policy is enabled. The seq policy enables you to determine when data was discarded.</td>
</tr>
<tr>
<td><strong>trail</strong></td>
<td>When disabled, this policy does not add a trailer token to audit records. When enabled, this policy adds a trailer token to every audit record.</td>
<td>The disabled option creates a smaller audit record. The enabled option clearly marks the end of each audit record with a trailer token. The trailer token is often used in conjunction with the sequence token. The trailer token provides easier and more accurate resynchronization of audit records.</td>
</tr>
<tr>
<td><strong>zonename</strong></td>
<td>When disabled, this policy does not include a zonename token in audit records. When enabled, this policy includes a zonename token in every audit record from a non-global zone.</td>
<td>The disabled option is useful when you do not need to compare audit behavior across zones. The enabled option is useful when you want to isolate and compare audit behavior across zones.</td>
</tr>
</tbody>
</table>

### Controlling Auditing Costs

Because auditing consumes system resources, you must control the degree of detail that is recorded. When you decide what to audit, consider the following costs of auditing:

- Cost of increased processing time
- Cost of analysis of audit data
- Cost of storage of audit data
Cost of Increased Processing Time of Audit Data

The cost of increased processing time is the least significant of the costs of auditing. The first reason is that auditing generally does not occur during computation-intensive tasks, such as image processing, complex calculations, and so forth. The other reason is that the cost for single-user systems is usually small enough to ignore.

Cost of Analysis of Audit Data

The cost of analysis is roughly proportional to the amount of audit data that is collected. The cost of analysis includes the time that is required to merge and review audit records. Cost also includes the time that is required to archive the records and keep the records in a safe place.

The fewer records that you generate, the less time that is required to analyze the audit trail.

Upcoming sections, "Cost of Storage of Audit Data" on page 520 and "Auditing Efficiently" on page 521, describe ways to audit efficiently. Efficient auditing reduces the amount of audit data, while still providing enough coverage to achieve your site’s security goals.

Cost of Storage of Audit Data

Storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users
- Number of systems
- Amount of use
- Degree of traceability and accountability that is required

Because these factors vary from site to site, no formula can predetermine the amount of disk space to set aside for audit data storage. Use the following information as a guide:

- Preselect audit classes judiciously to reduce the volume of records that are generated.
  
  Full auditing, that is, with the all class, fills disks quickly. Even a simple task such as compiling a program could generate a large audit file. A program of modest size could generate thousands of audit records in less than a minute.
  
  For example, by omitting the file_read audit class, fr, you can significantly reduce audit volume. By choosing to audit for failed operations only, you can at times reduce audit volume.
  
  For example, by auditing for failed file_read operations, -fr, you can generate far fewer records than by auditing for all file_read events.

- Efficient audit file management is also important. After the audit records are created, file management reduces the amount of storage that is required.

- Understand the audit classes
  
  Before you configure auditing, you should understand the types of events that the classes contain. You can change the audit event-class mappings to optimize audit record collection.
Develop a philosophy of auditing for your site.
Base your philosophy on sensible measures. Such measures include the amount of traceability that your site requires, and the types of users that you administer.

**Auditing Efficiently**

The following techniques can help you achieve your organization’s security goals while auditing more efficiently.

- Randomly audit only a certain percentage of users at any one time.
- Reduce the disk-storage requirements for audit files by combining, reducing, and compressing the files. Develop procedures for archiving the files, for transferring the files to removable media, and for storing the files offline.
- Monitor the audit data in real time for unusual behaviors. You can extend management and analysis tools that you have already developed to handle audit records in syslog files.

You can also set up procedures to monitor the audit trail for certain activities. You can write a script to trigger an automatic increase in the auditing of certain users or certain systems in response to detection of unusual events.

For example, you could write a script that does the following:

1. Monitors the creation of audit files on all the audit file servers.
2. Processes the audit files with the `tail` command.
   - The piping of the output from the `tail -Of` command through the `praudit` command can yield a stream of audit records as the records are generated. For more information, see the `tail(1)` man page.
3. Analyzes this stream for unusual message types or other indicators, and delivers the analysis to the auditor.
   - Or, the script can be used to trigger automatic responses.
4. Constantly monitors the audit directories for the appearance of new `not_terminated` audit files.
5. Terminates outstanding `tail` processes when their files are no longer being written to.
Managing Solaris Auditing (Tasks)

This chapter presents procedures to help you set up and manage a Solaris system that is audited. This chapter also includes instructions for administering the audit trail. The following is a list of the information in this chapter.

- "Solaris Auditing (Task Map)” on page 523
- "Configuring Audit Files (Task Map)” on page 524
- "Configuring and Enabling the Auditing Service (Task Map)” on page 532
- "Managing Audit Records (Task Map)” on page 542

For an overview of the auditing service, see Chapter 27. For planning suggestions, see Chapter 28. For reference information, see Chapter 30.

Solaris Auditing (Task Map)

The following task map points to the major tasks that are required to manage auditing. The tasks are ordered.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Plan for auditing</td>
<td>Contains configuration issues to decide before you configure the auditing service.</td>
<td>&quot;Planning Solaris Auditing (Task Map)” on page 513</td>
</tr>
<tr>
<td>2. Configure audit files</td>
<td>Defines which events, classes, and users require auditing.</td>
<td>&quot;Configuring Audit Files (Task Map)” on page 524</td>
</tr>
<tr>
<td>3. Configure and enable auditing</td>
<td>Configures each host for disk space and other auditing service requirements. Then, starts the auditing service.</td>
<td>&quot;Configuring and Enabling the Auditing Service (Task Map)” on page 532</td>
</tr>
<tr>
<td>4. Manage audit records</td>
<td>Collects and analyzes the audit data.</td>
<td>&quot;Managing Audit Records (Task Map)” on page 542</td>
</tr>
</tbody>
</table>
Configuring Audit Files (Task Map)

The following task map points to the procedures for configuring files to customize auditing at your site. Most of the tasks are optional.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
</table>
| Select audit classes, and customize audit_control settings | Involves:  
  - Preselecting system-wide audit classes  
  - Specifying the audit directories for each system  
  - Setting disk space limits on audit file systems | "How to Modify the audit_control File" on page 524 |
| (Optional) Log audit events in two modes | Enables you to monitor audit events in real time, in addition to storing audit records in binary format. | "How to Configure syslog Audit Logs" on page 526 |
| (Optional) Change audit characteristics for users | Sets user-specific exceptions to the system-wide preselected audit classes. | "How to Change a User's Audit Characteristics" on page 528 |
| (Optional) Add audit classes | Reduces the number of audit records by creating a new audit class to hold events. | "How to Add an Audit Class" on page 530 |
| (Optional) Change event-to-class mappings | Reduces the number of audit records by changing the event-class mapping. | "How to Change an Audit Event's Class Membership" on page 531 |

Configuring Audit Files

Before you enable auditing on your network, you can customize the audit configuration files for your site auditing requirements. You can also restart the auditing service or reboot the local system to read changed configuration files after the auditing service has been enabled. However, the recommended practice is to customize your audit configuration as much as possible before you start the auditing service.

If you have implemented zones, you can choose to audit all zones from the global zone. To differentiate between zones in the audit output, you can set the zonename policy option. Alternatively, to audit non-global zones individually, you can set the per zone policy in the global zone and customize the audit configuration files in the non-global zones. For an overview, see "Auditing and Solaris Zones" on page 564. For planning, see "How to Plan Auditing in Zones" on page 514.

**How to Modify the audit_control File**

The /etc/security/audit_control file configures system-wide auditing. The file determines which events are audited, when audit warnings are issued, and the location of the audit files.
1 **Assume the Primary Administrator role, or become superuser.**
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 **(Optional) Save a backup copy of the audit_control file.**
   
   # cp /etc/security/audit_control /etc/security/audit_control.orig

3 **Modify the audit_control file for your site.**
   Each entry has the following format:

   
<table>
<thead>
<tr>
<th>keyword</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword</td>
<td>Defines the type of line. The types are dir, flags, minfree, naflags, and plugin. The dir line can be repeated.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies data that is associated with the line type.</td>
</tr>
</tbody>
</table>

   For explanations of the keywords, see the following examples. For an example of a plugin entry, see “How to Configure syslog Audit Logs” on page 526.

**Example 29–1 Preselecting Audit Classes for All Users**
The flags line in the audit_control file defines which classes of attributable events are audited for all users on the system. The classes are separated by commas. White space is allowed. In this example, the events in the lo class are audited for all users.

   
   # audit_control file
dir:/var/audit
flags:lo
minfree:20
naflags:lo

   To see which events are in the lo class, read the audit_event file. You can also use the bsmrecord command, as shown in Example 29–22.

**Example 29–2 Preselecting Nonattributable Events**
In this example, all events in the na class, and all login events that are not attributable, are audited.

   
   # audit_control file
dir:/var/audit
flags:lo
minfree:20
naflags:lo,na
Specifying the Location of Binary Audit Data

The `dir` lines in the `audit_control` file list which audit file systems to use for binary audit data. In this example, three locations for binary audit data are defined.

```
# audit_control file
#
# Primary audit directory - NFS-mounted from audit server
# dir:/var/audit/egret.1/files
#
# Secondary audit directory - NFS-mounted from audit server
# dir:/var/audit/egret.2/files
#
# Directory of last resort local directory
dir:/var/audit
flags:lo
minfree:20
naflags:lo,na
plugin:
```

To set up file systems to hold audit binary audit data, see “How to Create Partitions for Audit Files” on page 533.

Changing the Soft Limit for Warnings

In this example, the minimum free-space level for all audit file systems is set so that a warning is issued when only 10 percent of the file system is available.

```
# audit_control file
#
dir:/var/audit/examplehost.1/files
dir:/var/audit/examplehost.2/files
dir:/var/audit/localhost/files
flags:lo
minfree:10
naflags:lo,na
```

The `audit_warn` alias receives the warning. To set up the alias, see “How to Configure the `audit_warn` Email Alias” on page 536.

How to Configure syslog Audit Logs

You can instruct the auditing service to collect only binary audit data, or you can instruct the auditing service to collect binary data and text data. In the following procedure, you collect binary audit data and text audit data. The collected text audit data is a subset of the binary data.
Preselected audit classes must be specified on the flags line or the naflags line of the audit_control file. The text data is a subset of the preselected binary data.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 (Optional) Save a backup copy of the audit_control file.

```
# cp /etc/security/audit_control /etc/security/audit_control.save
```

3 Add a plugin entry.

Plugins in the auditing service implement binary output and syslog output of audit data. The binary plugin is not specified. The syslog plugin must be specified. For more information, see “auditd Daemon” on page 554.

A plugin entry has the following format:

```
plugin: name=value; p_flags=classes
```

- **value**: Lists the name of the plugin to use. Currently, the only valid value is the audit_syslog.so.1 plugin.
- **classes**: Lists a subset of the audit classes that are specified in the flags line and the naflags line.

For more information about the plugin value, see the audit_syslog(5) man page.

4 Add an audit.notice entry to the syslog.conf file.

The entry includes the location of the log file.

```
# cat /etc/syslog.conf

... audit.notice /var/adm/auditlog
```

Text logs should not be stored where the binary audit files are stored. The audit reduce command assumes that all files in an audit partition are binary audit files.

5 Create the log file.

```
# touch /var/adm/auditlog
```

6 Refresh the configuration information for the syslog service.

```
# svcadm refresh system/system-log
```

7 Regularly archive the syslog log files.

The auditing service can generate extensive output. To manage the logs, see the logadm(1M) man page.
Example 29–5  Specifying Audit Classes for syslog Output

In the following example, the syslog utility collects a subset of the preselected audit classes.

```bash
# audit_control file
dir:/var/audit/host.1/files
dir:/var/audit/host.2/files
dir:/var/audit/localhost/files
flags:lo,ss
minfree:10
naflags:lo,na
plugin:name=audit_syslog.so.1; p_flags=-lo,-na,-ss
```

The flags and naflags entries instruct the system to collect all login/logout, nonattributable, and change of system state audit records in binary format. The plugin entry instructs the syslog utility to collect only failed logins, failed nonattributable events, and failed changes of system state.

Example 29–6  Putting syslog Audit Records on a Remote System

You can change the audit.notice entry in the syslog.conf file to point to a remote system. In this example, the name of the local system is example1. The remote system is remote1.

```bash
example1 # cat /etc/syslog.conf
...
audit.notice @remote1
```

The audit.notice entry in the syslog.conf file on the remote1 system points to the log file.

```bash
remote1 # cat /etc/syslog.conf
...
audit.notice /var/adm/auditlog
```

How to Change a User’s Audit Characteristics

Definitions for each user are stored in the audit_user database. These definitions modify, for the specified user, the preselected classes in the audit_control file. The nsswitch.conf file determines if a local file or if a name service database is used. To calculate the user’s final audit preselection mask, see “Process Audit Characteristics” on page 567.

1  Assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2  (Optional) Save a backup copy of the audit_user database.

   # cp /etc/security/audit_user /etc/security/audit_user.orig
3 **Add new entries to the audit_user database.**

In the local database, each entry has the following format:

```
username:always-audit:never-audit
```

- **username** Selects the name of the user to be audited.
- **always-audit** Selects the list of audit classes that should always be audited for the specified user.
- **never-audit** Selects the list of audit classes that should never be audited for the specified user.

You can specify multiple classes by separating the audit classes with commas.

The audit_user entries are in effect at the user’s next login.

**Example 29–7 Changing Which Events Are Audited for One User**

In this example, the audit_control file contains the preselected audit classes for the system:

```
# audit_control file
...
flags:lo,ss
minfree:10
naflags:lo,na
```

The audit_user file shows an exception. When the user jdoe uses a profile shell, that use is audited:

```
# audit_user file
jdoe:pf
```

The audit preselection mask for jdoe is a combination of the audit_user settings with the audit_control settings. The auditconfig -getaudit command shows the preselection mask for jdoe:

```
# auditconfig -getaudit
audit id = jdoe(1234567)
process preselection mask = ss,pf,lo(0x13000,0x13000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 454
```

**Example 29–8 Auditing Users Only, Not the System**

In this example, the login and role activities of four users only are audited on this system. The audit_control file does not preselect audit classes for the system:

```
# audit_control file
...
flags:
minfree:10
naflags:
```
The `audit_user` file preselects two audit classes for four users:

```
# audit_user file
jdoe:lo,pf
kdoe:lo,pf
pdoe:lo,pf
sdoe:lo,pf
```

The following `audit_control` file protects the system from unwarranted intrusion. In combination with the `audit_user` file, this file protects the system more than the first `audit_control` file in this example.

```
# audit_control file
...
flags:
minfree:10
naflags:lo
```

▼ **How to Add an Audit Class**

When you create your own audit class, you can place into it just those audit events that you want to audit for your site. When you add the class on one system, you should copy the change to all systems that are being audited.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **(Optional) Save a backup copy of the `audit_class` file.**
   ```
   # cp /etc/security/audit_class /etc/security/audit_class.orig
   ```

3. **Add new entries to the `audit_class` file.**
   Each entry has the following format:
   ```
   0xnumber:name:description
   ```
   - `0x` Identifies `number` as hexadecimal.
   - `number` Defines the unique audit class mask.
   - `name` Defines the letter name of the audit class.
   - `description` Defines the descriptive name of the audit class.
   The entry must be unique in the file. Do not use existing audit class masks.
Creating a New Audit Class

This example creates a class to hold a small set of audit events. The added entry to the `audit_class` file is as follows:

```
0x01000000:pf:profile command
```

The entry creates a new audit class that is called `pf`. Example 29–10 populates the new audit class.

How to Change an Audit Event’s Class Membership

You might want to change an audit event’s class membership to reduce the size of an existing audit class, or to place the event in a class of its own. When you reconfigure audit event-class mappings on one system, you should copy the change to all systems that are being audited.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **(Optional) Save a backup copy of the `audit_event` file.**
   
   ```
   # cp /etc/security/audit_event /etc/security/audit_event.orig
   ```

3. **Change the class to which particular events belong by changing the `class-list` of the events.**
   Each entry has the following format:

   ```
   number:name:description:class-list
   ```

   - `number` is the audit event ID.
   - `name` is the name of the audit event.
   - `description` typically is the system call or executable that triggers the creation of an audit record.
   - `class-list` is a comma-separated list of audit classes.

Mapping Existing Audit Events to a New Class

This example maps an existing audit event to the new class that was created in Example 29–9. In the `audit_control` file, the binary audit record captures successes and failures of events in the `pf` class. The `syslog` audit log contains only failures of events in the `pf` class.

```bash
# grep pf | /etc/security/audit_class
0x01000000:pf:profile command
# vi /etc/security/audit_event
6180:AUE prof cmd:profile command:ua,as,pf
# vi audit_control
```
flags:lo,pf
plugin:name=audit_syslog.so.1; p_flags=-lo,-pf

Example 29–11 Auditing the Use of setuid Programs

This example creates a class to hold events that monitor calls to the setuid and setgid programs. The `audit_control` entries audit all successful invocations of the events in the st class.

```bash
# vi /etc/security/audit_class
0x00000800:st:setuid class
# vi /etc/security/audit_event
26:AUE_SETGROUPS:setgroups(2):st
27:AUE_SETPRGRP:setprgrp(2):st
40:AUE_SETREUID:setreuid(2):st
41:AUE_SETREGID:setregid(2):st
214:AUE_SETEGID:setegid(2):st
215:AUE_SETEUID:seteuid(2):st
# vi audit_control
...
flags:lo,+st
plugin:name=audit_syslog.so.1; p_flags=-lo,+st
```

Configuring and Enabling the Auditing Service (Task Map)

The following task map points to procedures for configuring and enabling the auditing service. The tasks are ordered.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (Optional) Change the audit configuration files</td>
<td>Selects which events, classes, and users require auditing.</td>
<td>“Configuring Audit Files (Task Map)” on page 524</td>
</tr>
<tr>
<td>2. Create audit partitions</td>
<td>Creates disk space for the audit files, and protects them with file permissions.</td>
<td>“How to Create Partitions for Audit Files” on page 533</td>
</tr>
<tr>
<td>3. Create the audit_warn alias</td>
<td>Defines who should get email warnings when the auditing service needs attention.</td>
<td>“How to Configure the audit_warn Email Alias” on page 536</td>
</tr>
<tr>
<td>4. (Optional) Change audit policy</td>
<td>Defines additional audit data that your site requires.</td>
<td>“How to Configure Audit Policy” on page 536</td>
</tr>
<tr>
<td>5. Enable auditing</td>
<td>Turns on the auditing service.</td>
<td>“How to Enable Auditing” on page 539</td>
</tr>
<tr>
<td>6. (Optional) Disable auditing</td>
<td>Turns off the auditing service.</td>
<td>“How to Disable Auditing” on page 540</td>
</tr>
</tbody>
</table>
Configuring and Enabling the Auditing Service

After the configuration files have been set up for your site, you need to set up disk space for your audit files. You also need to set up other attributes of the auditing service, and then enable the service. This section also contains procedures to refresh the auditing service when you change configuration settings.

When a non-global zone is installed, you can choose to audit the zone exactly as the global zone is being audited. Alternatively, to audit the non-global zone individually, you can modify the audit configuration files in the non-global zone. To customize audit configuration files, see “Configuring Audit Files (Task Map)” on page 524.

How to Create Partitions for Audit Files

The following procedure shows how to create partitions for audit files, as well as the corresponding file systems and directories. Skip steps as necessary, depending on if you already have an empty partition, or if you have already mounted an empty file system.

1. Assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. Determine the amount of disk space that is required.

   Assign at least 200 Mbytes of disk space per host. However, how much auditing you require dictates the disk space requirements. So, your disk space requirements might be far greater than this figure. Remember to include a local partition for a directory of last resort.

3. Create dedicated audit partitions, as needed.

   This step is most easily done during server installation. You can also create the partitions on disks that have not yet been mounted on the server. For complete instructions on how to create the partitions, see Chapter 12, “Administering Disks (Tasks),” in System Administration Guide: Devices and File Systems.

   # newfs /dev/rdsk/cwtxdy sz

   where /dev/rdsk/cwtxdy sz is the raw device name for the partition.
If the local host is to be audited, also create an audit directory of last resort for the local host.

4 Create mount points for each new partition.
   # mkdir /var/audit/server-name.n

   where server-name.n is the name of the server plus a number that identifies each partition. The number is optional, but the number is useful when there are many audit directories.

5 Add entries to automatically mount the new partitions.
   Add a line to the /etc/vfstab file that resembles the following:
   
   /dev/dsk/cwtxdyzz /dev/rdsk/cwtxdyzz /var/audit/server-name.n ufs 2 yes

6 (Optional) Remove the minimum free space threshold on each partition.
   If you use the default configuration, a warning is generated when the directory is 80 percent full. The warning removes the reason to reserve free space on the partition.
   # tunefs -m 0 /var/audit/server-name.n

7 Mount the new audit partitions.
   # mount /var/audit/server-name.n

8 Create audit directories on the new partitions.
   # mkdir /var/audit/server-name.n/files

9 Correct the permissions on the mount points and new directories.
   # chmod -R 750 /var/audit/server-name.n/files

10 On a file server, define the file systems to be made available to other hosts.
   Often, disk farms are installed to store the audit records. If an audit directory is to be used by several systems, then the directory must be shared through the NFS service. Add an entry that resembles the following for each directory to the /etc/dfs/dfstab file:
   
   share -F nfs /var/audit/server-name.n/files

11 On a file server, restart the NFS service.
   If this command is the first share command or set of share commands that you have initiated, the NFS daemons might not be running.

   - If the NFS service is offline, enable the service.
   
   % svcs \*nfs\*
   disabled Nov_02 svc:/network/nfs/rquota:default
   offline Nov_02 svc:/network/nfs/server:default
   # svcadm enable network/nfs/server
If the NFS service is running, restart the service.

```bash
% svcs *
  online  Nov_02 svc:/network/nfs/client:default
  online  Nov_02 svc:/network/nfs/server:default
# svcadm restart network/nfs/server
```

For more information about the NFS service, refer to “Setting Up NFS Services” in *System Administration Guide: Network Services*. For information on managing persistent services, see Chapter 14, “Managing Services (Overview),” in *System Administration Guide: Basic Administration* and the `smf(5)` man page.

**Example 29–12** Creating an Audit Directory of Last Resort

All systems that run the auditing service should have a local file system that can be used if no other file system is available. In this example, a file system is being added to a system that is named `egret`. Because this file system is only used locally, none of the steps for a file server are necessary.

```bash
# newfs /dev/rdsk/c0t2d0
# mkdir /var/audit/egret
# grep egret /etc/vfstab
# tunefs -m 0 /var/audit/egret
# mount /var/audit/egret
# mkdir /var/audit/egret/files
# chmod -R 750 /var/audit/egret/files
```

**Example 29–13** Creating New Audit Partitions

In this example, a new file system is created on two new disks that are to be used by other systems in the network.

```bash
# newfs /dev/rdsk/c0t2d0
# newfs /dev/rdsk/c0t2d1
# mkdir /var/audit/egret.1
# mkdir /var/audit/egret.2
# grep egret /etc/vfstab
# tunefs -m 0 /var/audit/egret.1
# tunefs -m 0 /var/audit/egret.2
# mount /var/audit/egret.1
# mount /var/audit/egret.2
# mkdir /var/audit/egret.1/files
# mkdir /var/audit/egret.2/files
# chmod -R 750 /var/audit/egret.1/files /var/audit/egret.2/files
# grep egret /etc/dfs/dfstab
  share -F nfs /var/audit/egret.1/files
  share -F nfs /var/audit/egret.2/files
# svcadm enable network/nfs/server
```
How to Configure the `audit_warn` Email Alias

The `audit_warn` script generates mail to an email alias that is called `audit_warn`. To send this mail to a valid email address, you can follow one of the options that are described in Step 2:

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Configure the `audit_warn` email alias.**
   Choose one of the following options:
   - **OPTION 1 –** Replace the `audit_warn` email alias with another email account in the `audit_warn` script.
     Change the email alias in the following line of the script:
     ```
     ADDRESS=audit_warn                # standard alias for audit alerts
     ```
   - **OPTION 2 –** Redirect the `audit_warn` email to another mail account.
     In this case, you would add the `audit_warn` email alias to the appropriate mail aliases file. You could add the alias to the local `/etc/mail/aliases` file or to the `mail_aliases` database in the name space. The new entry would resemble the following if the root mail account was made a member of the `audit_warn` email alias:
     ```
     audit_warn: root
     ```

How to Configure Audit Policy

Audit policy determines the characteristics of the audit records for the local host. When auditing is enabled, the contents of the `/etc/security/audit_startup` file determine the audit policy.

You can inspect, enable, or disable the current audit policy options with the the `auditconfig` command. You can also modify the policy options to the `auditconfig` command in the `audit_startup` script to make permanent audit policy changes.

1. **Assume a role that includes the Audit Control profile, or become superuser.**
   To create a role that includes the Audit Control profile and to assign the role to a user, see "Configuring RBAC (Task Map)” on page 188.

2. **Review the audit policy.**
   Before auditing is enabled, the contents of the `audit_startup` file determine the audit policy:
   ```
   #!/bin/sh
   /usr/bin/echo "Starting BSM services."
   /usr/sbin/deallocate -I
   ```
3 **View the available policy options.**

```
$ auditconfig -lspolicy
```

**Note** – The per zone and ahlt policy options can be set only in the global zone.

4 **Enable or disable selected audit policy options.**

```
# auditconfig -setpolicy prefixpolicy

prefix  A prefix value of + enables the policy option. A prefix value of - disables the policy option.
policy   Selects the policy to be enabled or to be disabled.
```

The policy is in effect until the next boot, or until the policy is modified by the `auditconfig -setpolicy` command.

For a description of each policy option, see “Determining Audit Policy” on page 517.

**Example 29–14**

**Setting the cnt and ahlt Audit Policy Options**

In this example, the cnt policy is disabled and the ahlt policy is enabled. With these settings, system use is halted when the audit partitions are full. These settings are appropriate when security is more important than availability. For restrictions on setting this policy, see Step 3.

The following `audit_startup` entries disable the cnt policy option and enable the ahlt policy option across reboots:

```
# cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -I$s
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy -cnt
/usr/sbin/auditconfig -setpolicy +ahlt
```

**Example 29–15**

**Setting the seq Audit Policy Temporarily**

In this example, the auditd daemon is running and the ahlt audit policy has been set. The seq audit policy is added to the current policy. The seq policy adds a sequence token to every audit record. This is useful for debugging the auditing service when audit records are corrupted, or when records are being dropped.
The `+` prefix adds the `seq` option to the audit policy, rather than replaces the current audit policy with `seq`. The `auditconfig` command puts the policy in effect until the next invocation of the command, or until the next boot.

```
$ auditconfig -setpolicy +seq
$ auditconfig -getpolicy
audit policies = ahlt,seq
```

**Example 29–16  Setting the per zone Audit Policy**

In this example, the per zone audit policy is set in the `audit_startup` script in the global zone. When a zone boots, the non-global zone collects audit records according to the audit configuration settings in its zone.

```
$ cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy +perzone
/usr/sbin/auditconfig -setpolicy +cnt
```

**Example 29–17  Changing an Audit Policy**

In this example, the audit daemon is running and audit policy has been set. The `auditconfig` command changes the `ahlt` and `cnt` policies for the duration of the session. With these settings, audit records are dropped, but counted, when the audit file system is full. For restrictions on setting the `ahlt` policy, see Step 3.

```
$ auditconfig -setpolicy +cnt
$ auditconfig -setpolicy -ahlt
$ auditconfig -getpolicy
audit policies = cnt,seq
```

When the changes are put in the `audit_startup` file, the policies are permanently in effect:

```
$ cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy +cnt
```

The `-ahlt` option does not have to be specified in the file, because the `ahlt` policy option is disabled by default. This setting is appropriate when availability is more important than the security that audit records provide.
How to Enable Auditing

This procedure starts the auditing service in the global zone. To enable the auditing service in a non-global zone, see Example 29–18.

Before You Begin

You should perform this procedure after completing the following tasks:

- Planning – “Planning Solaris Auditing (Task Map)” on page 513
- Customizing audit files – “Configuring Audit Files (Task Map)” on page 524
- Setting up audit partitions – “How to Create Partitions for Audit Files” on page 533
- Setting up audit warning messages – “How to Configure the audit_warn Email Alias” on page 536
- Setting audit policy – “How to Configure Audit Policy” on page 536

1 Become superuser and bring the system into single-user mode.

```bash
% su
Password: <Type root password>
# init 5
```

For more information, see the init(1M) man page.

2 Run the script that enables the auditing service.

Go to the /etc/security directory, and execute the bsmconv script there.

```bash
# cd /etc/security
#/bsmconv
```

This script is used to enable the Basic Security Module (BSM).

Shall we continue with the conversion now? [y/n] y

bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation files.

The Basic Security Module is ready.

If there were any errors, please fix them now.
Configure BSM by editing files located in /etc/security.
Reboot this system now to come up with BSM enabled.

For the effects of the script, see the bsmconv(1M) man page.

3 Bring the system into multiuser mode.

```bash
# init 6
```

The startup file /etc/security/audit_startu causes the auditd daemon to run automatically when the system enters multiuser mode.

Another effect of the script is to turn on device allocation. To configure device allocation, see “Managing Device Allocation (Task Map)” on page 79.
Enabling Auditing in a Non-Global Zone

In the following example, the global zone administrator has turned on per zone policy after auditing was enabled in the global zone and after the non-global zone had booted. The zone administrator of the non-global zone has configured the audit files for the zone, and then starts the audit daemon in the zone.

```
zone1# /usr/sbin/audit -s
```

How to Disable Auditing

If the auditing service is no longer required at some point, this procedure returns the system to the system state before auditing was enabled. If non-global zones are being audited, their auditing service is also disabled.

Caution – This command also disables device allocation. Do not run this command if you want to be able to allocate devices. To disable auditing and retain device allocation, see Example 29–19.

1. Become superuser and bring the system into single-user mode.
   ```
   % su
   Password: <Type root password>
   # init 5
   ``
   For more information, see the `init(1M)` man page.

2. Run the script to disable auditing.
   Change to the `/etc/security` directory, and execute the `bsmunconv` script.
   ```
   # cd /etc/security
   # ./bsmunconv
   ``
   Another effect of the script is to disable device allocation.
   For information on the full effect of the `bsmunconv` script, see the `bsmconv(1M)` man page.

3. Bring the system into multiuser mode.
   ```
   # init 6
   ```

Disabling Auditing and Keeping Device Allocation

In this example, the auditing service stops collecting records, but device allocation continues to work. All values from the `flags`, `naflags`, and `plugin` entries in the `audit_control` file are removed, as are all user entries in the `audit_user` file.

Example 29–18  Enabling Auditing in a Non-Global Zone

Example 29–19  Disabling Auditing and Keeping Device Allocation
# audit_control file
...
flags:
minfree:10
naflags:
plugin:

# audit_user file

The auditd daemon runs, but no audit records are kept.

▼ How to Update the Auditing Service

This procedure restarts the auditd daemon when you have made changes to audit configuration files after the daemon has been running.

1 **Assume a role that includes the Audit Control rights profile, or become superuser.**
   To create a role that includes the Audit Control rights profile and assign the role to a user, see “Configuring RBAC (Task Map)” on page 188.

2 **Choose the appropriate command.**
   - **If you modify the naflags line in the audit_control file, change the kernel mask for nonattributable events.**
     $ /usr/sbin/auditconfig -aconf
     You can also reboot.

   - **If you modify other lines in the audit_control file, reread the audit_control file.**
     The audit daemon stores information from the audit_control file internally. To use the new information, either reboot the system or instruct the audit daemon to read the modified file.
     $ /usr/sbin/audit -s

   **Note** – Audit records are generated based on the audit preselection mask that is associated with each process. Executing `audit -s` does not change the masks in existing processes. To change the preselection mask for an existing process, you must restart the process. You can also reboot.

   The `audit -s` command causes the audit daemon to re-read the `audit_control` file. The command changes the generation of the preselection mask for processes spawned by subsequent logins.
If you modify the `audit_event` file or the `audit_class` file while the audit daemon is running, refresh the auditing service.

Read the modified event-class mappings into the system, and ensure that each user who uses the machine is correctly audited.

```
$ auditconfig -conf
$ auditconfig -setumask auid classes

auid   Is the user ID.
classes Are the preselected audit classes.
```

To change audit policy on a running system, see Example 29–15.

**Example 29–20**  
**Restarting the Audit Daemon**

In this example, the system is brought down to single-user mode, then back up to multiuser mode. When the system is brought into multiuser mode, modified audit configuration files are read into the system.

```
# init 5
# init 6
```

**Managing Audit Records (Task Map)**

The following task map points to procedures for selecting, analyzing, and managing audit records.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display the formats of audit records</td>
<td>Shows the kind of information that is collected for an audit event, and the order in which the information is presented.</td>
<td>&quot;How to Display Audit Record Formats&quot; on page 543</td>
</tr>
<tr>
<td>Merge audit records</td>
<td>Combines audit files from several machines into one audit trail.</td>
<td>&quot;How to Merge Audit Files From the Audit Trail&quot; on page 545</td>
</tr>
<tr>
<td>Select records to examine</td>
<td>Selects particular events for study.</td>
<td>&quot;How to Select Audit Events From the Audit Trail&quot; on page 546</td>
</tr>
<tr>
<td>Display audit records</td>
<td>Enables you to view binary audit records.</td>
<td>&quot;How to View the Contents of Binary Audit Files&quot; on page 548</td>
</tr>
<tr>
<td>Clean up incorrectly named audit files</td>
<td>Provides an end timestamp to audit files that were inadvertently left open by the auditing service.</td>
<td>&quot;How to Clean Up a not_terminated Audit File&quot; on page 550</td>
</tr>
<tr>
<td>Prevent audit trail overflow</td>
<td>Prevents the audit file systems from becoming full.</td>
<td>&quot;How to Prevent Audit Trail Overflow&quot; on page 551</td>
</tr>
</tbody>
</table>
Managing Audit Records

By managing the audit trail, you can monitor the actions of users on your network. Auditing can generate large amounts of data. The following tasks show you how to work with all this data.

How to Display Audit Record Formats

To write scripts that can find the audit data that you want, you need to know the order of tokens in an audit event. The bsmrecord command displays the audit event number, audit class, selection mask, and record format of an audit event.

- Put the format of all audit event records in an HTML file.
  The -a option lists all audit event record formats. The -h option puts the list in HTML format that can be displayed in a browser.
  % bsmrecord -a -h > audit.events.html
  When you display the *.html file in a browser, use the browser’s Find tool to find specific records.

  For more information, see the bsmrecord(1M) manpage.

Example 29–21  Displaying the Audit Record Formats of a Program

In this example, the format of all audit records that are generated by the login program are displayed. The login programs include rlogin, telnet, newgrp, role login to the Solaris Management Console, and Solaris Secure Shell.

% bsmrecord -p login
  terminal login
    program  /usr/sbin/login  See login(1)
    /usr/dt/bin/dtlogin  See dtlogin
    event ID 6152  AUE_login
    class lo  (0x00001000)
    header
    subject
    text error message or "successful login"
    return

  login: logout
    program  various  See login(1)
    event ID 6153  AUE_logout
    ...

  newgrp
    program  newgrp  See newgrp login
    event ID 6212  AUE_newgrp_login
    ...

Managing Audit Records
rlogin
program /usr/sbin/login See login(1) - rlogin
event ID 6155 AUE_rlogin
...

SMC: role login
program SMC server See role login
event ID 6173 AUE_role_login
...

/usr/lib/ssh/sshd
program /usr/lib/ssh/sshd See login - ssh
event ID 6172 AUE_ssh
...

telnet login
program /usr/sbin/login See login(1) - telnet
event ID 6154 AUE_telnet
...

Example 29–22  Displaying the Audit Record Formats of an Audit Class

In this example, the format of all audit records in the fd class are displayed.

% bsmrecord -c fd

rmdir
system call rmdir See rmdir(2)
event ID 48 AUE_RMDIR
class fd (0x00000020)
    header
    path
    [attribute]
    subject
    [use_of_privilege]
    return

unlink
system call unlink See unlink(2)
event ID 6 AUE_UNLINK
...

unlinkat
system call unlinkat See openat(2)
event ID 286 AUE_UNLINKAT
...

544  System Administration Guide: Security Services  •  May 2006
How to Merge Audit Files From the Audit Trail

By merging all audit files in all the audit directories, you can analyze the contents of the entire audit trail. The auditreduce command merges all the records from its input files into a single output file. The input files can then be deleted. When the output file is placed in a directory that is named /etc/security/auditserver-name/files, the auditreduce command can find the output file without your specifying the full path.

**Note** - This procedure applies only to binary audit records.

1 **Assume a role that includes the Audit Review profile, or become superuser.**

The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see "Configuring RBAC (Task Map)" on page 188.

2 **Create a directory for storing merged audit files.**

```
# mkdir audit-trail-directory
```

3 **Limit access to the directory.**

```
# chmod 700 audit-trail-directory
# ls -la audit-trail-directory
```

```
drwx------ 3 root sys 512 May 12 11:47 .
drwxr-xr-x 4 root sys 1024 May 12 12:47 ..
```

4 **Merge the audit records in the audit trail.**

Change directories to the `audit-trail-directory` and merge the audit records into a file with a named suffix. All directories that are listed in the `dir` lines of the `audit_control` file on the local system are merged.

```
# cd audit-trail-directory
# auditreduce -Uppercase-option -O suffix
```

The uppercase options to the auditreduce command manipulate files in the audit trail. The uppercase options include the following:

- **-A** Selects all of the files in the audit trail.
- **-C** Selects complete files only. This option ignores files with the suffix `not_terminated`.
- **-M** Selects files with a particular suffix. The suffix can be a machine name, or it can be a suffix that you have specified for a summary file.
- **-O** Creates an audit file with 14-character timestamps for both the start time and the end time, with the suffix `suffix` in the current directory.
### Example 29–23  Copying Audit Files to a Summary File

In the following example, the System Administrator role, sysadmin, copies all files from the audit trail into a merged file.

```
$ whoami
sysadmin
$ mkdir /var/audit/audit_summary.dir
$ chmod 700 /var/audit/audit_summary.dir
$ cd /var/audit/audit_summary.dir
$ auditreduce -A -O All
$ ls *All
20030827183214.20030827214324723.20030827215318.20030827215318.All
```

In the following example, only complete files are copied from the audit trail into a merged file.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O Complete
$ ls *Complete
20030827183214.20030827214217.20030827214217.Complete
```

In the following example, only complete files are copied from the example1 machine into a merged file.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -M example1 -O example1summ
$ ls *summ
20030827183214.20030827214217.20030827214217.example1summ
```

### Example 29–24  Moving Audit Files to a Summary File

The `-D` option to the `auditreduce` command deletes an audit file when you copy it to another location. In the following example, the complete audit files from one system are copied to the summary directory for later examination.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O daily_example1 -D example1
$ ls *example1
20030827183214.20030827214217.daily_example1
```

The audit files from the `example1` system that were the input to the `*daily_example1` file are removed when this command successfully completes.

▼ How to Select Audit Events From the Audit Trail

You can filter audit records for examination. For the complete list of filtering options, see the `auditreduce(1M)` man page.
1 Assume a role that includes the Audit Review profile, or become superuser.

The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see "Configuring RBAC (Task Map)" on page 188.

2 Select the kinds of records that you want from the audit trail, or from a specified audit file.

   auditreduce -lowercase-option argument [optional-file]

   argument Specific argument that a lowercase option requires. For example, the -c option requires an argument of an audit class, such as ua.

   -d Selects all of the events on a particular date. The date format for argument is yyyyMMdd. Other date options, -b and -a, select events before and after a particular date.

   -u Selects all of the events attributable to a particular user. The argument is a user name. Another user option, -e, selects all of the events attributable to an effective user ID.

   -c Selects all of the events in a preselected audit class. The argument is an audit class name.

   -m Selects all of the instances of a particular audit event. The argument is an audit event.

   optional-file Is the name of an audit file.

Example 29–25 Combining and Reducing Audit Files

The auditreduce command can eliminate the less interesting records as it combines the input files. For example, you might use the auditreduce command to retain only the login and logout records in audit files that are over a month old. If you need to retrieve the complete audit trail, you could recover the trail from backup media.

   # cd /var/audit/audit_summary.dir
   # auditreduce -O lo.summary -b 20030827 -c lo; compress *lo.summary

Example 29–26 Copying na Audit Records to a Summary File

In this example, all the records of nonattributable audit events in the audit trail are collected into one file.

   $ whoami
   sysadmin
   $ cd /var/audit/audit_summary.dir
   $ auditreduce -c na -O nasumm
   $ ls *nasumm
   20030827183214.20030827215318.nasumm
The merged nasumm audit file is time stamped with the beginning and ending date of the na records.

**Example 29–27** Finding Audit Events in a Specified Audit File

You can select audit files manually to search just the named set of files. For example, you can further process the *nasumm file in the previous example to find system boot events. To do so, you would specify the file name as the final argument to the auditreduce command.

```
$ auditreduce -m 113 -O systemboot 20030827183214.20030827215318.nasumm
```

The 20030827183214.20030827183214.systemboot file contains only system boot audit events.

**Example 29–28** Copying One User’s Audit Records to a Summary File

In this example, the records in the audit trail that contain the name of a particular user are merged. The -e option finds the effective user. The -u option finds the audit user.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -e tamiko -O tamiko
```

You can look for specific events in this file. In the following example, what time the user logged in and out on Sept 7, 2003, your time, is checked. Only those files with the user’s name as the file suffix are checked. The short form of the date is yyyyymmdd.

```
# auditreduce -M tamiko -O tamikolo -d 20030907 -u tamiko -c lo
```

**Example 29–29** Copying Selected Records to a Single File

In this example, login and logout messages for a particular day are selected from the audit trail. The messages are merged into a target file. The target file is written in a directory other than the normal audit root directory.

```
# auditreduce -c lo -d 20030827 -O /var/audit/audit_summary.dir/logins
# ls /var/audit/audit_summary.dir/*logins
```

```
/var/audit/audit_summary.dir/20030827183936.20030827232326.logins
```

**How to View the Contents of Binary Audit Files**

The praudit command enables you to view the contents of binary audit files. You can pipe the output from the auditreduce command, or you can read a particular audit file. The -x option is useful for further processing.
1 Assume a role that includes the Audit Review profile, or become superuser.

The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see “Configuring RBAC (Task Map)” on page 188.

2 Use one of the following praudit commands to produce the output that is best for your purposes.

The following examples show praudit output from the same audit event. Audit policy has been set to include the sequence and trailer tokens.

- The praudit -s command displays audit records in a short format, one token per line. Use the -l option to place each record on one line.

```
$ auditreduce -c lo | praudit -s
header,101,2,AUE_rlogin,,example1,2003-10-13 11:23:31.050 -07:00
subject,jdoe,jdoe,staff,jdoe,staff,749,749,195 1234 server1
text,successful login
return,success,0
sequence,1298
```

- The praudit -r command displays audit records in their raw format, one token per line. Use the -l option to place each record on one line.

```
$ auditreduce -c lo | praudit -r
21,101,2,6155,0x0000,192.168.60.83,1062021202,64408258
36,2026700,2026700,10,2026700,10,749,749,195 1234 192.168.60.17
40,successful login
39,0,0
47,1298
```

- The praudit -x command displays audit records in XML format, one token per line. Use the -l option to place the XML output for one record on one line.

```
$ auditreduce -c lo | praudit -x
<record version="2" event="login - rlogin" host="example1" time="Wed Aug 27 14:53:22 PDT 2003" msec="64">
<subject audit-uid="jdoe" uid="jdoe" gid="staff" ruid="jdoe" rgid="staff" pid="749" sid="749" tid="195 1234 server1"/>
<text>successful login</text>
<return errval="success" retval="0"/>
<sequence seq-num="1298"/>
</record>
```

Example 29–30 Printing the Entire Audit Trail

With a pipe to the lp command, the output for the entire audit trail goes to the printer. The printer should have limited access.

```
# auditreduce | praudit | lp -d example.protected.printer
```
Example 29–31  Viewing a Specific Audit File

In this example, a summary login file is examined in a terminal window.

```
# cd /var/audit/audit_summary.dir/logins
# praudit 20030827183936.20030827232326.logins | more
```

Example 29–32  Putting Audit Records in XML Format

In this example, the audit records are converted to XML format.

```
# praudit -x 20030827183214.20030827215318.logins > 20030827.logins.xml
```

The *xml file can be displayed in a browser. The contents of the file can be operated on by a script to extract the relevant information.


\section*{How to Clean Up a not_terminated Audit File}

Occasionally, an audit daemon exits while its audit file is still open. Or, a server becomes inaccessible and forces the machine to switch to a new server. In such instances, an audit file remains with the string not_terminated as the end timestamp, even though the file is no longer used for audit records. Use the auditreduce -O command to give the file the correct timestamp.

1  List the files with the not_terminated string on your audit file system in order of creation.
```
# ls -Rlt audit-directory*/files/* | grep not_terminated
```

- Lists files in subdirectories.
- Lists files from most recent to oldest.
- Lists the files in one column.

2  Clean up the old not_terminated file.
```
# auditreduce -O system-name old-not-terminated-file
```

3  Remove the old not_terminated file.
```
# rm system-name old-not-terminated-file
```

Example 29–33  Cleaning Up Closed not_terminated Audit Files

In the following example, not_terminated files are found, renamed, then the originals are removed.

```
ls -Rlt */files/* | grep not_terminated
.../egret.1/20030908162220.not_terminated.egret
.../egret.1/20030827215359.not_terminated.egret
```
The start timestamp on the new file reflects the time of the first audit event in the `not_terminated` file. The end timestamp reflects the time of the last audit event in the file.

### How to Prevent Audit Trail Overflow

If your security policy requires that all audit data be saved, do the following:

1. **Set up a schedule to regularly archive audit files.**
   - Archive audit files by backing up the files to offline media. You can also move the files to an archive file system.
   - If you are collecting text audit logs with the `syslog` utility, archive the text logs. For more information, see the `logadm(1M)` man page.

2. **Set up a schedule to delete the archived audit files from the audit file system.**

3. **Save and store auxiliary information.**
   - Archive information that is necessary to interpret audit records along with the audit trail.

4. **Keep records of which audit files have been archived.**

5. **Store the archived media appropriately.**

6. **Reduce the volume of audit data that you store by creating summary files.**
   - You can extract summary files from the audit trail by using options to the `auditreduce` command. The summary files contain only records for specified types of audit events. To extract summary files, see Example 29–25 and Example 29–29.
Solaris Auditing (Reference)

This chapter describes the important components of Solaris auditing. The following is a list of the reference information in this chapter.

- “Audit Commands” on page 553
- “Files Used in the Auditing Service” on page 558
- “Rights Profiles for Administering Auditing” on page 563
- “Auditing and Solaris Zones” on page 564
- “Audit Classes” on page 564
- “Audit Policy” on page 567
- “Process Audit Characteristics” on page 567
- “Audit Trail” on page 568
- “Conventions for Binary Audit File Names” on page 568
- “Audit Record Structure” on page 569
- “Audit Token Formats” on page 571

For an overview of Solaris auditing, see Chapter 27. For planning suggestions, see Chapter 28. For procedures to configure auditing at your site, see Chapter 29.

Audit Commands

This section provides information about the following commands:

- “auditd Daemon” on page 554
- “audit Command” on page 554
- “bsmrecord Command” on page 555
- “auditreduce Command” on page 555
- “praudit Command” on page 556
- “auditconfig Command” on page 558
auditd Daemon

The following list summarizes what the auditd daemon does.

- The auditd daemon opens and closes audit files in the directories that are specified in the audit_control file. The files are opened in order.
- The auditd daemon loads one or more plugins. Sun provides two plugins. The /lib/security/audit_binfile.so.1 plugin writes binary audit data to a file. The /lib/security/audit_syslog.so.1 plugin sends text summaries of audit records to the syslogd daemon.
- The auditd daemon reads audit data from the kernel and outputs the data by using an auditd plugin.
- The auditd daemon executes the audit_warn script to warn of configuration errors. The binfile.so.1 plugin executes the audit_warn script. The script, by default, sends warnings to the audit_warn email alias and to the console. The syslog.so.1 plugin does not execute the audit_warn script.
- By default, when all audit directories are full, processes that generate audit records are suspended. In addition, the auditd daemon writes a message to the console and to the audit_warn email alias. At this point, only the system administrator can fix the auditing service. The administrator can log in to write audit files to offline media, delete audit files from the system, and do other cleanup tasks.

The audit policy can be reconfigured with the auditconfig command.

The auditd daemon can be started automatically when the system is brought up to multiuser mode. Or, you can start the daemon from the command line. When the auditd daemon is started, it calculates the amount of free space necessary for audit files.

The auditd daemon uses the list of audit directories in the audit_control file as possible locations for creating audit files. The daemon maintains a pointer into this list of directories, starting with the first directory. Every time the auditd daemon needs to create an audit file, the daemon puts the file into the first available directory in the list. The list starts at the auditd daemon’s current pointer. You can reset the pointer to the beginning of the list by running the audit -s command. The audit -n command instructs the daemon to switch to a new audit file. The new file is created in the same directory as the current file.

audit Command

The audit command controls the actions of the auditd daemon. The audit command can do the following tasks:

- Enable and disable auditing
- Reset the auditd daemon
- Adjust the auditing preselection mask on the local system
- Write audit records to a different audit file

For a discussion of the available options, see the audit(1M) man page.
bsmrecord **Command**

The bsmrecord command displays the format of audit events that are defined in the `/etc/security/audit_event` file. The output includes the event’s audit ID, audit class, audit flag, and the record’s audit tokens in order. With no option, the bsmrecord output displays in a terminal window. With the `-h` option, the output is suitable for viewing in a browser. For examples of the use of the bsmrecord command, see “How to Display Audit Record Formats” on page 543. Also, see the bsmrecord(1M) man page.

**auditreduce Command**

The auditreduce command summarizes audit records that are stored in binary format. The command can merge audit records from one or more input audit files. The command can also be used to perform a post selection of audit records. The records remain in binary format. To merge the entire audit trail, run this command on the audit server. The audit server is the system that mounts all the audit file systems for the installation. For more information, see the auditreduce(1M) man page.

The auditreduce command enables you to track all audited actions on multiple systems from a single location. The command can read the logical combination of all audit files as a single audit trail. You must identically configure all systems at a site for auditing, and create servers and local directories for the audit files. The auditreduce command ignores how the records were generated or where the records are stored. Without options, the auditreduce command merges audit records from all the audit files in all of the subdirectories in the audit root directory. Typically, `/etc/security/audit` is the audit root directory. The auditreduce command sends the merged results to standard output. You can also place the results into a single, chronologically ordered output file. The file contains binary data.

The auditreduce command can also select particular types of records for analysis. The merging functions and selecting functions of the auditreduce command are logically independent. The auditreduce command captures data from the input files as the records are read, before the files are merged and then written to disk.

By specifying options to the auditreduce command, you can also do the following:

- Request audit records that were generated by specified audit classes
- Request audit records that were generated by one particular user
- Request audit records that were generated on specific dates

With no arguments, the auditreduce command checks the subdirectories within the `/etc/security/audit` directory, the default audit root directory. The command checks for a files directory in which the `start-time.end-time.hostname` files reside. The auditreduce command is very useful when audit data resides in separate directories. Figure 30–1 illustrates audit data in separate directories for different hosts. Figure 30–2 illustrates audit data in separate directories for different audit servers.
If the partition for the `/etc/security/audit` directory is very small, you might not store audit data in the default directory. You can pass the `auditreduce` command another directory by using the `-R` option:

```
# auditreduce -R /var/audit-alt
```

You can also specify a particular subdirectory by using the `-S` option:

```
# auditreduce -S /var/audit-alt/host1
```

For other options and more examples, see the `auditreduce(1M)` man page.

**praudit Command**

The `praudit` command makes the binary output of the `auditreduce` command readable. The `praudit` command reads audit records in binary format from standard input and displays the
records in a presentable format. The input can be piped from the audit reduce command or from a single audit file. Input can also be produced with the cat command to concatenate several files, or the tail command for a current audit file.

The praudit command can generate four output formats. A fifth option, -l (long), prints one audit record per line of output. The default is to place one audit token per line of output. The -d option changes the delimiter that is used between token fields and between tokens. The default delimiter is a comma.

- **Default** – The praudit command with no options displays one audit token per line. The command displays the audit event by its description, such as the ioctl(2) system call. Any value that can be displayed as text is displayed in text format. For example, a user is displayed as the user name, not as the user ID.

- **-r option** – The raw option displays as a number any value that could be numeric. For example, a user is displayed by user ID, Internet addresses are in hexadecimal format, and modes are in octal format. The audit event is displayed as its event number, such as 158.

- **-s option** – The short option displays the audit event by its table name, for example, AUE_IOCTL. The option displays the other tokens as the default option displays them.

- **-x option** – The XML option displays the audit record in XML format. This option is useful as input to browsers, or as input to scripts that manipulate XML. The XML is described by a DTD that the auditing service provides. Solaris software also provides a style sheet. The DTD and the style sheet are in the /usr/share/lib/xml directory.

In the default output format of the praudit command, each record is easily identified as a sequence of audit tokens. Each token is presented on a separate line. Each record begins with a header token. You could, for example, further process the output with the awk command.

Here is the output from the praudit -l command for a header token:

```
header,173,2,settppriv(2),,example1,2003-10-13 13:46:02.174 -07:00
```

Here is the output from the praudit -r command for the same header token:

```
121,173,2,289,0x0000,192.168.86.166,1066077962,174352445
```

**EXAMPLE 30-1 Processing praudit Output With a Script**

You might want to process output from the praudit command as lines of text. For example, you might want to select records that the audit reduce command cannot select. You can use a simple shell script to process the output of the praudit command. The following simple example script puts one audit record on one line, searches for a user-specified string, then returns the audit file to its original form.

```
#!/bin/sh

# # This script takes an argument of a user-specified string.
# The sed command prefixes the header tokens with Control-A
# The first tr command puts the audit tokens for one record
```
EXAMPLE 30–1 Processing praudit Output With a Script (Continued)

# onto one line while preserving the line breaks as Control-A
#
praudit | sed -e '1,2d' -e 's/\$s/\$//' -e 's/\$s/\$header/\$aheader/' \ 
| tr '\012\001' '\002\012' \ 
| grep "$1" \ 
| tr '\002' '\012' \ Finds the user-specified string
| tr '\012' '\002' \ Restores the original newline breaks

Note that the ^a in the script is Control-A, not the two characters ^ and a. The prefix distinguishes the header token from the string header that might appear as text.

auditconfig Command

The auditconfig command provides a command-line interface to retrieve and set audit configuration parameters. The auditconfig command can do the following tasks:

- Display, check, and configure audit policy
- Determine if auditing is turned on or turned off
- Turn auditing off and turn auditing on
- Manage the audit directory and the audit file
- Manage the audit queue
- Get and set preselection masks
- Get and set audit event to audit class mappings
- Get and set configuration information, such as session ID and audit ID
- Configure audit characteristics for a process, a shell, and a session
- Reset audit statistics

For a discussion of the command options, see the auditconfig(1M) man page.

Files Used in the Auditing Service

The auditing service uses the following files:

- "system File" on page 559
- "syslog.conf File" on page 559
- "audit_class File" on page 559
- "audit_control File" on page 559
- "audit_event File" on page 561
- "audit_startup Script" on page 561
- "audit_user Database" on page 561
- "audit_warn Script" on page 562
- "bsmconv Script" on page 563
system File

The /etc/system file contains commands that the kernel reads during initialization to customize the system operations. The bsmconv and bsmunconv shell scripts, which are used to activate and deactivate auditing, modify the /etc/system file. The bsmconv shell script adds the following line to the /etc/system file:

```
set c2audit:audit_load=1
```

The set c2audit:audit_load=1 entry causes the kernel module for auditing to be loaded when the system is booted. The bsmunconv shell script disables auditing when the system is rebooted. The command removes the c2audit line from the /etc/system file.

syslog.conf File

The /etc/syslog.conf file works with the audit_control file to store audit records in text format. The syslog.conf file can be configured to enable the syslog utility to store audit records. For an example, see "How to Configure syslog Audit Logs" on page 526.

audit_class File

The /etc/security/audit_class file defines the audit classes. Audit classes are groups of audit events. You use the class name in the audit_control file to preselect the classes whose events you want to audit. The classes accept prefixes to select only failed events or only successful events. For more information, see "Audit Class Syntax" on page 566.

The superuser, or an administrator in an equivalent role, can modify the definitions of audit classes. This administrator can define new audit classes, rename existing classes, or otherwise change existing classes by editing the audit_class file in a text editor. For more information, see the audit_class(4) man page.

audit_control File

The /etc/security/audit_control file on each system contains configuration information for the auditd daemon. The file enables every system to mount a remote audit file system to store their audit records.

You can specify five kinds of information in the audit_control file. Each line of information begins with a keyword.

- **flags keyword** – Begins the entry that preselects which classes of events are audited for all users on the system. The audit classes that are specified here determine the system-wide audit preselection mask. The audit classes are separated by commas.
flags keyword – Begins the entry that preselects which classes of events are audited when an action cannot be attributed to a specific user. The audit classes are separated by commas. The na event class belongs in this entry. The naflags entry can be used to log other event classes that are normally attributable but cannot be attributed. For example, if a program that starts at boot reads a file, then an fr in the naflags entry would create a record for that event.

minfree keyword – Begins the entry that defines the minimum free-space level for all audit file systems. The minfree percentage must be equal to 0 or greater than 0. The default is 20 percent. When an audit file system is 80 percent full, the audit data is then stored in the next available audit directory. For more information, see the audit_warn(1M) man page.

dir keyword – Begins the directory definition lines. Each line defines an audit file system and directory that the system uses to store its audit files. You can define one or more directory definition lines. The order of the dir lines is significant. The auditd daemon creates audit files in the directories in the specified order. The first directory is the primary audit directory for the system. The second directory is the secondary audit directory where the auditd daemon creates audit files when the first directory becomes full, and so on. For more information, see the audit(1M) man page.

plugin keyword – Specifies the plugin path and the audit classes for the syslog plugin module. The module provides real-time conversion of Solaris audit records to text. The audit classes in the plugin line must be a subset of the audit classes in the flags line and the naflags line.

For more information about the audit_control file, see the audit_control(4) man page.

EXAMPLE 30–2 Sample audit_control File

The following is a sample audit_control file for the system noddy. noddy uses two audit file systems on the audit server blinken, and a third audit file system that is mounted from the second audit server winken. The third file system is used only when the audit file systems on blinken become full or unavailable. The minfree value of 20 percent specifies that the warning script is run when the file systems are 80 percent full. The settings specify that logins and administrative operations are to be audited. The operations are audited for success and for failure. Failures of all types, except failures to create a file system object, are to be audited. Nonattributable events are also audited. The syslog audit log records fewer audit events. This log contains text summaries of failed logins and failed administrative operations.

flags:lo,am,-all,-^fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/blcken/files
dir:/etc/security/audit/blcken.1/files
#
# Audit filesystem used when blcken fills up
#
dir:/etc/security/audit/wiken
plugin:name=audit_syslog.so.1; p_flags=-lo,-am
audit_event File

The `/etc/security/audit_event` file contains the default audit event-class mappings. You can edit this file to change the class mappings. When you change class mappings, you must reboot the system or run the `auditconfig -conf` command to read the changed mappings into the kernel. For more information, see the `audit_event(4)` man page.

audit_startup Script

The `/etc/security/audit_startup` script automatically configures the auditing service when the system enters multiuser mode. The `auditd` daemon starts after the script performs the following tasks:

- Configures the audit event-class mappings
- Sets the audit policy options

For more information, see the `audit_startup(1M)` man page.

audit_user Database

The `/etc/security/audit_user` database modifies the system-wide preselected classes for an individual user. The classes that you add to a user's entry in the `audit_user` database modify the settings in the `audit_control` file in two ways:

- By specifying audit classes that are always to be audited for this user
- By specifying audit classes that are never to be audited for this user

Each user entry in the `audit_user` database contains three fields:

`username:always-audit-classes:never-audit-classes`

The audit fields are processed in sequence. The `always-audit-classes` field turns on the auditing of the classes in that field. The `never-audit-classes` field turns off the auditing of the classes in that field.

**Note** – Avoid the common mistake of placing the `all` audit class in the `never-audit-classes` field. This mistake causes all auditing to be turned off for that user, which overrides the settings in the `always-audit-classes` field. The setting also overrides system-wide audit class settings in the `audit_control` file.

The `never-audit-classes` settings for a user override the system defaults. You might not want to override system defaults. For example, suppose you want to audit everything for user tamiko, except for successful reads of file system objects. You also want to apply the system defaults to tamiko. Note the placement of the second colon (`:`) in the following `audit_user` entries:

`tamiko:all,^+fr: correct entry`
The correct entry means, “always audit everything, except for successful file-reads.”

tamiko:all:+fr incorrect entry

The incorrect entry means, “always audit everything, but never audit successful file-reads.” The never-audit-classes field, which follows the second colon, overrides the system defaults. In the correct entry, the always-audit-classes field includes the exception to the all audit class. Because no audit class is in the never-audit-classes field, the system defaults from the audit_control file are not overridden.

Note – Successful events and failed events are treated separately. A process could generate more audit records for failed events than for successful events.

audit_warn Script

The /etc/security/audit_warn script notifies an email alias when the auditd daemon encounters an unusual condition while writing audit records. You can customize this script for your site to warn of conditions that might require manual intervention. Or, you could specify how to handle those conditions automatically. For all error conditions, the audit_warn script writes a message to syslog with the severity of daemon.alert. You can use syslog.conf to configure console display of syslog messages. The audit_warn script also sends a message to the audit_warn email alias. You should set up this alias when you enable auditing.

When the auditd daemon detects the following conditions, the daemon invokes the audit_warn script. The script sends email to the audit_warn alias.

- An audit directory has become more full than the minfree value allows. The minfree value or soft limit is a percentage of the available space on an audit file system.

  The audit_warn script is invoked with the string soft and the name of the directory whose available space is below the minimum value. The auditd daemon switches automatically to the next suitable directory. The daemon writes the audit files in this new directory until the directory reaches its minfree limit. The auditd daemon then goes to each remaining directory in the order that is listed in the audit_control file. The daemon writes audit records until each directory is at its minfree limit.

- All the audit directories have reached the minfree threshold.

  The audit_warn script is invoked with the string allsoft. A message is written to the console. Email is also sent to the audit_warn alias.

  When all audit directories that are listed in the audit_control file have reached their minfree threshold, the auditd daemon switches back to the first directory. The daemon writes audit records until the directory becomes completely full.

- An audit directory has become completely full with no space remaining.

  The audit_warn script is invoked with the string hard and the name of the directory. A message is written to the console. Email is also sent to the audit_warn alias.
The `auditd` daemon switches automatically to the next suitable directory with any space available. The `auditd` daemon goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is full.

- All the audit directories are completely full. The `audit_warn` script is invoked with the string `all hard` as an argument.
  
  By default, a message is written to the console. Email is also sent to the `audit_warn` alias. Processes that would otherwise generate audit records continue to occur, but audit records are counted. Audit records are not generated. For an example of how to handle this situation, see Example 29–14 and “How to Prevent Audit Trail Overflow” on page 551.

- An internal error occurs. Possible internal errors include the following:
  - `ebusy` – Another `auditd` daemon process is already running
  - `tmpfile` – A temporary file cannot be used
  - `postsigterm` – A signal was received during auditing shutdown

- A problem is discovered with the syntax of the `audit_control` file. By default, a message is sent to the console. Email is also sent to the `audit_warn` alias.

For further information, see the `audit_warn(1M)` man page.

**bsmconv Script**

The `/etc/security/bsmconv` script enables the auditing service. The `bsmunconv` command disables the auditing service. After the `bsmconv` script is run, you configure the audit directories and audit configuration files. Upon reboot, auditing is enabled.

For further information, see the `bsmconv(1M)` man page.

**Rights Profiles for Administering Auditing**

The Solaris OS provides rights profiles for configuring the auditing service and for analyzing the audit trail.

- **Audit Control** – Enables a role to configure Solaris auditing. This rights profile grants authorizations to configure files that are used by the auditing service. The profile also enables a role to run audit commands. A role with the Audit Control profile can run the following commands: `audit`, `auditd`, `auditconfig`, `bsmconv`, and `bsmunconv`.

- **Audit Review** – Enables a role to analyze Solaris audit records. This rights profile grants authorization to read audit records with the `praudit` and `auditreduce` commands. A role with this rights profile can also run the `auditstat` command.

- **System Administrator** – Includes the Audit Review rights profile. A role with the System Administrator rights profile can analyze audit records.

To configure roles to handle the auditing service, see “Configuring RBAC (Task Map)” on page 188.
Auditing and Solaris Zones

A zone is a virtualized operating system environment that is created within a single instance of the Solaris Operating System. Audit policy can be set in the global zone for all zones to be audited identically.

When all zones are being audited identically, only configuration files in the global zone are customized for the auditing service. The +zonename policy option is useful. When this option is set, the audit records from all zones include the name of the zone. Audit records can then be postselected by zone name. To understand audit policy, see “Determining Audit Policy” on page 517. For an example, see “How to Configure Audit Policy” on page 536.

Zones can also be audited individually. When the policy option, per zone, is set in the global zone, each non-global zone runs its own audit daemon, handles its own audit queue, and specifies the content and location of its audit records. A non-global zone can also set most audit policy options. It cannot set policy that affects the entire system, so a non-global zone cannot set the ahlt or per zone policy. For further discussion, see “How to Plan Auditing in Zones” on page 514.

Note – If name service files are customized in non-global zones, and per zone policy is not set, then careful use of the audit tools is required to select usable records. A user ID in one zone can refer to a different user from the same ID in a different zone.

To generate usable records, set the zonename audit policy in the global zone. In the global zone, run the audit reduce command with the zonename option. Then, in the zonename zone, run the praudit command on the audit reduce output.


Audit Classes

System-wide defaults for Solaris auditing are preselected by specifying one or more classes of events. The classes are preselected for each system in the system’s audit_control file. Anyone who uses the system is audited for these classes of events. The file is described in “audit_control File” on page 559.

You can configure audit classes and make new audit classes. Audit class names can be up to 8 characters in length. The class description is limited to 72 characters. Numeric and non-alphanumeric characters are allowed.

You can modify what is audited for individual users by adding audit classes to a user’s entry in the audit_user database. The audit classes are also used as arguments to the audit_config command. For details, see the audit_config(1M) man page.
Definitions of Audit Classes

The following table shows each predefined audit class, the descriptive name for each audit class, and a short description.

**TABLE 30-1 Predefined Audit Classes**

<table>
<thead>
<tr>
<th>Audit Class</th>
<th>Descriptive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all</td>
<td>All classes (meta-class)</td>
</tr>
<tr>
<td>no</td>
<td>no_class</td>
<td>Null value for turning off event preselection</td>
</tr>
<tr>
<td>na</td>
<td>non_attrib</td>
<td>Nonattributable events</td>
</tr>
<tr>
<td>fr</td>
<td>file_read</td>
<td>Read of data, open for reading</td>
</tr>
<tr>
<td>fw</td>
<td>file_write</td>
<td>Write of data, open for writing</td>
</tr>
<tr>
<td>fa</td>
<td>file_attr_acc</td>
<td>Access of object attributes: stat, pathconf</td>
</tr>
<tr>
<td>fm</td>
<td>file_attr_mod</td>
<td>Change of object attributes: chown, flock</td>
</tr>
<tr>
<td>fc</td>
<td>file_creation</td>
<td>Creation of object</td>
</tr>
<tr>
<td>fd</td>
<td>file_deletion</td>
<td>Deletion of object</td>
</tr>
<tr>
<td>cl</td>
<td>file_close</td>
<td>close system call</td>
</tr>
<tr>
<td>ap</td>
<td>application</td>
<td>Application-defined event</td>
</tr>
<tr>
<td>ad</td>
<td>administrative</td>
<td>Administrative actions (old administrative meta-class)</td>
</tr>
<tr>
<td>am</td>
<td>administrative</td>
<td>Administrative actions (meta-class)</td>
</tr>
<tr>
<td>ss</td>
<td>system state</td>
<td>Change system state</td>
</tr>
<tr>
<td>as</td>
<td>system-wide</td>
<td>System-wide administration</td>
</tr>
<tr>
<td>ua</td>
<td>user administration</td>
<td>User administration</td>
</tr>
<tr>
<td>aa</td>
<td>audit administration</td>
<td>Audit utilization</td>
</tr>
<tr>
<td>ps</td>
<td>process start</td>
<td>Process start and process stop</td>
</tr>
<tr>
<td>pm</td>
<td>process modify</td>
<td>Process modify</td>
</tr>
<tr>
<td>pc</td>
<td>process</td>
<td>Process (meta-class)</td>
</tr>
<tr>
<td>ex</td>
<td>exec</td>
<td>Program execution</td>
</tr>
<tr>
<td>io</td>
<td>ioctl</td>
<td>ioctl() system call</td>
</tr>
<tr>
<td>ip</td>
<td>ipc</td>
<td>System V IPC operations</td>
</tr>
</tbody>
</table>
### TABLE 30–1 Predefined Audit Classes (Continued)

<table>
<thead>
<tr>
<th>Audit Class</th>
<th>Descriptive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>login_logout</td>
<td>Login and logout events</td>
</tr>
<tr>
<td>nt</td>
<td>network</td>
<td>Network events: bind, connect, accept</td>
</tr>
<tr>
<td>ot</td>
<td>other</td>
<td>Miscellaneous, such as device allocation and memcntl()</td>
</tr>
</tbody>
</table>

You can define new classes by modifying the `/etc/security/audit_class` file. You can also rename existing classes. For more information, see the `audit_class(4)` manpage.

### Audit Class Syntax

Events can be audited for success, events can be audited for failure, and events can be audited for both. Without a prefix, a class of events is audited for success and for failure. With a plus (+) prefix, a class of events is audited for success only. With a minus (-) prefix, a class of events is audited for failure only. The following table shows some possible representations of audit classes.

### TABLE 30–2 Plus and Minus Prefixes to Audit Classes

<table>
<thead>
<tr>
<th>[prefix]class</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>Audit all successful attempts to log in and log out, and all failed attempts to log in. A user cannot fail an attempt to log out.</td>
</tr>
<tr>
<td>+lo</td>
<td>Audit all successful attempts to log in and log out.</td>
</tr>
<tr>
<td>-all</td>
<td>Audit all failed events.</td>
</tr>
<tr>
<td>+all</td>
<td>Audit all successful events.</td>
</tr>
</tbody>
</table>

**Caution** – The `all` class can generate large amounts of data and quickly fill audit file systems. Use the `all` class only if you have extraordinary reasons to audit all activities.

Audit classes that were previously selected can be further modified by a caret prefix, `. The following table shows how the caret prefix modifies a preselected audit class.

### TABLE 30–3 Caret Prefix That Modifies Already-Specified Audit Classes

<table>
<thead>
<tr>
<th><code>^prefixclass</code></th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all, <code>^·fc</code></td>
<td>Audit all failed events, except do not audit failed attempts to create file objects</td>
</tr>
</tbody>
</table>
### Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask** – A combination of the audit classes from the audit_control file and the audit_user database. When a user logs in, the login process combines the preselected classes to establish the process preselection mask for the user’s processes. The process preselection mask specifies whether events in each audit class are to generate audit records.

The following algorithm describes how the system obtains the user’s process preselection mask:

\[(\text{flags line} + \text{always-audit-classes}) - \text{never-audit-classes}\]

Add the audit classes from the flags line in the audit_control file to the classes from the always-audit-classes field in the user’s entry in the audit_user database. Then, subtract from the total the classes from the user’s never-audit-classes field.

- **Audit ID** – A process acquires an audit ID when the user logs in. The audit ID is inherited by all child processes that were started by the user’s initial process. The audit ID helps enforce accountability. Even after a user becomes root, the audit ID remains the same. The audit ID that is saved in each audit record always allows you to trace actions back to the original user who had logged in.

- **Audit Session ID** – The audit session ID is assigned at login. The session ID is inherited by all child processes.
Terminal ID (port ID, machine ID) – The terminal ID consists of the host name and the Internet address, followed by a unique number that identifies the physical device on which the user logged in. Most often, the login is through the console. The number that corresponds to the console device is 0.

Audit Trail

The audit trail is contains binary audit files. The trail is created by the auditd daemon. Once the auditing service has been enabled with the bsmconv command, the auditd daemon starts when the system is booted. The auditd daemon is responsible for collecting the audit trail data and writing the audit records.

The audit records are stored in binary format on file systems that are dedicated to audit files. Even though you can physically locate audit directories within file systems that are not dedicated to auditing, do not do so except for directories of last resort. Directories of last resort are directories where audit files are written only when no other suitable directory is available.

There is one other scenario where locating audit directories outside of dedicated audit file systems could be acceptable. You might do so in a software development environment where auditing is optional. To make full use of disk space might be more important than to keep an audit trail. However, in a security-conscious environment, the placement of audit directories within other file systems is not acceptable.

You should also consider the following factors when administering audit file systems:

- A host should have at least one local audit directory. The local directory can be used as a directory of last resort if the host is unable to communicate with the audit server.
- Mount audit directories with the read-write (rw) option. When you mount audit directories remotely, also use the int r and noac options.
- List the audit file systems on the audit server where they reside. The export list should include all systems that are being audited at the site.

Conventions for Binary Audit File Names

Each binary 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 system that generated them.

Binary Audit File Names

Audit files that are complete have names of the following form:

```
start-time.end-time.system
```

`start-time` is the time that the first audit record in the audit file was generated.
end-time Is the time that the last record was written to the file
system Is the name of the system that generated the file

An audit file that is still active has a name of the following form:
start-time.not_terminated.system

For examples of not_terminated and closed audit file names, see "How to Clean Up a not_terminated Audit File" on page 550.

**Binary Audit File Timestamps**

The timestamps in file names are used by the audit reduce command to locate records within a specific time range. These timestamps are important because there can be a month’s accumulation or more of audit files online. To search all the files for records that were generated in the last 24 hours would be unacceptably expensive.

The start-time and end-time are timestamps with one-second resolution. They are specified in Greenwich Mean Time (GMT). The format is four digits for the year, followed by two digits for each month, day, hour, minute, and second, as follows:

YYYYMMDDHHMMSS

The timestamps are in GMT to ensure that they sort in proper order, even across time zones. Because they are in GMT, the date and hour must be translated to the current time zone to be meaningful. Beware of this point whenever you manipulate these files with standard file commands rather than with the audit reduce command.

**Audit Record Structure**

An audit record is a sequence of audit tokens. Each audit token contains event information such as user ID, time, and date. A header token begins an audit record, and an optional trailer token concludes the record. Other audit tokens contain information relevant to the audit event. The following figure shows a typical audit record.
Audit Record Analysis

Audit record analysis involves postselecting records from the audit trail. You can use one of two approaches to parsing the binary data that was collected.

- You can parse the binary data stream. To parse the data stream, you need to know the order of the fields in each token, and the order of tokens in each record. You also need to know the variants of an audit record. For example, the `ioctl()` system call creates an audit record for "Bad file name" that contains different tokens from the audit record for "Invalid file descriptor".

  - For a description of the order of binary data in each audit token, see the `audit(4)` man page.
  - For a description of the order of tokens in an audit record, use the `bsmrecord` command. Output from the `bsmrecord` command includes the different formats that occur under different conditions. Square brackets ([ ]) indicate that an audit token is optional. For more information, see the `bsmrecord(1M)` man page. For examples, see also "How to Display Audit Record Formats" on page 543.

- You can use the `praudit` command. Options to the command provide different text outputs. For example, the `praudit -x` command provides XML for input into scripts and browsers. `praudit` outputs do not include fields whose sole purpose is to help to parse the binary data. The outputs do not necessarily follow the order of the binary fields. Also, the order and format of `praudit` output is not guaranteed between Solaris releases.

  For examples of `praudit` output, see "How to View the Contents of Binary Audit Files" on page 548, and the `praudit(1M)` man page.

  For a description of the `praudit` output for each audit token, see the individual tokens in the "Audit Token Formats" on page 571 section.
### Audit Token Formats

Each audit token has a token type identifier, which is followed by data that is specific to the token. Each token type has its own format. The following table shows the token names with a brief description of each token. Obsolete tokens are maintained for compatibility with previous Solaris releases.

<table>
<thead>
<tr>
<th>Token Name</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl</td>
<td>Access Control List (ACL) information</td>
<td>“acl Token” on page 572</td>
</tr>
<tr>
<td>arbitrary</td>
<td>Data with format and type information</td>
<td>“arbitrary Token (Obsolete)” on page 572</td>
</tr>
<tr>
<td>arg</td>
<td>System call argument value</td>
<td>“arg Token” on page 573</td>
</tr>
<tr>
<td>attribute</td>
<td>File vnode tokens</td>
<td>“attribute Token” on page 574</td>
</tr>
<tr>
<td>cmd</td>
<td>Command arguments and environment variables</td>
<td>“cmd Token” on page 574</td>
</tr>
<tr>
<td>exec_args</td>
<td>Exec system call arguments</td>
<td>“exec_args Token” on page 575</td>
</tr>
<tr>
<td>exec_env</td>
<td>Exec system call environment variables</td>
<td>“exec_env Token” on page 575</td>
</tr>
<tr>
<td>exit</td>
<td>Program exit information</td>
<td>“exit Token (Obsolete)” on page 575</td>
</tr>
<tr>
<td>file</td>
<td>Audit file information</td>
<td>“file Token” on page 576</td>
</tr>
<tr>
<td>group</td>
<td>Process groups information</td>
<td>“group Token (Obsolete)” on page 576</td>
</tr>
<tr>
<td>groups</td>
<td>Process groups information</td>
<td>“groups Token” on page 576</td>
</tr>
<tr>
<td>header</td>
<td>Indicates start of audit record</td>
<td>“header Token” on page 577</td>
</tr>
<tr>
<td>in_addr</td>
<td>Internet address</td>
<td>“in_addr Token” on page 577</td>
</tr>
<tr>
<td>ip</td>
<td>IP header information</td>
<td>“ip Token (Obsolete)” on page 578</td>
</tr>
<tr>
<td>ipc</td>
<td>System V IPC information</td>
<td>“ipc Token” on page 578</td>
</tr>
<tr>
<td>ipc_perm</td>
<td>System V IPC object tokens</td>
<td>“ipc_perm Token” on page 579</td>
</tr>
<tr>
<td>iport</td>
<td>Internet port address</td>
<td>“iport Token” on page 579</td>
</tr>
<tr>
<td>opaque</td>
<td>Unstructured data (unspecified format)</td>
<td>“opaque Token (Obsolete)” on page 579</td>
</tr>
<tr>
<td>path</td>
<td>Path information</td>
<td>“path Token” on page 580</td>
</tr>
<tr>
<td>path_attr</td>
<td>Access path information</td>
<td>“path_attr Token” on page 580</td>
</tr>
<tr>
<td>privilege</td>
<td>Privilege set information</td>
<td>“privilege Token” on page 581</td>
</tr>
<tr>
<td>process</td>
<td>Process token information</td>
<td>“process Token” on page 581</td>
</tr>
</tbody>
</table>
An audit record always begins with a header token. The header token indicates where the audit record begins in the audit trail. In the case of attributable events, the subject and the process tokens refer to the values of the process that caused the event. In the case of nonattributable events, the process token refers to the system.

**acl Token**

The acl token records information about Access Control Lists (ACLs). This token consists of four fixed fields:

- A token ID that identifies this token as an acl token
- A field that specifies the ACL type
- An ACL value field
- A field that lists the permissions associated with this ACL

The praudit command displays the acl token as follows:

```
acl,jdoe,staff,0755
```
The remainder of the token is composed of count of the specified type. The praudit command displays the arbitrary token as follows:

```
arbitrary,decimal,int,1
42
```

The following table shows the possible values of the print format field.

**TABLE 30–5 Values for the arbitrary Token’s Print Format Field**

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUP_BINARY</td>
<td>Prints the date in binary format</td>
</tr>
<tr>
<td>AUP_OCTAL</td>
<td>Prints the date in octal format</td>
</tr>
<tr>
<td>AUP_DECIMAL</td>
<td>Prints the date in decimal format</td>
</tr>
<tr>
<td>AUP_HEX</td>
<td>Prints the date in hexadecimal format</td>
</tr>
<tr>
<td>AUP_STRING</td>
<td>Prints the date as a string</td>
</tr>
</tbody>
</table>

The following table shows the possible values of the item size field.

**TABLE 30–6 Values for the arbitrary Token’s Item Size Field**

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUR_BYTE</td>
<td>Data is printed in units of bytes in 1 byte</td>
</tr>
<tr>
<td>AUR_SHORT</td>
<td>Data is printed in units of shorts in 2 bytes</td>
</tr>
<tr>
<td>AUR_LONG</td>
<td>Data is printed in units of longs in 4 bytes</td>
</tr>
</tbody>
</table>

**arg Token**

The arg token contains information about the arguments to a system call: the argument number of the system call, the argument value, and an optional description. This token allows a 32-bit integer system-call argument in an audit record. The arg token has five fields:

- A token ID that identifies this token as an arg token
- An argument ID that tells which system call argument that the token refers to
- The argument value
- The length of the descriptive text string
- The text string

The praudit command displays the arg token without the fourth field, as follows:

```
argument,4,0xffbfe0ac,pri
```

The praudit -x command includes the names of the fields that are displayed:
attribute Token

The attribute token contains information from the file vnode. This token has seven fields:

- A token ID that identifies this token as an attribute token
- The file access mode and type
- The owner user ID
- The owner group ID
- The file system ID
- The node ID
- The device ID that the file might represent

For further information about the file system ID and the device ID, see the `statvfs(2)` man page.

The attribute token usually accompanies a path token. The attribute token is produced during path searches. If a path-search error occurs, there is no vnode available to obtain the necessary file information. Therefore, the attribute token is not included as part of the audit record. The `praudit` command displays the attribute token as follows:

```
attribute,20666,root,root,247,4829,450971566127
```

cmd Token

The cmd token records the list of arguments and the list of environment variables that are associated with a command.

The cmd token contains the following fields:

- A token ID that identifies this token as a cmd token
- A count of the command’s arguments
- The argument list
- The length of the next field
- The content of the arguments
- A count of the environment variables
- The list of environment variables
- The length of the next field
- The content of the environment variables

The `praudit` command displays the cmd token as follows:

```
cmd,argcnt,3,ls,-l,/etc,envcnt,0,
```
exec_args Token

The exec_args token records the arguments to an exec() system call. The exec_args token has two fixed fields:

- A token ID field that identifies this token as an exec_args token
- A count that represents the number of arguments that are passed to the exec() system call

The remainder of this token is composed of count strings. The praudit command displays the exec_args token as follows:

exec_args,2,vi,/etc/security/audit_user

Note - The exec_args token is output only when the argv audit policy option is active.

exec_env Token

The exec_env token records the current environment variables to an exec() system call. The exec_env token has two fixed fields:

- A token ID field that identifies this token as an exec_env token
- A count that represents the number of arguments that are passed to the exec() system call

The remainder of this token is composed of count strings. The praudit command displays the exec_env token as follows:

exec_env,25,
GROUP=staff,HOME=/export/home/jdoe,HOST=exm1,HOSTTYPE=sun4u,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=jdoe,MACHTYPE=sparc,
MAIL=/var/mail/jdoe,OSTYPE=solaris,PATH=/usr/sbin:/usr/bin,PS1=,
PWD=/var/audit,REMOTEHOST=192.168.13.5,SHLL=/usr/bin/csh,SHLVL=1,
TERM=dtterm,TZ=US/Pacific,USER=jdoe,VENDOR=sun

Note - The exec_env token is output only when the argv audit policy option is active.

exit Token (Obsolete)

The exit token records the exit status of a program. The exit token contains the following fields:

- A token ID that identifies this token as an exit token
- A program exit status as passed to the exit() system call
A return value that describes the exit status or that provides a system error number.

The `praudit` command displays the `exit` token as follows:

```
exit, Error 0, 0
```

### file Token

The `file` token is a special token that is generated by the `auditd` daemon. The token marks the beginning of a new audit file and the end of an old audit file as the old file is deactivated. The `auditd` daemon builds a special audit record that contains 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 timestamp that identifies the date and the time that the file was created or was closed
- The file name length
- A field that holds the file null-terminated name

The `praudit -x` command shows the fields of the `file` token:

```
file, 2003-10-13 11:21:35.506 -07:00, 
/var/audit/localhost/files/20031013175058.20031013182135.example1
```

### group Token (Obsolete)

This token has been replaced by the `groups` token. See “`groups` Token” on page 576.

### groups Token

The `groups` token replaces the `group` token. The `groups` token records the group entries from the process’s credential. The `groups` token has two fixed fields:

- A token ID field that identifies this token as a `groups` token
- A count that represents the number of groups that are contained in this audit record

The remainder of this token is composed of `count` group entries. The `praudit` command displays the `groups` token as follows:

```
groups, staff, admin
```

**Note** – The `groups` token is output only when the `group` audit policy option is active.
header Token

The header token is special in that it marks the beginning of an audit record. The header token combines with the trailer token to bracket all the other tokens in the record. The header token has eight fields:

- A token ID field that identifies this token as a header token
- A byte count of the total length of the audit record, including both the header and the trailer tokens
- A version number that identifies the version of the audit record structure
- The audit event ID that identifies the audit event that the record represents
- The ID modifier that identifies special characteristics of the audit event
- The address type, either IPv4 or IPv6
- The machine's IP address
- The time and date that the record was created

On 64-bit systems, the header token is displayed with a 64-bit timestamp, in place of the 32-bit timestamp.

The praudit command displays the header token for a ioctl() system call as follows:

header,176,2,ioctl(2),fe,example1,2003-09-08 11:23:31.050 -07:00

The ID modifier field has the following flags defined:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4000</td>
<td>PAD_NOTATTR  nonattributable event</td>
</tr>
<tr>
<td>0x8000</td>
<td>PAD_FAILURE  failed audit event</td>
</tr>
</tbody>
</table>

in_addr Token

The in_addr token contains an Internet Protocol address. Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address. The in_addr token has three fields:

- A token ID that identifies this token as an in_addr token
- The IP address type, either IPv4 or IPv6
- An IP address

The praudit command displays the in_addr token, without the second field, as follows:

ip address,192.168.113.7
**ip Token (Obsolete)**

The ip token contains a copy of an Internet Protocol header. The ip token has two fields:

- A token ID that identifies this token as an ip token
- A copy of the IP header, that is, 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 the `/usr/include/netinet/ip.h` file.

**ipc Token**

The ipc token contains the System V IPC message handle, semaphore handle, or shared-memory handle that is used by the caller to identify a particular IPC object. The ipc token has three fields:

- A token ID that identifies this token as an ipc token
- A type field that specifies the type of IPC object
- The handle that identifies the IPC object

**Note** – The IPC object identifiers violate the context-free nature of the Solaris audit tokens. No global “name” uniquely identifies IPC objects. Instead, IPC objects are identified by their handles. The handles are valid only during the time that the IPC objects are active. However, the identification of IPC objects should not be a problem. The System V IPC mechanisms are seldom used, and the mechanisms all share the same audit class.

The following table shows the possible values for the IPC object type field. The values are defined in the `/usr/include/bsm/audit.h` file.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU_IPC_MSG</td>
<td>1</td>
<td>IPC message object</td>
</tr>
<tr>
<td>AU_IPC_SEM</td>
<td>2</td>
<td>IPC semaphore object</td>
</tr>
<tr>
<td>AU_IPC_SHM</td>
<td>3</td>
<td>IPC shared-memory object</td>
</tr>
</tbody>
</table>

The praudit command displays the ipc token as follows:

```
IPC,msg,3
```
**ipc_perm Token**

The `ipc_perm` token contains a copy of the System V IPC access permissions. This token is added to audit records that are generated by IPC shared-memory events, IPC semaphore events, and IPC message events. The `ipc_perm` 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 mode of the IPC
- The sequence number of the IPC
- The IPC key value

The `praudit` command displays the `ipc_perm` token as follows:

```
IPC perm,root,sys,root,sys,0,0,0x00000000
```

The values are taken from the `ipc_perm` structure that is associated with the IPC object.

**iport Token**

The `iport` token contains the TCP or UDP port address. The `iport` token has two fields:

- A token ID that identifies this token as an `iport` token
- The TCP or UDP port address

The `praudit` command displays the `iport` token as follows:

```
iport,0xf6d6
```

**opaque Token (Obsolete)**

The opaque token contains unformatted data as a sequence of bytes. The opaque token has three fields:

- A token ID that identifies this token as an `opaque` token
- A byte count of the data
- An array of byte data

The `praudit` command displays the opaque token as follows:

```
opaque,12,0x4f5041515545204441544100
```
The path token contains access path information for an object. This token contains the following fields:

- A token ID that identifies this token as a path token
- The path length
- The absolute path to the object that is based on the real root of the system

The `praudit` command displays the path token, without the second field, as follows:

```
path,/etc/security/audit_user
```

The `praudit -x` command displays the path token as follows:

```
<path>/etc/security/audit_user</path>
```

The following figure shows the format of a path token.

![Path Token Format](image)

The path_attr token contains access path information for an object. The access path specifies the sequence of attribute file objects below the path token object. Systems calls such as `openat()` access attribute files. For more information on attribute file objects, see the `fsattr(5)` man page.

The path_attr token contains the following fields:

- A token ID that identifies this token as a path_attr token
- A count that represents the number of sections of attribute file paths
- `count` null-terminated strings

The `praudit` command displays the path_attr token as follows:

```
path_attr,1,attr_file_name
```
**privilege Token**

The privilege token records the use of privileges on a process. The privilege token is not recorded for privileges in the basic set. If a privilege has been removed from the basic set by administrative action, then the use of that privilege is recorded. For more information on privileges, see “Privileges (Overview)” on page 178

The privilege token contains the following fields:

- A token ID that identifies this token as a privilege token
- The length of the following field
- The name of privilege set
- The length of the following field
- The list of privileges

The praudit command displays the privilege token as follows:

```
privilege,effective,
```

**process Token**

The process token contains information about a user who is associated with a process, such as the recipient of a signal. The process token has nine fields:

- A token ID that identifies this token as a process token
- The 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
- A terminal ID that consists of a device ID and a machine ID

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 value is then set to -1.

Any token that contains a terminal ID has several variations. The praudit command hides these variations. So, the terminal ID is handled the same way for any token that contains a terminal ID. The terminal ID is either an IP address and port number, or a device ID. A device ID, such as the serial port that is connected to a modem, can be zero. The terminal ID is specified in one of several formats.

The terminal ID for device numbers is specified as follows:
### Audit Token Formats

- **32-bit applications** – 4-byte device number, 4 bytes unused
- **64-bit applications** – 8-byte device number, 4 bytes unused

In releases prior to the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- **32-bit applications** – 4-byte port number, 4-byte IP address
- **64-bit applications** – 8-byte port number, 4-byte IP address

Since the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- **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

The `praudit` command displays the process token as follows:

```
process, root, root, sys, root, sys, 0, 0, 0, 0.0.0.0
```

The following figure shows the format of a process token.

![Process Token Format](image)

#### return Token

The `return` token contains the return status of the system call (`u_error`) and the process return value (`u_rval1`). This token has three fields:

- A token ID that identifies this token as a return token
- The error status of the system call
- The return value of the system call

The `return` token is always returned as part of kernel-generated audit records for system calls. In application auditing, this token indicates exit status and other return values.

The `praudit` command displays the `return` token for a system call as follows:

```
return, failure: Operation now in progress, -1
```

The `praudit -x` command displays the `return` token as follows:
sequence Token

The sequence token contains a sequence number. This token is useful for debugging. The sequence token has two fields:

- A token ID that identifies this token as a sequence token
- A 32-bit unsigned long field that contains the sequence number

The sequence number is incremented every time an audit record is added to the audit trail. The `praudit` command displays the sequence token as follows:

```
sequence, 1292
```

The `praudit -x` command displays the sequence token as follows:

```
<sequence seq-num="1292"/>
```

Note – The sequence token is output only when the seq audit policy option is active.

socket Token

The socket token contains information that describes an Internet socket. In some instances, the token has four fields:

- A token ID that identifies this token as a socket token
- A socket type field that indicates the type of socket referenced, either TCP, UDP, or UNIX
- The local port
- The local IP address

The `praudit` command displays this instance of the socket token as follows:

```
socket, 0x0002, 0x83b1, localhost
```

In most instances, the token has eight fields:

- A token ID that identifies this token as a socket token
- The socket domain
- A socket type field that indicates the type of socket referenced, either TCP, UDP, or UNIX
- The local port
- The address type, either IPv4 or IPv6
- The local IP address
The remote port
The remote IP address

Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address.

The `praudit` command displays the socket token as follows:

```
socket, 0x0002, 0x0002, 0x83cf, example1, 0x2383, server1.Subdomain.Domain.COM
```

The `praudit -x` command describes the socket token fields. The lines are wrapped for display purposes.

```
<socket sock_domain="0x0002" sock_type="0x0002" lport="0x83cf"
laddr="example1" fport="0x2383" faddr="server1.Subdomain.Domain.COM"/>
```

subject Token

The subject token describes a user who performs or attempts to perform an operation. The format is the same as the process token. The subject token has nine fields:

- A token ID that identifies this token as a subject token
- The 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
- A terminal ID that consists of a device ID and a machine ID

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 value is then set to -1.

Any token that contains a terminal ID has several variations. The `praudit` command hides these variations. So, the terminal ID is handled the same way for any token that contains a terminal ID. The terminal ID is either an IP address and port number, or a device ID. A device ID, such as the serial port that is connected to a modem, can be zero. The terminal ID is specified in one of several formats.

The terminal ID for device numbers is specified as follows:

- **32-bit applications** – 4-byte device number, 4 bytes unused
64-bit applications – 8-byte device number, 4 bytes unused

In releases prior to the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- 32-bit applications – 4-byte port number, 4-byte IP address
- 64-bit applications – 8-byte port number, 4-byte IP address

Since the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- 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

The subject token is always returned as part of kernel-generated audit records for system calls. The praudit command displays the subject token as follows:

```
subject,jdoe,root,staff,root,staff,424,223,0 0 example1
```

The following figure shows the format of the subject token.

![Figure 30-6: Subject Token Format](image)

**text Token**

The text token contains a text string. This token has three fields:

- A token ID that identifies this token as a text token
- The length of the text string
- The text string itself

The praudit command displays the text token as follows:

```
text,logout jdoe
```
**trailer Token**

The two tokens, header and trailer, are special in that they distinguish the end points of an audit record and bracket all the other tokens. A header token begins an audit record. A trailer token ends an audit record. The trailer token is an optional token. The trailer token is added as the last token of each record only when the trail audit policy option has been set.

When an audit record is generated with trailers turned on, the auditreduce command can verify that the trailer correctly points back to the record header. 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
- The total number of characters in the audit record, including both the header and trailer tokens

The praudit command displays the trailer token, without the second field, as follows:
```
trailer, 136
```

**uauth Token**

The uauth token records the use of authorization with a command or action.

The uauth token contains the following fields:
- A token ID that identifies this token as a uauth token
- The length of the text in the following field
- A list of authorizations

The praudit command displays the uauth token as follows:
```
use of authorization, solaris.admin.printer.delete
```

**zonename Token**

The zonename token records the zone in which the audit event occurred. The string “global” indicates audit events that occur in the global zone.

The zonename token contains the following fields:
- A token ID that identifies this token as a zonename token
- The length of the text in the following field
- The name of the zone

The praudit command displays the zonename token as follows:
zonename, graphzone
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Control List (ACL)</strong></td>
<td>An access control list (ACL) provides finer-grained file security than traditional UNIX file protection provides. For example, an ACL enables you to allow group read access to a file, while allowing only one member of that group to write to the file.</td>
</tr>
<tr>
<td><strong>admin principal</strong></td>
<td>A user principal with a name of the form <code>username/admin</code> (as in <code>jdoe/admin</code>). An admin principal can have more privileges (for example, to change policies) than a regular user principal. See also principal name, user principal.</td>
</tr>
<tr>
<td><strong>algorithm</strong></td>
<td>A cryptographic algorithm. This is an established, recursive computational procedure that encrypts or hashes input.</td>
</tr>
<tr>
<td><strong>application server</strong></td>
<td>See network application server.</td>
</tr>
<tr>
<td><strong>audit files</strong></td>
<td>Binary audit logs. Audit files are stored separately in an audit partition.</td>
</tr>
<tr>
<td><strong>audit partition</strong></td>
<td>A hard disk partition that is configured to hold audit files.</td>
</tr>
<tr>
<td><strong>audit policy</strong></td>
<td>The global and per-user settings that determine which audit events are recorded. The global settings that apply to the audit service typically affect which pieces of optional information are included in the audit trail. Two settings, <code>cnt</code> and <code>ahlt</code>, affect the operation of the system when the audit queue fills. For example, audit policy might require that a sequence number be part of every audit record.</td>
</tr>
<tr>
<td><strong>audit trail</strong></td>
<td>The collection of all audit files from all hosts.</td>
</tr>
<tr>
<td><strong>authentication</strong></td>
<td>The process of verifying the claimed identity of a principal.</td>
</tr>
<tr>
<td><strong>authenticator</strong></td>
<td>Authenticators are passed by clients when requesting tickets (from a KDC) and services (from a server). They contain information that is generated by using a session key known only by the client and server, that can be verified as of recent origin, thus indicating that 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 time</td>
</tr>
</tbody>
</table>
stamp. Unlike a ticket, an authenticator can be used only once, usually when access to a service is requested. An authenticator is encrypted by using the session key for that client and that server.

**authorization**

1. In Kerberos, the process of determining if a principal can use a service, which objects the principal is allowed to access, and the type of access that is allowed for each object.

2. In role-based access control (RBAC), 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.

**Basic Security Module (BSM)**

The Solaris auditing service and device allocation. Together, these features satisfy the C2 level of security.

**basic set**

The set of privileges that are assigned to a user’s process at login. On an unmodified system, each user’s initial inheritable set equals the basic set at login.

**Blowfish**

A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.

**client**

Narrowly, a process that makes use of a network service on behalf of a user; for example, an application that uses `rlogin`. 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 that is 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 `rpc_gss_principal_t` structures.

**clock skew**

The maximum amount of time that the internal system clocks on all hosts that are participating in the Kerberos authentication system can differ. If the clock skew is exceeded between any of the participating hosts, requests are rejected. Clock skew can be specified in the `krb5.conf` file.

**confidentiality**

See `privacy`.

**consumer**

In the Solaris Cryptographic Framework, a consumer is a user of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations. Kerberos, IKE, and IPsec are examples of consumers. For examples of providers, see `provider`.

**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`.
<table>
<thead>
<tr>
<th><strong>credential cache</strong></th>
<th>A storage space (usually a file) that contains credentials that are received from the KDC.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cryptographic algorithm</strong></td>
<td>See algorithm.</td>
</tr>
<tr>
<td><strong>DES</strong></td>
<td>Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.</td>
</tr>
<tr>
<td><strong>device allocation</strong></td>
<td>Device protection at the user level. Device allocation enforces the exclusive use of a device by one user at a time. Device data is purged before device reuse. Authorizations can be used to limit who is permitted to allocate a device.</td>
</tr>
<tr>
<td><strong>device policy</strong></td>
<td>Device protection at the kernel level. Device policy is implemented as two sets of privileges on a device. One set of privileges controls read access to the device. The second set of privileges controls write access to the device. See also policy.</td>
</tr>
<tr>
<td><strong>Diffie-Hellman protocol</strong></td>
<td>Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by Kerberos.</td>
</tr>
<tr>
<td><strong>digest</strong></td>
<td>See message digest.</td>
</tr>
<tr>
<td><strong>DSA</strong></td>
<td>Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA1 for input.</td>
</tr>
<tr>
<td><strong>effective set</strong></td>
<td>The set of privileges that are currently in effect on a process.</td>
</tr>
<tr>
<td><strong>flavor</strong></td>
<td>Historically, security flavor and authentication flavor had the same meaning, as a flavor that 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.</td>
</tr>
<tr>
<td><strong>forwardable ticket</strong></td>
<td>A ticket that a client can use to request a ticket on a remote host without requiring the client to go through the full authentication process on that host. For example, if the user david obtains a forwardable ticket while on user jennifer’s machine, he can log in to his own machine without being required to get a new ticket (and thus authenticate himself again). See also proxiable ticket.</td>
</tr>
<tr>
<td><strong>FQDN</strong></td>
<td>Fully qualified domain name. For example, central.example.com (as opposed to simply denver).</td>
</tr>
<tr>
<td><strong>GSS-API</strong></td>
<td>The Generic Security Service Application Programming Interface. A network layer that provides support for various modular security services, including the Kerberos service. GSS-API provides for security authentication, integrity, and privacy services. See also authentication, integrity, privacy.</td>
</tr>
<tr>
<td><strong>hardening</strong></td>
<td>The modification of the default configuration of the operating system to remove security vulnerabilities that are inherent in the host.</td>
</tr>
</tbody>
</table>
In the Solaris Cryptographic Framework, a device driver and its hardware accelerator. Hardware providers offload expensive cryptographic operations from the computer system, thus freeing CPU resources for other uses. See also provider.

A machine that is accessible over a network.

A particular instance of a service principal in which the principal (signified by the primary name host) is set up to provide a range of network services, such as ftp, rcp, or rlogin. An example of a host principal is host/central.example.com@EXAMPLE.COM. See also server principal.

The set of privileges that a process can inherit across a call to exec.

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. This assurance is important because an initial ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might existed for a long time).

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. The instance the host’s fully qualified domain name, as in host/central.example.com. For user principals, an instance is optional. Note, however, that jdoe and jdoe/admin are unique principals. See also primary, principal name, service principal, user principal.

A security service that, in addition to user authentication, provides for the validity of transmitted data through cryptographic checksumming. See also authentication, privacy.

A postdated ticket that has not yet become usable. An invalid ticket is rejected by an application server until it becomes validated. To be validated, an invalid ticket 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.

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.

An authentication service, the protocol that is used by that service, or the code that is used to implement that service.

The Solaris Kerberos implementation that is closely based on Kerberos V5 implementation.

While technically different, "Kerberos" and "Kerberos V5" are often used interchangeably in the Kerberos documentation.
Kerberos (also spelled Cerberus) was a fierce, three-headed mastiff who guarded the gates of Hades in Greek mythology.

**Kerberos policy** A set of rules that governs password usage in the Kerberos service. Policies can regulate principals' accesses, or ticket parameters, such as lifetime.

**Key** 1. Generally, one of two main types of keys:
   - A symmetric key – An encryption key that is identical to the decryption key. Symmetric keys are used to encrypt files.
   - An asymmetric key or public key – A key that is used in public key algorithms, such as Diffie-Hellman or RSA. Public keys include a private key that is known only by one user, a public key that is used by the server or general resource, and a private-public key pair that combines the two. A private key is also called a secret key. The public key is also called a shared key or common key.

2. An entry (principal name) in a keytab file. See also keytab file.

3. In Kerberos, an encryption key, of which there are three types:
   - A private key – An encryption key that is shared by a principal and the KDC, and distributed outside the bounds of the system. See also private key.
   - A service key – This key serves the same purpose as the private key, but is used by servers and services. See also service key.
   - A session key – A temporary encryption key that is used between two principals, with a lifetime limited to the duration of a single login session. See also session key.

**Keytab file** A key table file that contains 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 that tracks a particular key in order of generation. The highest kvno is the latest and most current key.

**Limit set** The outside limit of what privileges are available to a process and its children.

**Mac** 1. See message authentication code (MAC).

2. Also called labeling. In government security terminology, MAC is Mandatory Access Control. Labels such as Top Secret and Confidential are examples of MAC. MAC contrasts with DAC, which is Discretionary Access Control. UNIX permissions are an example of DAC.

3. In hardware, the unique machine address on a LAN. If the machine is on an Ethernet, the MAC is the Ethernet address.
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>master KDC</strong></td>
<td>The main KDC in each realm, which includes a Kerberos administration server, <code>kadmind</code>, and an authentication and ticket-granting daemon, <code>krb5kdc</code>. Each realm must have at least one master KDC, and can have many duplicate, or slave, KDCs that provide authentication services to clients.</td>
</tr>
<tr>
<td><strong>MD5</strong></td>
<td>An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest.</td>
</tr>
<tr>
<td><strong>mechanism</strong></td>
<td>1. A software package that specifies cryptographic techniques to achieve data authentication or confidentiality. Examples: Kerberos V5, Diffie–Hellman public key.</td>
</tr>
<tr>
<td></td>
<td>2. In the Solaris Cryptographic Framework, an implementation of an algorithm for a particular purpose. For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.</td>
</tr>
<tr>
<td><strong>message authentication code (MAC)</strong></td>
<td>MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping.</td>
</tr>
<tr>
<td><strong>message digest</strong></td>
<td>A message digest is a hash value that is computed from a message. The hash value almost uniquely identifies the message. A digest is useful for verifying the integrity of a file.</td>
</tr>
<tr>
<td><strong>minimization</strong></td>
<td>The installation of the minimal operating system that is necessary to run the server. Any software that does not directly relate to the operation of the server is either not installed, or deleted after the installation.</td>
</tr>
<tr>
<td><strong>name service scope</strong></td>
<td>The scope in which a role is permitted to operate, that is, an individual host or all hosts that are served by a specified name service such as NIS, NIS+, or LDAP. Scopes are applied to Solaris Management Console toolboxes.</td>
</tr>
<tr>
<td><strong>network application server</strong></td>
<td>A server that provides a network application, such as <code>ftp</code>. A realm can contain several network application servers.</td>
</tr>
<tr>
<td><strong>nonattributable audit event</strong></td>
<td>An audit event whose initiator cannot be determined, such as the AUE_BOOT event.</td>
</tr>
<tr>
<td><strong>NTP</strong></td>
<td>Network Time Protocol. Software from the University of Delaware that enables you to manage precise time or network clock synchronization, or both, in a network environment. You can use NTP to maintain clock skew in a Kerberos environment. See also clock skew.</td>
</tr>
<tr>
<td><strong>PAM</strong></td>
<td>Pluggable Authentication Module. A framework that allows for multiple authentication mechanisms to be used without having to recompile the services that use them. PAM enables Kerberos session initialization at login.</td>
</tr>
<tr>
<td><strong>passphrase</strong></td>
<td>A phrase that is used to verify that a private key was created by the passphrase user. A good passphrase is 10–30 characters long, mixes alphabetic and numeric characters, and avoids simple prose and simple names. You are prompted for the passphrase to authenticate use of the private key to encrypt and decrypt communications.</td>
</tr>
</tbody>
</table>
password policy

The encryption algorithms that can be used to generate passwords. Can also refer to more general issues around passwords, such as how often the passwords must be changed, how many mis-entries are permitted, and other security considerations. Security policy requires passwords. Password policy might require passwords to be encrypted with the MD5 algorithm, and might make further requirements related to password strength.

permitted set

The set of privileges that are available for use by a process.

policy

Generally, a plan or course of action that influences or determines decisions and actions. For computer systems, policy typically means security policy. Your site's security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. For example, security policy might require that systems be audited, that devices be protected with privileges, and that passwords be changed every six weeks.

For the implementation of policy in specific areas of the Solaris OS, see audit policy, policy in the cryptographic framework, device policy, Kerberos policy, password policy, and RBAC policy.

policy in the cryptographic framework

In the Solaris Cryptographic Framework, policy is the disabling of existing cryptographic mechanisms. The mechanisms then cannot be used. Policy in the cryptographic framework might prevent the use of a particular mechanism, such as CKM_DES_CBC, from a provider, such as DES.

postdated ticket

A postdated ticket does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to run late at night, since the ticket, if stolen, cannot be used until the batch job is run. When a postdated ticket is issued, it is issued as invalid 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 the postdated ticket is marked renewable, its lifetime is normally set to be equal to the duration of the full life time 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

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. In other words, 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

1. The name of a principal, in the format primary/instance@REALM. See also instance, primary, realm.

2. (RPCSEC_GSS API) See client principal, server principal.
<table>
<thead>
<tr>
<th><strong>privacy</strong></th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>private key</strong></td>
<td>A key that 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.</td>
</tr>
<tr>
<td><strong>private-key encryption</strong></td>
<td>In private-key encryption, the sender and receiver use the same key for encryption. See also public-key encryption.</td>
</tr>
<tr>
<td><strong>privilege</strong></td>
<td>A discrete right on a process in a Solaris system. Privileges offer a finer-grained control of processes than does root. Privileges are defined and enforced in the kernel. For a full description of privileges, see the privileges(5) man page.</td>
</tr>
<tr>
<td><strong>privilege model</strong></td>
<td>A stricter model of security on a computer system than the superuser model. In the privilege model, processes require privilege to run. Administration of the system can be divided into discrete parts that are based on the privileges that administrators have in their processes. Privileges can be assigned to an administrator’s login process. Or, privileges can be assigned to be in effect for certain commands only.</td>
</tr>
<tr>
<td><strong>privilege set</strong></td>
<td>A collection of privileges. Every process has four sets of privileges that determine whether a process can use a particular privilege. See limit set, effective set set, permitted set set, and inheritable set set. Also, the basic set set of privileges is the collection of privileges that are assigned to a user’s process at login.</td>
</tr>
<tr>
<td><strong>privileged application</strong></td>
<td>An application that can override system controls. The application checks for security attributes, such as specific UIDs, GIDs, authorizations, or privileges.</td>
</tr>
<tr>
<td><strong>profile shell</strong></td>
<td>In RBAC, a shell that enables a role (or user) to run from the command line any privileged applications that are assigned to the role’s rights profiles. The profile shells are pfsh, pfcsh, and pfksh. They correspond to the Bourne shell (sh), C shell (csh), and Korn shell (ksh), respectively.</td>
</tr>
<tr>
<td><strong>provider</strong></td>
<td>In the Solaris Cryptographic Framework, a cryptographic service that is provided to consumers. PKCS #11 libraries, kernel cryptographic modules, and hardware accelerators are examples of providers. Providers plug in to the Solaris Cryptographic Framework, so are also called plugins. For examples of consumers, see consumer.</td>
</tr>
<tr>
<td><strong>proxiable ticket</strong></td>
<td>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 a proxiable ticket to obtain a service ticket to another service, but it cannot obtain a ticket-granting ticket. The difference between a proxiable ticket and a forwardable ticket is that a proxiable ticket is only valid for a single operation. See also forwardable ticket.</td>
</tr>
</tbody>
</table>
public-key encryption
An encryption scheme in which each user has two keys, one public key and one private key. 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. The Kerberos service is a private-key system. See also private-key encryption.

QOP
Quality of Protection. A parameter that is used to select the cryptographic algorithms that are used in conjunction with the integrity service 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 responsibilities.

RBAC policy
The security policy that is associated with a command. Currently, suser and solaris are the valid policies. The solaris policy recognizes privileges and setuid security attributes. The suser policy recognizes only setuid security attributes. Trusted Solaris™ systems, which can interoperate with a Solaris system, provide a tsol policy, which recognizes privileges, setuid security attributes, and labels on processes.

realm
1. The logical network that is served by a single Kerberos database and a set of Key Distribution Centers (KDCs).
2. The third part of a principal name. For the principal name jdoe/admin@ENG.EXAMPLE.COM, the realm is ENG.EXAMPLE.COM. See also principal name.

relation
A configuration variable or relationship that is defined in the kdc.conf or krb5.conf files.

renewable ticket
Because having tickets with very long lives is a security risk, tickets can be designated as renewable. A renewable ticket has two expiration times: a) the time at which the current instance of the ticket expires, and b) maximum lifetime for any ticket. If a client wants to continue to use a ticket, the client renews the ticket 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 that holds the ticket wants to keep it for more than an hour, the client must renew the ticket. When a ticket reaches the maximum ticket lifetime, it automatically expires and cannot be renewed.

rights profile
Also referred to as a right or a 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 security attributes, and other rights profiles.

role
A special identity for running privileged applications that only assigned users can assume.

RSA
A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.

SEAM
Sun Enterprise Authentication Mechanism. The product name for the initial versions of a system for authenticating users over a network, based on the Kerberos V5 technology that was developed at the
Massachusetts Institute of Technology. The product is now called the Kerberos service. SEAM refers to parts the Kerberos service that were not included in various Solaris releases.

**secret key**
See private key.

**Secure Shell**
A special protocol for secure remote login and other secure network services over an insecure network.

**security attributes**
In RBAC, overrides to security policy that enable an administrative command to succeed when the command is run by a user other than superuser. In the superuser model, the setuid and setgid programs are security attributes. When these attributes are applied to a command, the command succeeds no matter who runs the command. In the privilege model, security attributes are privileges. When a privilege is given to a command, the command succeeds. The privilege model is compatible with the superuser model, in that the privilege model also recognizes the setuid and setgid programs as security attributes.

**security flavor**
See flavor.

**security mechanism**
See mechanism.

**security policy**
See policy.

**security service**
See service.

**seed**
A numeric starter for generating random numbers. When the starter originates from a random source, the seed is called a random seed.

**server**
A principal that provides a resource to network clients. For example, if you rlogin to the machine central.example.com, then that machine is the server that provides the rlogin service. See also service principal.

**server principal**
(RPCSEC_GSS API) A principal that provides a service. The server principal is stored as an ASCII string in the form service@host. See also client principal.

**service**
1. A resource that is provided to network clients, often by more than one server. For example, if you rlogin to the machine central.example.com, then that machine is the server that provides the rlogin service.

2. A security service (either integrity or privacy) that provides a level of protection beyond authentication. See also integrity and privacy.

**service key**
An encryption key that is shared by a service principal and the KDC, and is distributed outside the bounds of the system. See also key.
service principal  A principal that provides Kerberos authentication for a service or services. For service principals, the primary name is a name of a service, such as `ftp`, and its instance is the fully qualified host name of the system that provides the service. See also host principal, user principal.

session key  A key that is 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. The lifetime of a session key is limited to a single login session. See also key.

SHA1  Secure Hashing Algorithm. The algorithm operates on any input length less than $2^{64}$ to produce a message digest. The SHA1 algorithm is input to DSA.

slave KDC  A copy of a master KDC, which is capable of performing most functions of the master. Each realm usually has several slave KDCs (and only one master KDC). See also KDC, master KDC.

software provider  In the Solaris Cryptographic Framework, a kernel software module or a PKCS #11 library that provides cryptographic services. See also provider.

stash file  A stash file contains an encrypted copy of the master key for the KDC. This master key is used when a server is rebooted to automatically authenticate the KDC before it starts the `kadmind` and `krb5kdc` processes. Because the stash file includes the master key, the stash file and any backups of it should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.

superuser model  The typical UNIX model of security on a computer system. In the superuser model, an administrator has all-or-nothing control of the machine. Typically, to administer the machine, a user becomes superuser (`root`) and can do all administrative activities.

TGS  Ticket-Granting Service. That portion of the KDC that is responsible for issuing tickets.

TGT  Ticket-Granting Ticket. A ticket that is issued by the KDC that enables a client to request tickets for other services.

ticket  An information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains the principal name of the service, the principal name of the user, the IP address of the user’s host, a time stamp, and a value that defines 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 it is presented along with a fresh authenticator. See also authenticator, credential, service, session key.

ticket file  See credential cache.

user principal  A principal that is attributed to a particular user. A user principal’s primary name is a user name, and its optional instance is a name that is used to describe the intended use of the corresponding credentials (for example, `jdoe` or `jdoe/admin`). Also known as a user instance. See also service principal.
| **virtual private network (VPN)** | A network that provides secure communication by using encryption and tunneling to connect users over a public network. |
Index

Numbers and Symbols

*(asterisk)*
- checking for in RBAC authorizations, 211
- deviceAllocate file, 92, 93
- wildcard character
  - inASET, 157, 159
  - in RBAC authorizations, 218, 221
@ (at sign), deviceAllocate file, 93
\ (backslash)
- deviceAllocate file, 93
- deviceMaps file, 92
^ (caret) in audit class prefixes, 566
.(dot)
- authorization name separator, 218
- displaying hidden files, 129
- path variable entry, 47
=(equal sign), file permissions symbol, 124
- (minus sign)
- audit class prefix, 566
- file permissions symbol, 124
- file type symbol, 120
- syslog file, 70
+ (plus sign)
- ACL entry, 135
- audit class prefix, 566
- file permissions symbol, 124
- syslog file, 70
# (pound sign)
- deviceAllocate file, 93
- deviceMaps file, 92
? (question mark), ASET tune files, 159
; (semicolon)
- deviceAllocate file, 92
- separator of security attributes, 224
$ (dollar sign), parent shell process number, 230
\$ (double dollar sign), parent shell process number, 230
[] (square brackets), bsmrecord output, 570
> (redirect output), preventing, 47
>> (append output), preventing, 47
~/gkadmin file, description, 485
~/k5login file, description, 485
~/rhosts file, description, 333
~/shosts file, description, 333
~/ssh/authorized_keys file
  - description, 332
  - override, 334
~/ssh/config file
  - description, 333
  - override, 333
~/ssh/environment file, description, 333
~/ssh/id_dsa file, override, 334
~/ssh/id_rsa file, override, 334
~/ssh/identity file, override, 334
~/ssh/known_hosts file
  - description, 332
  - override, 334
~/ssh/rc file, description, 333
3des–cbc encryption algorithm, ssh_config file, 326
3des encryption algorithm, ssh_config file, 326

A
- -a option
  - bsmrecord command, 543
  - digest command, 258
  - encrypt command, 261
  - getfacl command, 139
  - Kerberized commands, 478
-a option (Continued)
  mac command, 260
  smrole command, 195-196
-A option, audit reduce command, 546
absolute mode
  changing file permissions, 124, 132-133
  changing special file permissions, 133-134
  description, 123
  setting special permissions, 125
access
  control lists
    See ACL
  getting to server
    with Kerberos, 494-497
  granting to your account, 476-477
  login authentication with Solaris Secure Shell, 316-317
  obtaining for a specific service, 496-497
  restricting for
    devices, 44-46 76
    system hardware, 72-73
  restricting for KDC servers, 415
root access
  displaying attempts on console, 70-71
  monitoring su command attempts, 46, 70
  preventing login (RBAC), 198-199
  restricting, 51, 70-71
Secure RPC authentication, 277
security
  ACLs, 50, 125-127
  controlling system usage, 46-50
  devices, 76
  file access restriction, 48
  firewall setup, 54
  login access restrictions, 39
  login authentication, 316-317
  login control, 39
  monitoring system usage, 49
  network control, 51-55
  NFS client-server, 279-281
  PATH variable setting, 47
  peripheral devices, 44
  physical security, 38
  remote systems, 303
  reporting problems, 55
  root login tracking, 46
  saving failed logins, 61
access, security (Continued)
  setuid programs, 48
    system hardware, 72-73
  sharing files, 51
  system logins, 42
access control list
  See ACL
ACL
  changing entries, 137-138
  checking entries, 135
  commands, 127
  copying ACL entries, 137
  default entries for directories, 127
  deleting entries, 127, 138
  description, 50, 125-127
  directory entries, 127
  displaying entries, 127, 139-140
  format of entries, 125-127
  kadm5.acl file, 442, 444, 448
  modifying entries, 137
  restrictions on copying entries, 126
  setting entries, 135-137
  setting on a file, 135
  task map, 134-140
  user procedures, 134-140
  valid file entries, 126-127
acl audit token, format, 572
add_drv command, description, 89
adding
  ACL entries, 135-137
  administration principals (Kerberos), 365
  allocatable device, 80
  attributes to a rights profile, 206-208
  audit classes, 530-531
  audit directories, 533-535
  audit policy, 537-538
  auditing of roles, 197
  auditing of zones, 514-517
  cryptomgt role, 197
  custom roles (RBAC), 195-196
  customized role, 195-196
  DH authentication to mounted file systems, 282
  dial-up passwords, 63-65
  hardware provider mechanisms and features, 273
  keys for DH authentication, 282-283
adding (Continued)
library plugin, 267-268
local user, 198
new rights profile, 206-208
Operator role, 192
PAM modules, 293
password encryption module, 68-69
plugins to cryptographic framework, 266-268
privileges directly to user or role, 234-235
privileges to command, 233-234
RBAC properties to legacy applications, 210-211
rights profiles with Solaris Management Console, 207
roles
for particular profiles, 191-193
from command line, 193-196
to a user, 193
with limited scope, 192-193
security attributes to legacy applications, 210-211
security-related role, 197
security-related roles, 192
security to devices, 77-78, 80-84
security to system hardware, 72-73
service principal to keytab file (Kerberos), 463-464
software provider, 266-268
System Administrator role, 191-192
user-level software provider, 267-268
admin_server section, krb5.conf file, 364
administering
ACLs, 134-140
auditing
audit classes, 507-508, 564 565
audit events, 506
audit files, 548-550
audit records, 508
audit trail overflow prevention, 551
audit reduce command, 545-546
cost control, 519
description, 504
efficiency, 521
process preselection mask, 554
reducing storage-space requirements, 520
task map, 523
in zones, 564
auditing in zones, 514-515
cryptographic framework, 252
cryptographic framework and zones, 254
administering (Continued)
cryptographic framework task map, 263
device allocation, 79-80
device policy, 76
dial-up logins, 63
file permissions, 128-129 129-134
Kerberos
keytabs, 461-467
policies, 449-456
principals, 436-448
metaslot, 252
NFS client-server file security, 279-281
password algorithms, 65
privileges, 230
properties of a role, 204-206
RBAC properties, 206-208
remote logins with Solaris Secure Shell, 312-314
rights profiles, 206-208
roles, 191-193
roles to replace superuser, 189-190
Secure RPC task map, 281-282
Solaris Secure Shell
clients, 325-326
overview, 323-325
servers, 326
task map, 307
without privileges, 180
administrative (old) audit class, 565
administrative audit class, 565
AES kernel provider, 264
aes128-cbc encryption algorithm, ssh_config file, 326
eaes128-ctr encryption algorithm, ssh_config file, 326
agent daemon, Solaris Secure Shell, 316-317
ahlt audit policy
description, 518
setting, 537
algorithms
definition in cryptographic framework, 250
listing in the cryptographic framework, 264-266
password
configuration, 66-67
password encryption, 40
all, in user audit fields, 561
All (RBAC), rights profile, 217
all audit class
cautions for using, 566
all audit class (Continued)
description, 565
allhard string, audit_warn script, 563
allocate command
allocate error state, 91
authorizations for, 91
authorizations required, 227
description, 91
tape drive, 85
user authorization, 80
using, 85
allocate error state, 91
allocating devices
by users, 85
forcibly, 82
task map, 84-85
AllowGroups keyword, sshd_config file, 326
AllowTcpForwarding keyword
changing, 311
sshd_config file, 326
AllowUsers keyword, sshd_config file, 326
allsoft string, audit_warn script, 562
ALTSHSEll in Solaris Secure Shell, 331
always-audit classes
audit_user database, 561
process preselection mask, 567
analysis, praudit command, 556
appending arrow (>>, preventing appending, 47
application audit class, 565
application server, configuring, 375-376
arbitrary audit token
format, 572
item size field, 572
print format field, 573
arcfour encryption algorithm, ssh_config file, 326
ARCFOUR kernel provider, 264
Archive tape drive device-clean script, 93
archiving, audit files, 551
arg audit token, format, 573
arge audit policy
and exec_env token, 575
description, 518
argv audit policy
and exec_args token, 575
description, 518
ASET
aliases file
description, 151
examples, 159
UID_ALIASES variable, 153
set command
interactive version, 160-161
-p option, 162
starting, 144
aset.restore command, 154
ASETDIR variable, 156
asetenv file, 152
ASETSECLEVEL variable, 156
CKLISTPATH level variable, 158
collecting reports, 162-163
configuring, 152-154 154
description, 48,143-160
environment file, 152
environment variables, 155
error messages, 164
evaluation log, 147
master files, 145 151
NFS services and, 155
PERIODIC_SCHEDULE variable, 153 157
restoring original system state, 154
running ASET periodically, 161-162
running interactively, 160-161
running periodically, 161-162
scheduling ASET execution, 153,157
stopping from running periodically, 162
task map, 160-163
tASKS variable, 152,157
troubleshooting, 164
tune file examples, 158
tune files, 151, 154
uid_aliases file, 151
UID_ALIASES variable, 151, 153,157
working directory, 156
YPCHECK variable, 154, 158
assigning
privileges to commands in a rights profile, 233-234
privileges to commands in a script, 236-237
privileges to user or role, 234-235
role to a user, 191, 193
role to a user locally, 196-197
assuming role
how to, 200
in a terminal window, 200-202
in Solaris Management Console, 202-203
Primary Administrator, 201
root, 201
System Administrator, 201-202
asterisk (*)
checking for in RBAC authorizations, 211
device_allocate file, 92, 93
wildcard character
in ASET, 157-159
in RBAC authorizations, 218, 221
at command, authorizations required, 226
at sign (@), device_allocate file, 93
atq command, authorizations required, 226
attribute audit token, 574
attributes, keyword in BART, 115
audio devices, security, 94
audit administration audit class, 565
audit characteristics
audit ID, 567
process preselection mask, 554
processes, 567
session ID, 567
terminal ID, 568
user process preselection mask, 567
audit_class file, adding a class, 530-531
audit_class file, description file, 559
audit class preselection, effect on public objects, 506
audit classes
adding, 530-531
definitions, 565
description, 505, 506
entries in audit_control file, 559
exceptions in audit_user database, 561
exceptions to system-wide settings, 507
mapping events, 508
modifying default, 530-531
overview, 507-508
prefixes, 566
preselecting, 524-526
preselection, 506
process preselection mask, 567
setting system-wide, 564
syntax, 566
audit classes (Continued)
  system-wide, 559
audit command
description, 554
preselection mask for existing processes (-s option), 541
rereading audit files (-s option), 554
resetting directory pointer (-n option), 554
updating auditing service, 541-542
audit configuration file, See audit_control file
audit_control file
  audit daemon rereading after editing, 541
  changing kernel mask for nonattributable events, 541
  configuring, 524-526
description, 559
test cases, 559
test cases and zones, 564
examples, 560
exceptions in audit_user database, 561
flags line
  process preselection mask, 567
  minfree warning, 562
overview, 504
prefixes in flags line, 566
syntax problem, 563
AuditControlrights profile, 563
audit daemon, See auditd daemon
audit directory
  creating, 535
description, 506
partitioning for, 533-535
sample structure, 555
audit_event file
  changing class membership, 531-532
description, 506
audit events
  audit_event file, 506
  changing class membership, 531-532
description, 506
mapping to classes, 508
selecting from audit trail, 546-548
selecting from audit trail in zones, 564
summary, 506
viewing from binary files, 548-550
audit files
  audit reduce command, 555
audit files (Continued)
  combining, 545-546, 555
  configuring, 524-532
  copying messages to single file, 548
  managing, 551
  minimum free space for file systems, 560
  names, 569
  order for opening, 560
  partitioning disk for, 533-535
  printing, 549
  reducing, 545-546, 555
  reducing storage-space requirements, 520, 521
  switching to new file, 554
  time stamps, 569
audit ID
  mechanism, 567
  overview, 503-504
audit logs
  See also audit files
  comparing binary and textual, 508
  configuring textual audit logs, 526-528
  in text, 560
  modes, 508
audit messages, copying to single file, 548
audit.notice entry, syslog.conf file, 527
audit policy
  defaults, 517-519
  description, 506
  effects of, 517-519
  public, 519
  setting, 536-538
  setting ahlt, 537
  setting in global zone, 564
  setting per zone, 538
audit preselection mask, modifying for individual users, 528-530
audit records (Continued)
  displaying in XML format, 550
  events that generate, 504
  format, 569
  formatting example, 543
  merging, 545-546
  overview, 508
  reducing audit files, 545-546
  sequence of tokens, 569
  syslog.conf file, 505
  /var/adm/auditlog file, 527
Audit Review rights profile, 563
audit session ID, 567
audit.startup script
  configuring, 536-538
  description, 561
audit threshold, 560
audit tokens
  See also individual audit token names
  audit record format, 569
  description, 506, 508
  format, 571
  list of, 571
  new in current release, 511
audit trail
  analysis costs, 520
  analysis with praudit command, 556
  cleaning up not terminated files, 550-551
  creating
    auditd daemon's role, 554
    description, 506
  effect of audit policy on, 517
  events included, 508
  merging all files, 555
  monitoring in real time, 521
  no public objects, 506
  overview, 504
  preventing overflow, 551
  selecting events from, 546-548
  viewing events from, 548-550
  viewing events from different zones, 564
audit_user database
  exception to system-wide audit classes, 507
  prefixes for classes, 566
  process preselection mask, 567
  specifying user exceptions, 528-530
audit_user database (Continued)
  user audit fields, 561
audit_warn script
  auditd daemon execution of, 554
  conditions invoking, 562
  configuring, 536
  description, 562
  strings, 563
auditconfig command
  audit classes as arguments, 507, 564
  description, 558
  prefixes for classes, 566
  setting audit policy, 537-538
auditd daemon
  audit trail creation, 554, 568
  audit_warn script
    description, 562
    execution of, 554
  functions, 554
  order audit files are opened, 560
  rereading information for the kernel, 541
  rereading the audit_control file, 541
auditing
  changes in current release, 510-511
  changes in device policy, 78
  configuring in global zone, 514, 537
  device allocation, 84
  disabling, 540-541
  enabling, 539-540
  planning, 514-517
  planning in zones, 514-515
  preselection definition, 506
  privileges and, 245
  rights profiles for, 563
  roles, 197
  updating information, 541-542
  zones and, 564
audit_log file, text audit records, 527
audit_reduce command (Continued)
  options, 555
  selecting audit records, 546-548
  timestamp use, 569
  trailer tokens, and, 586
  using lowercase options, 546
  using uppercase options, 545
  without options, 555
auth_attr database
  description, 221-222
  summary, 219
AUTH DES authentication, See AUTH_DH authentication
AUTH_DH authentication, and NFS, 277
authentication
  AUTH_DH client-server session, 279-281
  configuring cross-realm, 372-374
  description, 52-53
  DH authentication, 278-281
  disabling with -X option, 479
  Kerberos and, 339
  name services, 277
  network security, 52-53
  NFS-mounted files, 286
  overview of Kerberos, 493-494
  Secure RPC, 277
  Solaris Secure Shell
    methods, 304-306
    process, 324-325
terminology, 488-489
types, 52-53
use with NFS, 277
authentication methods
  GSS-API credentials in Solaris Secure Shell, 304
  host-based in Solaris Secure Shell, 305 307-309
description, 52-53
  keyboard-interactive in Solaris Secure Shell, 305
  password in Solaris Secure Shell, 305
  public keys in Solaris Secure Shell, 305
  Solaris Secure Shell, 304-306
authenticator
  in Kerberos, 489, 495
auth_log file, saving failed login attempts, 62-63
authorizations
  Kerberos and, 339
  types, 52-53
authorizations (RBAC)
  checking for wildcards, 211
authentications (RBAC) (Continued)
  checking in privileged application, 175
  commands that require authentications, 226-227
  database, 219-225
  definition, 174
  delegating, 218
  description, 171, 218
  for allocating device, 80-81
  for device allocation, 91
  granularity, 218
  naming convention, 218
  not requiring for device allocation, 83
  solaris.device.allocate, 80, 91
  solaris.device.revoke, 91
authorized_keys file, description, 332
  AuthorizedKeysFile keyword, sshd_config file, 326
auths command, description, 225
  AUTHS_GRANTED keyword, policy.conf file, 224
auto_transition option, SASL and, 301
Automated Security Enhancement Tool, See ASET
automatic login
  disabling, 479
  enabling, 478
automatically enabling auditing, 561
automating principal creation, 437-438
auxprop_login option, SASL and, 301

B
  -b option, auditreduce command, 547
backup
  Kerberos database, 400-401
  slave KDCs, 356
Banner keyword, sshd_config file, 326
BART
  components, 98
  overview, 97-100
  programmatic output, 117
  security considerations, 101
  task map, 100-101
  verbose output, 117
bart command, 97
bart compare command, 99
bart create command, 98, 101
Basic Audit Reporting Tool, See BART
  basic privilege set, 182
Basic Security Module (BSM)
  See auditing
  See device allocation
Basic Solaris User rights profile, 216
Batchmode keyword, ssh_config file, 326
BindAddress keyword, ssh_config file, 326
binding control flag, PAM, 296
blowfish-cbc encryption algorithm, ssh_config file, 326
Blowfish encryption algorithm
  kernel provider, 264
  policy.conf file, 66-67
  ssh_config file, 326
  using for password, 66-67
Bourne shell, privileged version, 177
bsmconv script
  creating device_maps file, 91-92
  description, 563
  enabling auditing service, 539-540
bsmrecord command
  [] (square brackets) in output, 570
  description, 555
  displaying audit record formats, 543-544
  example, 543
  listing all formats, 543
  listing formats of class, 544
  listing formats of program, 543-544
  optional tokens (()), 570
bsmunconv script, disabling auditing service, 540-541

C
  -c option
    auditreduce command, 547, 548
    bsmrecord command, 544
  -C option, auditreduce command, 546
C shell, privileged version, 177
c2audit:audit_load entry, system file, 559
cache, credential, 494
canon_user_plugin option, SASL and, 301
caret (^) in audit class prefixes, 566
CD-ROM drives
  allocating, 87
  security, 94
cdrw command, authorizations required, 227
ChallengeResponseAuthentication keyword, See KbdInteractiveAuthentication keyword
changepw principal, 462
changing
ACL entries, 137-138
allocatable devices, 83-84
audit_class file, 530-531
audit_control file, 524-526
audit_event file, 531-532
default password algorithm, 65
device policy, 77-78
file ownership, 130-131
file permissions
absolute mode, 132-133
special, 133-134
symbolic mode, 132
group ownership of file, 131
passphrase for Solaris Secure Shell, 314
password algorithm for a domain, 67
password algorithm task map, 65
properties of role, 204-206
rights profile contents, 206-208
rights profile from command line, 207
root user into role, 198-199
secret keys, 279
special file permissions, 133-134
user properties from command line, 210
your password with kpasswd, 474
your password with passwd, 473
CheckHostIP keyword, ssh_config file, 326
chgrp command
description, 120
syntax, 131
chkey command, 279-285
chmod command
changing special permissions, 133-134
description, 120
syntax, 133
choosing, your password, 473
chown command, description, 119
Cipher keyword, sshd_config file, 326
Ciphers keyword, Solaris Secure Shell, 326
cklist.rpt file, 146, 150
CKLISTPATH_level variable (ASET), 158
classes, See audit classes
cleaning up, binary audit files, 550-551
clear protection level, 479
ClearAllForwardings keyword, Solaris Secure Shell port forwarding, 327
client names, planning for in Kerberos, 355-356
ClientAliveCountMax keyword, Solaris Secure Shell port forwarding, 327
ClientAliveInterval keyword, Solaris Secure Shell port forwarding, 327
clients
AUTH_DH client-server session, 279-281
configuring for Solaris Secure Shell, 324-325-326
configuring Kerberos, 382-394
definition in Kerberos, 488
clntrconfig principal, creating, 367
clock skew
Kerberos and, 358, 394-395
clock synchronizing
Kerberos and, 358, 368, 372, 410
cmd audit token, 511, 574
cnt audit policy, description, 518
combining audit files
audit reduce command, 545-546, 555
from different zones, 564
command execution, Solaris Secure Shell, 325
command-line equivalents of SEAM Administration Tool, 432-433
commands
See also individual commands
ACL commands, 127
auditing commands, 553
cryptographic framework commands, 252
determining user’s privileged commands, 239-240
device allocation commands, 90
device policy commands, 89
file protection commands, 119
for administering privileges, 243
Kerberos, 487
RBAC administration commands, 225-226
Secure RPC commands, 279
Solaris Secure Shell commands, 334-336
that assign privileges, 183
that check for privileges, 175
user-level cryptographic commands, 253
common keys
calculating, 280
common keys (Continued)
  DH authentication and, 278-281
components
  BART, 98
  device allocation mechanism, 89
  RBAC, 171-174
  Solaris Secure Shell user session, 325
Compression keyword, Solaris Secure Shell, 327
CompressionLevel keyword, ssh_config file, 327
Computer Emergency Response Team/Coordination Center (CERT/CC), 55
computer security, See system security computing
  DH key, 285
digest of a file, 258-259
MAC of a file, 259-260
secret key, 256-258
configuration decisions
  auditing
    file storage, 515
    policy, 517-519
    who and what to audit, 516-517
    zones, 514-515
Kerberos
  client and service principal names, 355-356
clock synchronization, 358
  database propagation, 357
  mapping host names onto realms, 355
  number of realms, 354
  ports, 356
  realm hierarchy, 354-355
  realm names, 354
  realms, 354-355
  slave KDCs, 356
password algorithm, 40
configuration files (Continued)
  policy.conf file, 40 66-67, 226
  Solaris Secure Shell, 323
  syslog.conf file, 62-63, 245, 559
  system file, 559
  with privilege information, 244-245
configuring
  audit audit policy, 537
  ASET, 152-154, 154
  audit_class file, 530-531
  audit_control file, 524-526
  audit_event file, 531-532
  audit files, 524-532
  audit files task map, 524
  audit policy, 536-538
  audit policy temporarily, 537-538
  audit_startup script, 536-538
  audit trail overflow prevention, 551
  audit_user database, 528-530
  audit_warn script, 536
  auditconfig command, 558
  auditing in zones, 564
  auditing service task map, 532-533
  custom roles, 195-196
device allocation, 79-80
device policy, 76
devices task map, 75-76
DH key for NIS+ user, 284
DH key for NIS user, 285-286
DH key in NIS, 284-285
DH key in NIS+, 282-283
dial-up logins, 63
hardware security, 72-73
host-based authentication for Solaris Secure Shell, 307-309
Kerberos
  adding administration principals, 365
  clients, 382-394
cross-realm authentication, 372-374
  master KDC server, 363-368
  NFS servers, 377-378
  overview, 361-415
  slave KDC server, 368-372
task map, 361-362
name service, 199
password for hardware access, 72-73
Index

configuring (Continued)
  per zone audit policy, 538
  port forwarding in Solaris Secure Shell, 310-311
  RBAC, 188-199
  RBAC task map, 188
  rights profile from command line, 207
  rights profiles, 206-208
  roles, 191-193, 204-206
    from command line, 193-196
  root user as role, 198-199
  Solaris Secure Shell, 307
    clients, 325-326
    servers, 326
  Solaris Secure Shell task map, 307
  ssh-agent daemon, 317
textual audit logs, 526-528

configuring application servers, 375-376
ConnectionAttempts keyword, ssh_config file, 327
console, displaying su command attempts, 70-71
CONSOLE in Solaris Secure Shell, 330
consumers, definition in cryptographic framework, 251
context-sensitive help, SEAM Administration Tool, 433
control flags, PAM, 296
control manifests (BART), 97
controlling
  access to system hardware, 72
  system access, 57-58
  system usage, 46-50
conversation keys
  decrypting in secure RPC, 280
  generating in secure RPC, 279
converting
  audit records to readable format, 550, 556

copying
  ACL entries, 137
  files using Solaris Secure Shell, 319-320
copying audit messages to single file, 548
cost control, and auditing, 519
crammd.so.1 plug-in, SASL and, 300
creating
  audit trail
    auditd daemon, 568
    auditd daemon's role, 554
  credential table, 379
  customized role, 195-196
d_passwd file, 64

creating (Continued)
  dial-up passwords, 63-65
  /etc/d_passwd file, 64
  file digests, 258-259
  keytab file, 366
  local user, 198
  new device-clean scripts, 94-95
  new policy (Kerberos), 442 453-454
  new principal (Kerberos), 442-444
  Operator role, 192
  partitions for binary audit files, 533-535
  passwords for temporary user, 64
  rights profiles, 206-208
  rights profiles with Solaris Management Console, 207
  roles for particular profiles, 191-193
    on command line, 193-196
    with limited scope, 192-193
  root user as role, 198-199
  secret keys
    for encryption, 256-258
    security-related roles, 192
  Solaris Secure Shell keys, 312-314
  stash file, 372-410
  System Administrator role, 191-192
  tickets with kinit, 470
cred database
  adding client credential, 283
  adding user credential, 284
  DH authentication, 278-281
cred table
  DH authentication and, 279
  information stored by server, 280
credential
  cache, 494
  description, 280-489
  obtaining for a server, 495-496
  obtaining for a TGS, 494-495
  or tickets, 341
credential table, adding single entry to, 379-380
  credentials, mapping, 357
crontab files
  authorizations required, 227
  running ASET periodically, 144
  stop running ASET periodically, 162
cross-realm authentication, configuring, 372-374
Index

CRYPT_ALGORITHMS_ALLOW keyword, policy.conf file, 41
CRYPT_ALGORITHMS_DEPRECATE keyword, policy.conf file, 41
crypt_bsdbsf password algorithm, 41
crypt BSDmd5 password algorithm, 41
crypt command, file security, 50
crypt.conf file
  changing with new password module, 68-69
  third-party password modules, 68-69
CRYPT_DEFAULT keyword, policy.conf file, 41
CRYPT_DEFAULT system variable, 66
crypt_SunMD5 password algorithm, 41
crypt_unix password algorithm, 41 66-69
Crypto Management (RBAC)
  creating role, 197
  use of rights profile, 268 269
cryptoadm command
  description, 252
  disabling cryptographic mechanisms, 268, 269
  disabling hardware mechanisms, 272-274
  installing PKCS #11 library, 267
  listing providers, 264
  -m option, 268, 269
  -p option, 268, 270
  restoring kernel software provider, 270
cryptoadm install command, installing PKCS #11 library, 267
 cryptographic framework
  administering with role, 197
  connecting providers, 253
  consumers, 250
  cryptoadm command, 252
  definition of terms, 250
  description, 250
  elfsign command, 252, 253
  error messages, 262
  installing providers, 254
  interacting with, 252
  listing providers, 264-266
  PKCS #11 library, 250
  providers, 250
  refreshing, 274
  registering providers, 254
  restarting, 274
  signing providers, 254
  task maps, 255
cryptographic framework (Continued)
  user-level commands, 253
  zones and, 254, 274
cryptographic services, See cryptographic framework
Cryptoki, See PKCS #11 library
csh command, privileged version, 177
.cshrc file, path variable entry, 47
Custom Operator (RBAC), creating role, 195-196
customizing, manifests, 103-106
customizing a report (BART), 112-114

D
-d option
  auditreduce command, 548
  getfacl command, 139
  praudit command, 557
  setfacl command, 138
-D option
  auditreduce command, 546
  ppriv command, 232
d_passwd file
  creating, 64
  description, 43
  disabling dial-up logins temporarily, 65
daemons
  auditd, 554
  kcf, 252
  keyserv, 282
  nscd (name service cache daemon), 191 225
  rpc.nispasswd, 67
  running with privileges, 180
  ssh-agent, 316-317, 317
  sshd, 323-325
table of Kerberos, 487-488
  vold, 81
Data Encryption Standard, See DES encryption
data forwarding, Solaris Secure Shell, 325
databases
  audit_user, 561
  auth_attr, 221-222
  backing up and propagating KDC, 400-401
  creating KDC, 365
  cred for Secure NFS, 278, 283
  exec_attr, 223-224
Index

Index 613

databases (Continued)
  KDCpropagation, 357
  prof.attr, 222-223
  public key for Secure NFS, 278
  RBAC, 219-225
  secret keys, 279
  user.attr, 220-221
  with privilege information, 244-245
deallocate command
  allocate error state, 91
  authorizations for, 91
  authorizations required, 227
  description, 91
  device-clean scripts and, 94-95
  using, 88
deallocating
  devices, 88
  forcibly, 82
  microphone, 88
debugging, privileges, 232
debugging sequence number, 583
decrypt command
  description, 253
  syntax, 261
decrypting
  conversation keys, 280
  files, 261
  secret keys, 279
default/login file, description, 332
default_realm section, krb5.conf file, 364
defaultpriv keyword, user_attr database, 244
defaults
  ACL entries for directories, 127
  audit_startup script, 561
  praudit output format, 557
  privilege settings in policy.conf file, 244
  system-wide auditing, 564
  system-wide in policy.conf file, 40
  umask value, 123
delegating, RBAC authorizations, 218
delete_entry command, ktutil command, 467
deleting
  ACL entries, 127, 138
  archived audit files, 551
  audit files, 545
  host’s service, 467
deleting (Continued)
  not_terminated audit files, 550-551
  policies (Kerberos), 456
  principal (Kerberos), 445-446
  rights profiles, 206
  DenyGroups keyword, sshd_config file, 327
  DenyUsers keyword, sshd_config file, 327
  DES encryption
    kernel provider, 264
    secure NFS, 278
  destroying, tickets with kdestroy, 472
determining
  files with setuid permissions, 140
  if file has ACL, 135
  privileges on a process, 230-232
  privileges task map, 237
  /dev/arp device, getting IP MIB-II information, 78-79
  /dev/urandom device, 256-258
devfsadm command, description, 89
device_allocate file
  description, 92-93
  format, 92
  sample, 83, 92
device allocation
  adding devices, 79-80
  allocatable devices, 93
  allocate command, 91
  allocate error state, 91
  allocating devices, 85
  auditing, 84
  authorizations for commands, 91
  authorizing users to allocate, 80-81
  changing allocatable devices, 83-84
  commands, 90
  components of mechanism, 89
  configuration file, 91
  deallocate command, 91
    device-clean scripts and, 94-95
    using, 88
  deallocating devices, 88
device_allocate file, 92-93
device-clean scripts
  audio devices, 94
  CD-ROM drives, 94
  description, 93-95
  diskette drives, 94
device allocation, device-cleanup scripts (Continued)
  options, 95
  tape drives, 93-94
  writing new scripts, 94-95
device maps file, 91-92
disabling, 540
enabling, 80
examples, 85
forcibly allocating devices, 82
forcibly deallocating devices, 82
making device allocatable, 80
managing devices, 79-80
mounting devices, 86-87
not requiring authorization, 83
preventing, 83-84
requiring authorization, 83-84
task map, 79-80
unmounting allocated device, 88
user procedures, 84-85
using, 84-85
using allocate command, 85
viewing information, 81-82
device-cleanup scripts
  and object reuse, 93-95
  audio devices, 94
  CD-ROM drives, 94
description, 93-95
diskette drives, 94
options, 95
tape drives, 93-94
writing new scripts, 94-95
device management, See device policy
device maps file
  description, 91
  format, 91
  sample entries, 92
device policy
  add_drv command, 89
  auditing changes, 78
  changing, 77-78
  commands, 89
  configuring, 76-79
  kernel protection, 89-95
  managing devices, 76
  overview, 44-46
  removing from device, 78
device policy (Continued)
  task map, 76
  update_drv command, 77-78, 89
  viewing, 76-77
Device Security (RBAC), creating role, 192
devices
  adding device policy, 77-78
  allocating for use, 84-85
  auditing allocation of, 84
  auditing policy changes, 78
  authorizing users to allocate, 80-81
  changing device policy, 77-78
  changing which are allocatable, 83-84
  deallocating a device, 88
/dev/urandom device, 256-258
device allocation
  See device allocation
  forcibly allocating, 82
  forcibly deallocating, 82
  getting IP MIB-II information, 78-79
  listing, 76-77
  listing device names, 81
  login access control, 43
  making allocatable, 80
  managing, 76
  managing allocation of, 79-80
  mounting allocated devices, 86-87
  not requiring authorization for use, 83
  policy commands, 89
  preventing use of all, 83-84
  preventing use of some, 83
  privilege model and, 184
  protecting by device allocation, 44
  protecting in the kernel, 44
  removing policy, 78
  security, 44-46
  superuser model and, 184
  unmounting allocated device, 88
  viewing allocation information, 81-82
  viewing device policy, 76-77
  zones and, 44
dfs tab file
  security modes, 381
  sharing files, 51
DH authentication
  configuring in NIS, 284-285
DH authentication (Continued)
configuring in NIS+, 282-283
description, 278-281
for NIS+ client, 283
for NIS client, 284-285
mounting files with, 286
sharing files with, 286-287
DHCP Management (RBAC), creating role, 192-193
dial-up passwords
creating, 63-65
disabling, 44
disabling temporarily, 65
/etc/d_passwd file, 44
security, 43-44
dial-ups file, creating, 63
Diffie-Hellman authentication, See DH authentication
digest command
description, 253
element, 258
digests
computing for file, 258-259
of files, 258-259
dir line, audit_control file, 560
direct realms, 373-374
directories
See also files
ACL entries, 127
audit_control file definitions, 560
audit directories full, 554, 563
auditd daemon pointer, 554
checklist task setting (ASET), 153, 158
displaying files and related information, 119, 129-130
master files (ASET), 151
mounting audit directories, 568
permissions
defaults, 123
description, 120-121
public directories, 123
reports (ASET), 149
working directory (ASET), 156, 160-161
disabling
abort sequence, 73
audit policy, 536-538
auditing service, 540-541
disabling (Continued)
cryptographic mechanisms, 268
device allocation, 540
dial-up logins temporarily, 65
dial-up passwords, 65
executable stacks, 141-142
eexecutables that compromise security, 128
hardware mechanisms, 272-274
keyboard abort, 73
keyboard shutdown, 73
logging of executable stack messages, 142
logins temporarily, 60
programs from using executable stacks, 141-142
remote root access, 70-71
service on a host (Kerberos), 466-467
system abort sequence, 73
user logins, 60
disk partitioning, for binary audit files, 533-535
disk-space requirements, 520
diskette drives
allocating, 86-87
device-clean scripts, 94
displaying
ACL entries, 127, 135, 139-140
allocatable devices, 81-82
ASET task status, 145, 148
audit policies, 536
audit record formats, 543-544
audit records, 548-550
audit records in XML format, 550
device policy, 76-77
file information, 129-130
files and related information, 119
format of audit records, 543-544
providers in the cryptographic framework, 264-266
roles you can assume, 200, 226
root access attempts, 70-71
selected audit records, 545-546
su command attempts, 70-71
sublist of principals (Kerberos), 439
user’s login status, 58-59, 59
users with no passwords, 59-60
dmideinfo command, 91
DNS, Kerberos and, 355-356
domain_realm section
krb5.conf file, 355, 364
dot(,)
  authorization name separator, 218
  displaying hidden files, 129
  path variable entry, 47
double dollar sign ($$), parent shell process number, 230
DSAuthentication keyword, See PubkeyAuthentication keyword
DTD for praudit command, 557
  .dtprofile script, use in Solaris Secure Shell, 317
duplicating, principals (Kerberos), 444
DynamicForward keyword, ssh_config file, 327

E
  -e option
    auditreduce command, 548
    ppripriv command, 232
    ebussy string, audit_warn script, 563
    eeprom command, 38, 72-73
    eeprom.rpt file, 147, 150
effective privilege set, 181
efficiency, auditing and, 521
eject command, device cleanup and, 94
elfsign command
description, 252, 253
enabling
  auditing, 539-540
  auditing service, 539-540
  auditing service task map, 532-533
  cryptographic mechanisms, 269
device allocation, 80
  Kerberized applications only, 414
  kernel software provider use, 270
  keyboard abort, 73
  mechanisms and features on hardware provider, 273
encrypt command
description, 253
  error messages, 262
  syntax, 256
  troubleshooting, 262
encrypting
  communications between hosts, 315
  encrypt command, 261-263
description, 254, 261-263
  network traffic between hosts, 303-306
  encrypting (Continued)
    passwords, 65
    private key of NIS user, 285
    Secure NFS, 278
    using user-level commands, 253
encryption
  DES algorithm, 278
  generating symmetric key for, 256-258
  installing third-party password modules, 68-69
  list of password algorithms, 40
  password algorithm, 40
  privacy service, 339
  specifying algorithms in ssh_config file, 326
  specifying password algorithm
  locally, 65
  specifying password algorithms in policy.conf file, 40
  with -x option, 479
ending, signal received during auditing shutdown, 563
env.rpt file, 146, 150
evironment variables
  See also variables
  ASETDIR (ASET), 156
  ASETSECLEVEL (ASET), 156
  audit token for, 575
  CKiSTPATH_level (ASET), 153, 158
  overriding proxy servers and ports, 321
  PATH, 46
  PERIODIC_SCHEDULE (ASET), 153, 157
  presence in audit records, 518, 571
  Solaris Secure Shell and, 330-331
  summary (ASET), 155
  TASKS (ASET), 152, 157
  UID_ALIASES (ASET), 151, 153, 157
  use with ssh-agent command, 334
  YPCHECK (ASET), 154, 158
equal sign (=), file permissions symbol, 124
error messages
  encrypt command, 262
  Kerberos, 417-428
  with kpasswd, 474
errors
  allocate error state, 91
  audit directories full, 554, 563
  internal errors, 563
  EscapeChar keyword, ssh_config file, 327
Index

/etc/d_passwd file
   and /etc/passwd file, 43
   creating, 64
   disabling dial-up logins temporarily, 65
/etc/default/kbd file, 73
/etc/default/login file
   description, 332
   login default settings, 62
   restricting remote root access, 70-71
   Solaris Secure Shell and, 330-331
/etc/default/su file
   displaying su command attempts, 70-71
   monitoring access attempts, 70-71
   monitoring su command, 70
/etc/dfs/dfstab file
   security modes, 381
   sharing files, 51
/etc/dialups file, creating, 63
/etc/group file, ASET checks, 146
/etc/hosts.equiv file, description, 333
/etc/krb5/kadm5.acl file, description, 485
/etc/krb5/kadm5.keytab file, description, 485
/etc/krb5/kdc.conf file, description, 486
/etc/krb5/kpropd.acl file, description, 486
/etc/krb5/kpropd.conf file, description, 486
/etc/krb5/keytab file, description, 486
/etc/krb5/warn.conf file, description, 486
/etc/logindevperm file, 43
/etc/nologin file
   description, 332
   disabling user logins temporarily, 60
/etc/nsswitch.conf file, 39
/etc/pam.conf file
   control flags, 296
   description, 294
   examples, 297
   Kerberos and, 486
   service names, 295
/etc/passwd file, ASET checks, 146
/etc/publickey file, DH authentication and, 278
/etc/security/audit_event file, audit events and, 506
/etc/security/audit_startup file, 561
/etc/security/audit_warn script, 562
/etc/security/bsmconv script, 91-92
   description, 563
/etc/security/crypt.conf file
   changing with new password module, 68-69
   third-party password modules, 68-69
/etc/security/device_allocate file, 92
/etc/security/device_maps file, 91
/etc/security/policy.conf file, algorithms
   configuration, 66-67
/etc/ssh_host_dsa_key.pub file, description, 332
/etc/ssh_host_key.pub file, description, 332
/etc/ssh_host_rsa_key.pub file, description, 332
/etc/ssh/hosts.equiv file, description, 333
/etc/ssh/ssh_config file
   configuring Solaris Secure Shell, 325-326
   description, 333
   host-specific parameters, 330
   keywords, 326-331
   override, 333
/etc/ssh/ssh_host_dsa_key file, description, 332
/etc/ssh/ssh_host_key file
   description, 332
   override, 333
/etc/ssh/ssh_host_rsa_key file, description, 332
/etc/ssh/ssh_known_hosts file
   controlling distribution, 331
   description, 332
   override, 334
   secure distribution, 331
/etc/ssh/sshhd_config file
   description, 332
   keywords, 326-331
/etc/ssh/sshrpc file, description, 333
/etc/syslog.conf file
   auditing and, 527, 559
   executable stack messages and, 128
   failed logins and, 62-63
   PAM and, 294
/etc/system file, 559
   event, description, 506
   event modifier field flags (header token), 577
   exec_args audit token
      argv policy and, 575
      format, 575
   exec_attr database
      description, 223-224
      summary, 219
   exec audit class, 565

Index 617
exec_env audit token, format, 575
executable stacks
  disabling logging messages, 142
  logging messages, 128
  protecting against, 128 141-142
execute permissions, symbolic mode, 124
execution log (ASET), 147
exit audit token, format, 575
EXTERNAL security mechanism plug-in, SASL and, 300

F
-F option
  Kerberized commands, 478 480-481
  setfacl command, 137
  st clean script, 95
-f option
  deallocate command, 91
  Kerberized commands, 479 480-481
failed login attempts
  loginlog file, 61
  syslog.conf file, 62-63
failure
  audit class prefix, 566
  turning off audit classes for, 566
FallBackToRsh keyword, ssh_config file, 327
fd_clean script, description, 94
file_attr_acc audit class, 565
file_attr_mod audit class, 565
file audit token, format, 576
file_close audit class, 565
file_creation audit class, 565
file_deletion audit class, 565
file permission modes
  absolute mode, 124
  symbolic mode, 124
FILE privileges, 179
file_read audit class, 565
file systems
  NFS, 277
  security
    authentication and NFS, 277
    TMPFS file system, 123
  sharing files, 51
  TMPFS, 123
file vnode audit token, 574
file_write audit class, 565
files
  ACL entries
    adding or modifying, 137-138
    checking, 135
    deleting, 127 138
    displaying, 127 139-140
    setting, 135-137
    valid entries, 126-127
  ASET checks, 145 146
  BART manifests, 114-115
  changing ACL, 137-138
  changing group ownership, 131
  changing ownership, 119,130-131
  changing special file permissions, 133-134
  computing a digest, 258-259
  computing digests of, 258-259
  computing MAC of, 259-260
  copying ACL entries, 137
  copying with Solaris Secure Shell, 319-320
  decrypting, 261
  deleting ACL, 138
  determining if has ACL, 135
  digest of, 258-259
  displaying ACL entries, 139-140
  displaying file information, 129-130
  displaying hidden files, 129
  displaying information about, 119
  encrypting, 256 261-263
  file types, 120
  finding files with setuid permissions, 140
  for administering Solaris Secure Shell, 332
  hashing, 256
  kdc.conf, 491
  Kerberos, 485-486
  manifests (BART), 114-115
  mounting with DH authentication, 286
  ownership
    and setgid permission, 122
    and setuid permission, 122
  permissions
    absolute mode, 123,132-133
    changing, 120 123-125,132
    defaults, 123
    description, 120-121
files, permissions (Continued)
  setgid, 122
  setuid, 122
  sticky bit, 122-123
  symbolic mode, 123, 124, 132
  umask value, 123
privileges relating to, 179
protecting with ACLs, 134-140
protecting with UNIX permissions, 129-134
public objects, 506
security
  access restriction, 48
  ACL, 50
  changing ownership, 130-131
  changing permissions, 123-125 132
  directory permissions, 120-121
  displaying file information, 119 129-130
  encryption, 50, 256
  file permissions, 120-121
  file types, 120
  special file permissions, 125
  umask default, 123
  UNIX permissions, 119-125
  user classes, 120
setting ACL, 135-137
sharing with DH authentication, 286-287
special files, 121-123
symbols of file type, 120
syslog.conf file, 559
verifying integrity with digest, 258-259
  with privilege information, 244-245
find command, finding files with setuid permissions, 140
firewall.rpt file, 147, 150
firewall systems
  ASET setup, 147
  connecting from outside, 321
  outside connections with Solaris Secure Shell
    from command line, 321
    from configuration file, 320-321
  packet smashing, 55
  packet transfers, 55
  secure host connections, 320
  security, 54
  trusted hosts, 54
  flags line
    audit_control file, 559
    plugin line and, 527
    process preselection mask, 567
  forced cleanup, st_clean script, 95
  format of audit records, bsmrecord command, 543
forwardable tickets
  definition, 490
  description, 340
  example, 470
  with -F option, 479, 480-481
  with -f option, 478, 480-481
ForwardAgent keyword, Solaris Secure Shell forwarded authentication, 327
ForwardX11 keyword, Solaris Secure Shell port forwarding, 327
FQDN (Fully Qualified Domain Name), in Kerberos, 355-356
ftp command
  description, 487
  Kerberos and, 477-480
  setting protection level in, 479
ftp daemon, Kerberos and, 487-488

G
GatewayPorts keyword, Solaris Secure Shell, 327
gateways, See firewall systems
generating
  keys for Solaris Secure Shell, 312-314
  secret keys, 279
  Solaris Secure Shell keys, 312-314
  symmetric key for encryption, 256-258
Generic Security Service API, See GSS-API
getdevpolicy command, See GSS-API
getfacl command
  -a option, 139
  -d option, 139
  description, 127
  displaying ACL entries, 139-140
  examples, 139-140
  verifying ACL entries, 136
getting
  access to a specific service, 496-497
  credential for a server, 495-496
getting (Continued)
  credential for a TGS, 494-495

\texttt{gkadmin} command
  \textit{See also} SEAM Administration Tool
description, 487
\texttt{.gkadmin} file
description, 485
SEAM Administration Tool and, 433
\texttt{GlobalKnownHostsFile} keyword, \texttt{ssh\_config} file, 327
\texttt{GlobalKnownHostsFile2} keyword, See
\texttt{GlobalKnownHostsFile} keyword
granting access to your account, 476-477
group ACL entries
default entries for directories, 127
description, 126-127
setting, 135-137
group audit policy
and groups token, 518, 576
description, 518
group audit token, replaced by groups token, 576
group ID numbers (GIDs), special logins and, 42
groups, changing file ownership, 131
groups audit token, 576
GSS-API
authentication in Solaris Secure Shell, 304
credentials in secure RPC, 282-283
credentials in Solaris Secure Shell, 324
Kerberos and, 340 352
\texttt{gssapi.so.1} plug-in, SASL and, 300
GSSAPIAuthentication keyword, Solaris Secure Shell, 327
\texttt{GSSAPI\_Delegate\_Credentials} keyword, Solaris Secure Shell, 327
\texttt{GSSAPI\_Key\_Exchange} keyword, Solaris Secure Shell, 327
\texttt{GSSAPI\_Store\_Delegated\_Credentials} keyword,\texttt{ ssh\_config} file, 327
\texttt{gsscred} command, description, 487
\texttt{gsscred} table, using, 499
gssd daemon, Kerberos and, 487-488

\textbf{H}
-h option, \texttt{bsmrecord} command, 543
hard disk, space requirements for auditing, 520
hard string, \texttt{audit\_warn} script, 562
hardware
listing attached hardware accelerators, 272
protecting, 38, 72-73
requiring password for access, 72-73
hardware providers
disabling cryptographic mechanisms, 272-274
enabling mechanisms and features on, 273
listing, 272
loading, 272
hashing, files, 256
header audit token
event-modifier field flags, 577
format, 577
order in audit record, 577
help
SEAM Administration Tool, 433
URL for online, 359
Help Contents, SEAM Administration Tool, 433
hierarchical realms
configuring, 372-373
in Kerberos, 345-346 354-355
high ASET security level, 144
\texttt{hmac\_md5} algorithm, \texttt{ssh\_config} file, 328
\texttt{hmac\_sha1} encryption algorithm, \texttt{ssh\_config} file, 328
host-based authentication
configuring in Solaris Secure Shell, 307-309
description, 304
Host keyword
\texttt{ssh\_config} file, 327 330
host names, mapping onto realms, 355
host principal, creating, 367
host principal, DNS and, 355
HostbasedAuthentication keyword, Solaris Secure Shell, 327
HostbasedUsesNames\_From\_Packet\_Only keyword,
\texttt{sshd\_config} file, 327
HostKey keyword, \texttt{sshd\_config} file, 327
HostKeyAlgorithms keyword, \texttt{ssh\_config} file, 328
HostKeyAlias keyword, \texttt{ssh\_config} file, 328
hosts
disabling Kerberos service on, 466-467
Solaris Secure Shell hosts, 304
trusted hosts, 54
\texttt{hosts.equiv} file, description, 333
-i option
  bart create command, 101, 107
  encrypt command, 261
  st_clean script, 95
-I option
  bart create command, 101
  st_clean script, 95
identity files (Solaris Secure Shell), naming conventions, 332
IdentityFile keyword, ssh_config file, 328
IDs
  audit
    mechanism, 567
    overview, 503-504
    audit session, 567
    mapping UNIX to Kerberos principals, 499
IgnoreHosts keyword, sshd_config file, 328
IgnoreUserKnownHosts keyword, sshd_config file, 328
in_addr audit token, format, 577
inheritable privilege set, 181
initial ticket, definition, 490
install subcommand, cryptoadm command, 267
installing
  password encryption module, 68-69
  providers in cryptographic framework, 254
instance, in principal names, 344-345
integrity
  Kerberos and, 339
  security service, 346
interactively running ASET, 160-161
INTERNAL plug-in, SASL and, 300
Internet firewall setup, 54
Internet-related tokens
  in_addr token, 577
  ip token, 578
  iport token, 579
  socket token, 583
invalid ticket, definition, 490
ioctl audit class, 565
ioctl() system calls, 565
AUDIO_SETINFO(), 94
IP addresses, Solaris Secure Shell checking, 326
ip audit token, format, 578
IP MIB-II, getting information from /dev/arp, 78-79
ipc audit token, 578
  format, 578
ipc_perm audit token, format, 579
IPC privileges, 179
ipc type field values (ipc token), 578
iport audit token, format, 579
item size field, arbitrary token, 572

J
JASS toolkit, pointer to, 48

K
-k option
  encrypt command, 261
  Kerberized commands, 479
  mac command, 260
-K option
  Kerberized commands, 479
  usermod command, 234
.k5.REALM file, description, 486
.k5.login file
  description, 476-477, 485
  rather than revealing password, 476
kadm5.acl file
  description, 485
  format of entries, 448
  master KDC entry, 365, 398
  new principals and, 442 444
kadm5.keytab file
  description, 461, 485
kadmin command
  creating host principal, 367
  description, 487
  ktadd command, 463-464
  ktremove command, 464
  removing principals from keytab with, 464-465
  SEAM Administration Tool and, 431
kadmin.local command
  adding administration principals, 365
  automating creation of principals, 437
  creating keytab file, 366
  description, 487

Index
Kerberos (Continued)
Administration Tool
See SEAM Administration Tool commands, 477-483, 487
components of, 347-348
configuration decisions, 353-359
configuring KDC servers, 363-372
daemons, 487-488
dfstab file option, 381
enabling Kerberized applications only, 414
error messages, 417-428
examples of using Kerberized commands, 481-483
files, 485-486
gaining access to server, 494-497
granting access to your account, 476-477
Kerberos V5 protocol, 339
online help, 359
options to Kerberized commands, 478
overview
authentication system, 340-346, 493-494
Kerberized commands, 477-480
password management, 475-499
planning for, 353-359
realms
See realms (Kerberos)
reference, 485-499
remote applications, 344
table of network command options, 479
terminology, 488-493
troubleshooting, 429
using, 469-483
Kerberos authentication and Secure RPC, 278
dfstab file option, 381
Kerberos commands, 477-483
enabling only Kerberized, 414
examples, 481-483
kern.notice entry, syslog.conf file, 128
kernel providers, listing, 264
Key Distribution Center, See KDC
KEYBOARD_ABORT system variable, 73
KeyLogin command
use, 279
verifying DH authentication setup, 283
KeyRegenerationInterval keyword, sshd_config file, 328
keys
creating DH key for NIS user, 285-286
creating for Solaris Secure Shell, 312-314
definition in Kerberos, 489
generating for Solaris Secure Shell, 312-314
generating symmetric key, 256-258
service key, 461-467
session keys
Kerberos authentication and, 493-494
using for MAC, 260
keyserv daemon, 282
keyserver
description, 279
starting, 282
keytab file
adding master KDC’s host principal to, 368
adding service principal to, 461, 463-464
administering, 461-467
administering with ktutil command, 462
creating, 366
disabling a host’s service with delete_entry
command, 467
read into keytab buffer with read_kt command, 465, 466
removing principals with ktremove command, 464
removing service principal from, 464-465
viewing contents with ktutil command, 464, 465-466
viewing keylist buffer with list command, 465, 466
keytab option, SASL and, 301
keywords
See also specific keyword
attribute in BART, 115
command-line overrides in Solaris Secure Shell, 335
Solaris Secure Shell, 326-331
kinit command
description, 487
example, 470
-F option, 470
ticket lifetime, 491
klist command
description, 487
time, 470-472
-L option, 470-472
known_hosts file
controlling distribution, 331
description, 332
Korn shell, privileged version, 177
kpasswd command
description, 487
time message, 474
time example, 475
time passwd command and, 474
time prop command, description, 487
time kpropd.acl file, description, 486
time kpropd daemon, Kerberos and, 488
time kproplog command, description, 487
krb5.conf file
description, 486
domain_realm section, 355
editing, 363
ports definition, 356
krb5.keytab file, description, 486
krb5cc_uid file, description, 486
krb5kdc daemon
Kerberos and, 488
master KDC and, 488
starting, 372, 410
ksh command, privileged version, 177
ktadd command
adding service principal, 461, 463-464
syntax, 463
ktkt_warn daemon
Kerberos and, 487-488, 488
ktremove command, 464
ktutil command
administering keytab file, 462
delete_entry command, 467
description, 487
list command, 465, 466
read_kt command, 465, 466
viewing list of principals, 464, 465-466
-l option
digest command, 258
encrypt command, 256
mac command, 259
praudit command, 557
-L option, ssh command, 317-319
LDAP name service
  passwords, 40
  specifying password algorithm, 67-68
least privilege, principle of, 179
libraries, user-level providers, 264
lifetime of ticket, in Kerberos, 491-492
limit privilege set, 181
limiting, use of privileges by user or role, 235-236
limitpriv keyword, user_attr database, 244
list command, 465, 466
list_devices command
  authorizations for, 91
  authorizations required, 227
description, 90
list privilege, SEAM Administration Tool and, 460
LoginAddress keyword, sshd_config file, 328
listing
  available providers in cryptographic framework, 264-266
cryptographic framework providers, 272
device policy, 76-77
hardware providers, 272
providers in the cryptographic framework, 264-266
roles you can assume, 200: 226
users with no passwords, 59-60
LocalForward keyword, sshd_config file, 328
log files
  audit records, 508-550
BART
    programmatic output, 116-117
    verbose output, 116-117
configuring for auditing service, 526-528
examining audit records, 555
execution log (ASET), 147
failed login attempts, 62-63
monitoring su command, 70
space for audit records, 554
syslog audit records, 559
log_level option, SASL and, 301
logadm command, archiving textual audit files, 551
logging in (Continued)
  root login
    account, 42
    restricting to console, 70-71
    tracking, 46
security
  access control on devices, 43
  access restrictions, 39
  saving failed attempts, 61
  system access control, 39
  tracking root login, 46
system logins, 42
task map, 58
users' basic privilege set, 182
with Solaris Secure Shell, 315-316
login environment variables, Solaris Secure Shell and, 330-331
login file
  login default settings, 62
  login file, path variable entry, 47
login file
  restricting remote root access, 70-71
login_logout auditclass, 566
LoginGraceTime keyword, sshd_config file, 328
loginlog file, saving failed login attempts, 61
logins command
  displaying user's login status, 58-59, 59
  displaying users with no passwords, 59
  syntax, 58
LoginLevel keyword, Solaris Secure Shell, 328
LookupClientHostname keyword, sshd_config file, 328
low ASET security level, 144

M
  -m option
    cryptoadm command, 268, 269
    Kerberized commands, 479
  -M option, audit reduce command, 546
mac command
  description, 253
  syntax, 259
machine security, See system security
MACS keyword, Solaris Secure Shell, 328
mail, using with Solaris Secure Shell, 318-319
makedbm command, description 225
managing
See also administering
audit files, 545-546 551
audit records task map, 542-543
audit trail overflow, 551
auditing, 523
auditing in zones, 564
device allocation task map, 79-80
devices, 79-80
file permissions, 128-129
passwords with Kerberos, 473-477
privileges task map, 230
RBAC task map, 203-204
manifests
See also bart create
customizing, 103-106
file format, 114-115
test, 99
mapping
host names onto realms (Kerberos), 355
UIDs to Kerberos principals, 499
mapping GSS credentials, 357
mappings, events to classes (auditing), 508
mask (auditing)
description of process preselection, 567
system-wide process preselection, 559
mask ACL entries
default entries for directories, 127
description, 126-127
setting, 135-137
master files (ASET), 145 151
master KDC
configuring, 363-368
definition, 488
slave KDCs and, 345-346 363
swapping with slave KDC, 395-400
max_life value, description, 491
max_renewable_life value, description, 492
MaxAuthTries keyword, sshd_config file, 328
MaxAuthTriesLog keyword, sshd_config file, 328
MaxStartups keyword, sshd_config file, 328
MD5 encryption algorithm
kernel provider, 264
policy.conf file, 66-67
mech_dh mechanism
GSS-API credentials, 324
secure RPC, 282-283
mech_krb mechanism, GSS-API credentials, 324
mech_list option, SASL and, 301
mechanism, definition in cryptographic framework, 251
mechanisms
disabling all on hardware provider, 272-274
enabling some on hardware provider, 273
medium ASET security level, 144
merging, binary audit records, 545-546
message authentication code (MAC), computing for file, 259-260
messages file, executable stack messages, 128
metaslot
administering, 252
definition in cryptographic framework, 251
microphone
allocating, 85
deallocating, 88
minfree line
audit_control file, 560
audit_warn condition, 562
minus sign (-)
audit class prefix, 566
entry in su log file, 70
file permissions symbol, 124
symbol of file type, 120
mode, definition in cryptographic framework, 251
modifying
policies (Kerberos), 455-456
principal’s password (Kerberos), 445
principals (Kerberos), 444-445
role assignment to a user, 193
roles (RBAC), 204-206
users (RBAC), 208-210
modules
PAM, 297
password encryption, 40
types in PAM, 295
monitoring
audit trail in real time, 521
failed logins, 61
su command attempts, 46 70
superuser access attempts, 70-71
superuser task map, 69
monitoring (Continued)
  system usage, 49
  use of privileged commands, 197
mount command, with security attributes, 80
mounting
  allocated CD-ROM, 87
  allocated devices, 86-87
  allocated diskette, 86-87
  audit directories, 568
  files with DIF authentication, 286
mt command, tape device cleanup and, 94

N
-n option
  audit command, 554
  bart create command, 101
naflags line
  audit_control file, 560
  plugin line and, 527
name services
  See also individual name services
  scope and RBAC, 177
names
  audit classes, 565
  audit files, 569
  device names
    device_maps file, 92, 93
naming conventions
  audit directories, 526, 560
  audit files, 568
  devices, 81
  RBAC authorizations, 218
  Solaris Secure Shell identity files, 332
NET privileges, 179
network, privileges relating to, 179
network audit class, 566
network security
  authentication, 52-53
  authorizations, 52-53
  controlling access, 51-55
  firewall systems
    need for, 54
    packet smashing, 55
    trusted hosts, 54
  network security (Continued)
    overview, 51
    reporting problems, 55
Network Security (RBAC), creating role, 192
Network Time Protocol, See NTP
never-audit classes, audit_user database, 561
new features
  auditing enhancements, 510-511
  BART, 97-117
commands
  bart compare, 99
  bart create, 98
cryptoadm, 264
decrypt, 261
digest, 258-259
encrypt, 261-263
getdevpolicy, 76-77
kcfd, 274
cldent, 348
kpropd, 348
mac, 259-260
ppriv, 230-232
praudit -x, 549
ssh-keyscan, 334
ssh-keysign, 334
cryptographic framework, 249-254
device policy, 45
Kerberos enhancements, 348-350
metaslot, 249
PAM enhancements, 291
privileges, 178-185
process rights management, 178-185
SASL, 299
Solaris Cryptographic Framework, 249-254
Solaris Secure Shell enhancements, 306
strong password encryption, 40
system security enhancements, 37-38
newkey command
  creating key for NIS user, 285-286
generating keys, 279
NFS file systems
  ASET and, 155
  authentication, 277
  providing client-server security, 279-281
  secure access with AUTH_DH, 286
NFS servers, configuring for Kerberos, 377-378
Index

NIS+ name service
  adding authenticated user, 284
  ASET checks, 154
  authentication, 277
  cred database, 284
  cred table, 279
  passwords, 40
  specifying password algorithm, 67
NIS name service
  authentication, 277
  passwords, 40
  specifying password algorithm, 67
nisaddcred command
  adding client credential, 283
  generating keys, 279
no_class audit class, 565
nobody user, 51
noexec_user_stack_log variable, 128, 142
noexec_user_stack variable, 128, 141
NoHostAuthenticationForLocalHost keyword, ssh_config file, 328
nologin file, description, 332
non_attrib audit class, 565
nonattributable classes, 560
nonhierarchical realms, in Kerberos, 345-346
nscd (name service cache daemon)
  starting with svcadm command, 191
  use, 225
nsswitch.conf file, login access restrictions, 39
NTP
  Kerberos and, 358
  master KDC and, 368
  slave KDC and, 372, 410
null audit class, 565
NumberOfPasswordPrompts keyword, ssh_config file, 328

O
- o option, encrypt command, 261
- O option, auditreduce command, 545-546
object reuse requirements
  device-clean scripts
  tape drives, 93
  writing new scripts, 94-95

P
- p option
  aset command, 162
  bart create, 107
  bsmrecord command, 543-544
  cryptoadm command, 268, 270
  logins command, 59
packages, Solaris Secure Shell, 331
packet transfers
  firewall security, 54
  packet smashing, 55
Index

PAM
  adding a module, 293
  configuration file, 294, 297
  control flags, 296
  /etc/syslog.conf file, 294
  Kerberos and, 348, 351
  module types, 295
  modules, 297
  overview, 289
  planning, 292
  service names, 295
  stacking, 290
  task map, 292
pam_*.so.1 files, description, 297
pam.conf file
  control flags, 296
  description, 294
  examples, 297
  Kerberos and, 486
  service names, 295
pam_roles command, description, 225
PAMAuthenticationViaKBDInt keyword, sshd_config file, 328
panels, table of SEAM Administration Tool, 457-460
passphrases
  changing for Solaris Secure Shell, 314
  encrypt command, 261
  example, 315
  mac command, 259
  storing safely, 261
  using for MAC, 260
  using in Solaris Secure Shell, 314, 316-317
PASSREQ in Solaris Secure Shell, 330
passwd command
  and kpasswd command, 473
  and name services, 40
passwd file
  and /etc/d_passwd file, 43
ASET checks, 146
password authentication, Solaris Secure Shell, 304
PasswordAuthentication keyword, Solaris Secure Shell, 328
passwords (Continued)
  changing with passwd command, 473
  creating for dial-up, 63-65
  dial-up passwords
    disabling temporarily, 65
    /etc/d_passwd file, 43
  disabling dial-up temporarily, 65
  displaying users with no passwords, 59
  eliminating in Solaris Secure Shell, 316-317, 317
  encryption algorithms, 40
  finding users with no passwords, 59-60
  granting access without revealing, 476-477
  hardware access and, 72-73
  installing third-party encryption module, 68-69
  LDAP, 40
    specifying new password algorithm, 67-68
    local, 40
  login security, 39
  managing, 473-477
  modifying a principal's password, 445
  NIS, 40
    specifying new password algorithm, 67
    NIS+, 40
    specifying new password algorithm, 67
    policies and, 474
  PROM security mode, 38, 72-73
  requiring for hardware access, 72-73
  secret-key decryption, 279
  specifying algorithm, 66-67
    in name services, 67
    locally, 65
  suggestions on choosing, 473
  system logins, 39, 42
  task map, 58
  UNIX and Kerberos, 473-477
  using Blowfish encryption algorithm for, 66-67
  using MD5 encryption algorithm for, 66-67
  using new algorithm, 66
path_attr audit token, 511, 580
path audit policy, description, 518
path audit token, format, 580
PATH environment variable
  and security, 47
  setting, 47
PATH in Solaris Secure Shell, 331
PERIODIC_SCHEDULE variable (ASET), 153, 157
permissions
ACLs and, 50, 125-127
ASET handling of, 144, 145
changing file permissions
absolute mode, 123, 132-133
chmod command, 120
symbolic mode, 123, 124, 132
defaults, 123
directory permissions, 120-121
file permissions
absolute mode, 123, 123-123
changing, 123-125, 132
description, 120-121
special permissions, 123, 125
symbolic mode, 123, 124, 132
finding files with setuid permissions, 140
setgid permissions
absolute mode, 125, 134
description, 122
symbolic mode, 124
setuid permissions
absolute mode, 125, 134
description, 122
security risks, 122
symbolic mode, 124
special file permissions, 121-123, 123, 125
sticky bit, 122-123
tune files (ASET), 151, 154
umask value, 123
user classes and, 120
PermitEmptyPasswords keyword, sshd_config file, 328
PermitRootLogin keyword, sshd_config file, 329
permitted privilege set, 181
PermitUserEnvironment keyword, sshd_config file, 329
perzone audit policy
description, 519
setting, 538
using, 564
pfcsh command, description, 177
pfexec command, description, 225
pfksh command, description, 177
pfsh command, description, 177
physical security, description, 38
PKCS #11 library
adding as provider, 267-268
in Solaris Cryptographic Framework, 250
pkcs11_kernel.so user-level provider, 264
pkcs11_softtoken.so user-level provider, 264
pkgadd command
installing third-party providers, 267
installing third-party software, 68
plain.so.1 plug-in, SASL and, 300
planning
auditing, 514-517
auditing in zones, 514-515
auditing task map, 513
Kerberos
client and service principal names, 355-356
clock synchronization, 358
configuration decisions, 353-359
database propagation, 357
number of realms, 354
ports, 356
realm hierarchy, 354-355
realm names, 354
realms, 354-355
slave KDCs, 356
PAM, 292
RBAC, 189-190
plug-ins, SASL and, 300
pluggable authentication module, See PAM
plugin line
audit_control file, 560
flags line and, 527
plugin_list option, SASL and, 301
plugins
in auditing service, 527
in cryptographic framework, 250
plus sign (+)
ACL entry, 135
audit class prefix, 566
entry in sulog file, 70
file permissions symbol, 124
policies
administering, 431-467
creating (Kerberos), 442
creating new (Kerberos), 453-454
deleting, 456
for auditing, 517-519
modifying, 455-456
on devices, 76-77
overview, 32-33
policies (Continued)
  passwords and, 474
  SEAM Administration Tool panels for, 457-460
  specifying password algorithm, 65
  task map for administering, 449
  viewing attributes, 451-453
  viewing list of, 449-451
policy
  definition in cryptographic framework, 251
  definition in Solaris OS, 32-33
policy.conf file
  adding password encryption module, 68-69
  Basic Solaris User rights profile, 216
  description, 224-225, 226
  keywords
    for password algorithms, 41
    for privileges, 225, 224
    for RBAC authorizations, 224
    for rights profiles, 225
  specifying encryption algorithms in, 66-67
  specifying password algorithm
    in name services, 67
  specifying password algorithms, 66-67
port forwarding
  configuring in Solaris Secure Shell, 310-311
  Solaris Secure Shell, 318-319, 319
Port keyword, Solaris Secure Shell, 329
ports, for Kerberos KDC, 356
postdated ticket
  definition, 490
  description, 340
postsigterm string, audit_warn script, 563
pound sign (#)
  device_allocate file, 93
  device_maps file, 92
ppriv command
  for debugging, 232
  listing privileges, 230
praudit command
  converting audit records to readable format, 550, 556
  DTD for -x option, 557
  options, 556
  output formats, 556
  piping auditreduce output to, 549
  use in a script, 557-558
  viewing audit records, 548-550
praudit command (Continued)
  with no options, 557
  XML format, 550
PreferredAuthentications keyword, ssh_config file, 329
prefixes for audit classes, 566
preselecting, audit classes, 524-526
preselection in auditing, auditing, 506
preselection mask (auditing)
  description, 567
  reducing storage costs, 554
  system-wide, 559
preventing
  access to system hardware, 72
  audit trail overflow, 551
  executables from compromising security, 128
  kernel software provider use, 269-271
  use of hardware mechanism, 272-274
primary, in principal names, 344-345
Primary Administrator (RBAC)
  assuming role, 201
  recommended role, 170
  rights profile contents, 214
primary audit directory, 560
principal
  adding administration, 365
  adding service principal to keytab, 461, 463-464
  administering, 431-467
  automating creation of, 437-438
  creating, 442-444
  creating clntconfig, 367
  creating host, 367
  deleting, 445-446
  duplicating, 444
  Kerberos, 344-345
  modifying, 444-445
  principal name, 344-345
  removing from keytab file, 464
  removing service principal from keytab, 464-465
  SEAM Administration Tool panels for, 457-460
  service principal, 344
  setting up defaults, 446-447
  task map for administering, 437
  user ID comparison, 379
  user principal, 344
  viewing attributes, 440-442
principal (Continued)
  viewing list of, 438-440
  viewing sublist of principals, 439
principal file, description, 486
principal.kadm5 file, description, 486
principal.kadm5.lock file, description, 486
principal.ok file, description, 486
principal.ulo file, description, 486
principle of least privilege, 179
print format field, arbitrary token, 573
Printer Management rights profile, 215
printing, audit log, 549
PrintMotd keyword, sshd_config file, 329
priv.debug entry, syslog.conf file, 245
PRIV_DEFAULT keyword
  policy.conf file, 225-244
PRIV_LIMIT keyword
  policy.conf file, 225-244
privacy
  availability, 480
  Kerberos and, 339
  security service, 346
private keys
  See also secret keys
    definition in Kerberos, 489
    Solaris Secure Shell identity files, 332
private protection level, 480
privilege audit token, 511, 581
privilege checking, in applications, 175
privilege sets
  adding privileges to, 183
  basic, 182
  effective, 181
  inheritable, 181
  limit, 181
  listing, 182
  permitted, 181
  removing privileges from, 184
privileged application
  authorization checking, 175
  description, 172
  ID checking, 174
  privilege checking, 175
privileged ports, alternative to Secure RPC, 53
privileges (Continued)
  administering, 230
  assigning to a command, 183
  assigning to a script, 184
  assigning to a user, 183
  assigning to user or role, 234-235
  auditing and, 245
  categories, 179
  commands, 243
  compared to superuser model, 178-185
  debugging, 185, 232
  description, 171, 179
  determining directly assigned ones, 238-239
  devices and, 184
  differences from superuser model, 180
  effects on SEAM Administration Tool, 460
  escalation, 245
  executing commands with privilege, 183
  files, 244-245
  finding missing, 232-233
  how to use, 237
  implemented in sets, 181
  inherited by processes, 182
  limiting use by user or role, 235-236
  listing on a process, 230-232
  processes with assigned privileges, 182
  programs aware of privileges, 182
  protecting kernel processes, 178
  removing from a user, 184
  removing from basic set, 235
  removing from limit set, 236
  task map, 229
  using in shell script, 236-237
privileges file, description, 180
PROC privileges, 179
process audit characteristics
  audit ID, 567
  audit session ID, 567
  process preselection mask, 567
  terminal ID, 568
process audit class, 565
process audit token, format, 581
process modify audit class, 565
process preselection mask, description, 567
process privileges, 179
process rights management, See privileges
| process start audit class, 565 |
| processing time costs, of auditing service, 520 |
| prof_attr database |
| description, 222-223 |
| summary, 219 |
| .profile file, path variable entry, 47 |
| profile shells, description, 177 |
| profiles, see rights profiles |
| profiles command, description, 226 |
| PROFS_GRANTED keyword, policy.conf file, 225 |
| programs |
| checking for RBAC authorizations, 211 |
| privilege-aware, 181, 182 |
| PROM security mode, 72-73 |
| propagation |
| KDC database, 357 |
| Kerberos database, 400-401 |
| protecting |
| BIOS, pointer to, 72-73 |
| files with cryptographic framework, 256 |
| PROM, 72-73 |
| system from risky programs, 140-142 |
| protecting files |
| task map, 128 |
| user procedures, 129-134 |
| with ACLs, 125-127, 134-140 |
| with ACLs task map, 134 |
| with UNIX permissions, 119-125, 129-134 |
| with UNIX permissions task map, 129 |
| protection level |
| clear, 479 |
| private, 480 |
| safe, 480 |
| setting in ftp, 479 |
| Protocol keyword, ssh_config file, 329 |
| providers (Continued) |
| preventing use of kernel software provider, 269-271 |
| registering, 254 |
| restoring use of kernel software provider, 270 |
| signing, 254 |
| proxiable ticket, definition, 490 |
| proxy ticket, definition, 490 |
| ProxyCommand keyword, ssh_config file, 329 |
| pseudo-tty, use in Solaris Secure Shell, 325 |
| PublicKeyAuthentication keyword, Solaris Secure Shell, 329 |
| public audit policy |
| description, 519 |
| read-only events, 519 |
| public directories |
| auditing, 506 |
| sticky bit and, 123 |
| public key authentication, Solaris Secure Shell, 304 |
| public key cryptography |
| AUTH_DH client-server session, 279-281 |
| changing public keys and secret keys, 279 |
| common keys |
| calculation, 280 |
| database of public keys, 279 |
| generating keys |
| conversation keys, 279 |
| public and secret, 279 |
| secret keys, 279 |
| public keys |
| changing passphrase, 314 |
| DH authentication and, 278-281 |
| generating public-private key pair, 312-314 |
| Solaris Secure Shell identity files, 332 |
| public objects, auditing, 506 |
| publickey map, DH authentication, 278-281 |
| pwcheck_method option, SASL and, 301 |

Q

question mark (?), in ASET tune files, 159

quoting syntax in BART, 116
Index

R

- r option
  bart create, 107
  passwd command, 40
  praudit command, 557
-R option
  bart create, 101, 107
  ssh command, 317-319
random numbers, od command, 256-258
raw praudit output format, 557
RBAC
  adding custom roles, 195-196
  adding new rights profile, 207
  adding roles, 191-193
  adding roles from command line, 193-196
  administration commands, 225-226
  audit profiles, 563
  auditing roles, 197
  authorization database, 221-222
  authorizations, 174
  basic concepts, 171-174
changing user properties
  from command line, 210
checking scripts or programs for authorizations, 211
commands for managing, 225-226
compared to superuser model, 169-171
configuring, 188-199
database relationships, 219-220
databases, 219-225
editing rights profiles, 206-208
  elements, 171-174
  modifying roles, 204-206
  modifying users, 208-210
name services and, 220
planning, 189-190
profile shells, 177
rights profile database, 222-223
rights profiles, 175-176
securing scripts, 211
using privileged applications, 202-203
RC4, See ARCFOUR kernel provider
rcp command
  description, 487
  Kerberos and, 477-480
read kt command, 465-466
read permissions, symbolic mode, 124
readable audit record format
  converting audit records to, 550, 556
realms (Kerberos)
  configuration decisions, 354-355
  configuring cross-realm authentication, 372-374
  contents of, 346
  direct, 373-374
  hierarchical, 372-373
  hierarchical or nonhierarchical, 345-346
  hierarchy, 354-355
  in principal names, 344-345
  mapping host names onto, 355
  names, 354
  number of, 354
  requesting tickets for specific, 479
  servers and, 345-346
reauth_timeout option, SASL and, 301
redirecting arrow (>), preventing redirection, 47
reducing
  audit files, 545-546, 555
  storage-space requirements for audit files, 521
refreshing, cryptographic services, 274
registering providers, cryptographic framework, 254
rem_drv command, description, 89
remote logins
  authentication, 52-53
  authorization, 52-53
  preventing superuser from, 70-71
  security and, 281
RemoteForward keyword, ssh_config file, 329
removing
  ACL entries, 138
  cryptographic providers, 269
  device policy, 78
  policy from device, 78
  principals with kt remove command, 464
  privileges from basic set, 235
  privileges from limit set, 236
  service principal from keytab file, 464-465
  software providers
    permanently, 271
    temporarily, 270
  renewable ticket, definition, 490
  replacing, superuser with roles, 189-190
  replayed transactions, 281
reporting tool, See bart compare
rights profiles (Continued)
  using the System Administrator profile, 72
  viewing contents, 217
Rights tool, description, 206-208
rlogin command
description, 487
  Kerberos and, 477-480
rlogin daemon, Kerberos and, 487-488
role-based access control, See RBAC
roleadd command
description, 226
  using, 194
roledel command, description, 226
rolemod command
  changing properties of role, 204
  description, 226
roles
  adding custom roles, 195-196
  adding for particular profiles, 191-193
  adding from command line, 193-196
  assigning privileges to, 234-235
  assigning with usermod command, 196-197
  assuming, 200-202 202-203
  assuming after login, 176
  assuming in a terminal window, 177 200-202
  assuming in Solaris Management Console, 202-203
  assuming Primary Administrator role, 201
  assuming root role, 201
  assuming System Administrator role, 201-202
  auditing, 197
  changing properties of, 204-206
creating
  Crypto Management role, 197
  Custom Operator role, 195-196
  Device Security role, 192
  DHCP Management role, 192-193
  for particular profiles, 191-193
  Network Security role, 192
  on command line, 193-196
  Operator role, 192
  role with limited scope, 192-193
  root role, 198-199
  security-related roles, 192
  System Administrator role, 191-192
  description, 176
determining directly assigned privileges, 238-239
roles (Continued)
determining role's privileged commands, 240-242
listing local roles, 200, 226
making root user into role, 198-199
modifying, 204-206
modifying assignment to a user, 193
recommended roles, 169
summary, 172
troubleshooting, 193
use in RBAC, 169
using an assigned role, 200-202, 202-203
using to access the hardware, 72-73
roles command
description, 226
using, 200
root principal, adding to host's keytab, 462
root role (RBAC), assuming role, 201
root user
changing to root role, 198-199
displaying access attempts on console, 70-71
login account
description, 42
monitoring su command attempts, 46, 70
replacing in RBAC, 176
restricting access, 51
restricting remote access, 70-71
tracking logins, 46
RPCSEC_GSSAPI, Kerberos and, 352
RSA kernel provider, 264
RSA Authentication keyword, Solaris Secure Shell, 329
rsh command
description, 487
Kerberos and, 477-480
rsh command (restricted shell), 47
rshd daemon, Kerberos and, 487-488
rstchown system variable, 131
rules file (BART), 99-100
rules file attributes, See keywords
rules file format (BART), 115-116
rules file specification language, See quoting syntax
Running ASET task map, 160-163

S
-s option
audit command, 554
praudit command, 557
-S option, st_clean script, 95
safe protection level, 480
SASL
environment variable, 300
options, 301
overview, 299
plug-ins, 300
saslauthd_path option, SASL and, 301
saving, failed login attempts, 61
scope (RBAC), description, 177
scp command
copying files with, 319-320
description, 335
scripts
audit_startup script, 561
audit_warn script, 562
bsmconv effect, 559
bsmconv for device allocation, 80
bsmconv script, 563
bsmconv to enable auditing, 539-540
checking for RBAC authorizations, 211
device-clean scripts
See also device-clean scripts
for cleaning devices, 93-95
monitoring audit files example, 521
processing praudit output, 557-558
running with privileges, 184
securing, 211
use of privileges in, 236-237
SCSI devices, st_clean script, 93
SEAM Administration Tool
and limited administration privileges, 460-461
and list privileges, 460
and X Window system, 432-433
command-line equivalents, 432-433
context-sensitive help, 433
creating a new policy, 442, 453-454
creating a new principal, 442-444
default values, 435
deleting a principal, 445-446
deleting policies, 456
displaying sublist of principals, 439
SEAM Administration Tool (Continued)
duplicating a principal, 444
files modified by, 433
Filter Pattern field, 439
gkadmin command, 431
.gkadmin file, 433
help, 433
Help Contents, 433
how affected by privileges, 460
kadmin command, 431
login window, 434
modifying a policy, 455-456
modifying a principal, 444-445
online help, 433
or kadmin command, 432
overview, 432-436
panel descriptions, 457-460
privileges, 460
setting up principal defaults, 446-447
starting, 434-436
table of panels, 457-460
viewing a principal’s attributes, 440-442
viewing list of policies, 449-451
viewing list of principals, 438-440
viewing policy attributes, 451-453
secondary audit directory, 560
secret keys
creating, 256-258
generating, 256-258 279
secure connection
across a firewall, 320
logging in, 315-316
Secure NFS, 277
Secure RPC
alternative, 53
and Kerberos, 278
description, 277
implementation of, 279-281
keyserver, 279
overview, 52-53
securing
logins task map, 58
passwords task map, 58
scripts, 211
security
across insecure network, 320
security (Continued)
auditing and, 505
BART, 101
computing digest of files, 258-259
computing MAC of files, 259-260
devices, 44-46
DH authentication, 279-281
encrypting files, 261-263
Kerberos authentication, 381
NFS client-server, 279-281
password encryption, 40
pointer to JASS toolkit, 48
policy overview, 32-33
preventing remote login, 70-71
protecting against denial of service, 49
protecting against Trojan horse, 47
protecting devices, 93-95
protecting hardware, 72-73
protecting PROM, 72-73
system hardware, 72-73
security attributes
checking for, 174
considerations when directly assigning, 177-178
description, 172
Printer management rights profile, 173
privileges on commands, 175
special ID on commands, 175
using to mount allocated device, 80
security mechanism, specifying with -m option, 479
security modes, setting up environment with
multiple, 380-382
security policy, default (RBAC), 219
security service, Kerberos and, 346
selecting
audit classes, 524-526
audit records, 546-548
events from audit trail, 546-548
semicolon (;)
device_allocate file, 92
separator of security attributes, 224
sendmail command, authorizations required, 227
seq audit policy
and sequence token, 519 583
description, 519
sequence audit token
and seq audit policy, 583
sequence audit token (Continued)
  format, 583
ServerKeyBits keyword, sshd_config file, 329
servers
  AUTH DH client-server session, 279-281
  configuring for Solaris Secure Shell, 326
  definition in Kerberos, 489
  gaining access with Kerberos, 494-497
  obtaining credential for, 495-496
  realms and, 345-346
service
  definition in Kerberos, 489
  disabling on a host, 466-467
  obtaining access for specific service, 496-497
service keys
  definition in Kerberos, 489
  keytab files and, 461-467
service management facility
  enabling keyserver, 282
  refreshing cryptographic framework, 267
  restarting cryptographic framework, 274
  restarting Solaris Secure Shell, 311
service names, PAM, 295
service principal
  adding to keytab file, 461, 463-464
  description, 344
  planning for names, 355-356
  removing from keytab file, 464-465
session ID, audit, 567
session keys
  definition in Kerberos, 489
  Kerberos authentication and, 493-494
setfacl command
  -d option, 138
  description, 127
  examples, 138
  -f option, 137
  syntax, 135-137
setgid permissions
  absolute mode, 125, 134
  description, 122
  finding files with permissions set, 140
  security risks, 48, 122
  symbolic mode, 124
setuid permissions
  absolute mode, 125, 134
  description, 122
  finding files with permissions set, 140
  security risks, 48, 122
  symbolic mode, 124
sh command, description, 335
sh command, privileged version, 177
SHA1 kernel provider, 264
sharing files
  and network security, 51
  with DH authentication, 286-287
shell, privileged versions, 177
shell commands
  /etc/d_passwd file entries, 44
  passing parent shell process number, 230
  shell process, listing its privileges, 230-232
  shell scripts, writing privileged, 236
  short praudit output format, 557
  .shosts.equiv file, description, 333
  .shosts file, description, 333
  signal received during auditing shutdown, 563
  signing providers, cryptographic framework, 254
  single-sign-on system, 477-483
  Kerberos and, 339
size of audit files
  reducing, 545-546, 555
  reducing storage-space requirements, 521
slave_datatrans file
  description, 486
  KDC propagation and, 400-401
slave_datatrans_slave file, description, 486
slave KDCs
  configuring, 368-372
  definition, 488
  master KDC and, 345-346
  or master, 363
  planning for, 356
  swapping with master KDC, 395-400
slot, definition in cryptographic framework, 251
smartcard documentation, pointer to, 31
smairport command, description, 226
smexec command, description, 226
smmultiuser command, description, 226
smprofile command
  changing rights profile, 206
smprofile command (Continued)
description, 226
smrole command
changing properties of role, 205
description, 226
using, 195-196
smuser command
changing user's RBAC properties, 209
description, 226
socket audit token, 583
soft limit
audit_warn condition, 562
minfree line description, 560
soft string, audit_warn script, 562
Solaris auditing task map, 523
Solaris Cryptographic Framework, See cryptographic framework
solaris.device.revoke command
authorization, 91
Solaris Secure Shell
adding to system, 331
administering, 323-325
administrator task map, 307
authentication
requirements for, 304-306
authentication methods, 304-306
authentication steps, 324-325
basis from OpenSSH, 306
changes in current release, 306
changing passphrase, 314
command execution, 325
configuring clients, 325-326
configuring port forwarding, 310-311
configuring server, 326
connecting across a firewall, 320
connecting outside firewall
from command line, 321
from configuration file, 320-321
copying files, 319-320
creating keys, 312-314
data forwarding, 325
description, 303
files, 332
forwarding mail, 318-319
generating keys, 312-314
keywords, 326-331
local port forwarding, 318-319, 319
Solaris Secure Shell (Continued)
logging in fewer prompts, 316-317
logging in to remote host, 315-316
login environment variables and, 330-331
naming identity files, 332
packages, 331
protocol versions, 303
public key authentication, 304
remote port forwarding, 319
scp command, 319-320
TCP and, 310
typical session, 323-325
user procedures, 311-312
using port forwarding, 317-319
using without password, 316-317
solaris security policy, 224
special permissions
setgid permissions, 122
setuid permissions, 122
sticky bit, 122-123
square brackets ([), bsmrecord output, 570
sr_clean script, description, 94
ssh-add command
description, 334
example, 316-317
storing private keys, 316-317
ssh-agent command
calibrating, 317
description, 334
from command line, 316-317
in scripts, 317
ssh command
description, 334
overriding keyword settings, 335
port forwarding options, 317-319
using, 315-316
using a proxy command, 321
.ssh/config file
description, 333
override, 333
ssh_config file
configuring Solaris Secure Shell, 325-326
host-specific parameters, 330
keywords, 326-331
See specific keyword
override, 333
stash file (Continued)
  definition, 488
sticky bit permissions
  absolute mode, 125-134
description, 122-123
  symbolic mode, 124
stopping, dial-up logins temporarily, 65
storage costs, and auditing, 520
storage overflow prevention, audit trail, 551
storing
  audit files, 515 533-535
  passphrase, 261
StrictHostKeyChecking keyword, ssh_config file, 329
StrictModes keyword, sshd_config file, 329
su command
  displaying access attempts on console, 70-71
  in role assumption, 200-202 202-203
  monitoring use, 70
su file, monitoring su command, 70
subject audit token, format, 584
Subsystem keyword, sshd_config file, 329
success
  audit class prefix, 566
turning off audit classes for, 567
sufficient control flag, PAM, 297
sulog file, 70
monitoring contents of, 70
SUPATH in Solaris Secure Shell, 331
superuser
  compared to privilege model, 178-185
compared to RBAC model, 169-171
differences from privilege model, 180
eliminating in RBAC, 176
monitoring access attempts, 70-71
suser security policy, 224
svcadm command
  administering cryptographic framework, 252
  enabling cryptographic framework, 274
  enabling keyserver daemon, 282
  refreshing cryptographic framework, 266-268
  restarting name service, 191
  restarting NFS server, 534
  restarting Solaris Secure Shell, 311
  restarting syslog daemon, 62 527
svcs command
  listing cryptographic services, 274
svcs command (Continued)
  listing keyserver service, 282
swapping master and slave KDCs, 395-400
symbolic links, file permissions, 121
symbolic mode
  changing file permissions, 124 132
description, 123
synchronizing clocks, 368 372 394-395, 410
SYS privileges, 179
sysconf.rpt file, 146, 150
syslog.conf file
  and auditing, 559
  audit.notice level, 527
  audit records, 505
  executable stack messages, 128
  kern notice level, 128
  priv.debug entry, 245
  saving failed login attempts, 62-63
SYSLOG_FAILED_LOGINS
  in Solaris Secure Shell, 330
  system variable, 62
syslog facility keyword, sshd_config file, 329
System Administrator (RBAC)
  assuming role, 201-202
  creating role, 191-192
  protecting hardware, 72
  recommended role, 170
  rights profile, 214
system calls
  arg audit token, 573
  close, 565
exec args audit token, 575
exec env audit token, 575
ioctl(), 565
ioctl to clean audio device, 94
  return audit token, 582
system file, bsconv effect on, 559
system hardware, controlling access to, 72-73
system properties, privileges relating to, 179
system security
  ACL, 125-127
  dial-up logins and passwords, 43-44
dial-up passwords
  disabling temporarily, 65
system security (Continued)
  displaying
    user’s login status, 58-59, 59
    users with no passwords, 59
  firewall systems, 54
  hardware protection, 38, 72-73
  login access restrictions, 39
  machine access, 38
  overview, 38
  password encryption, 40
  passwords, 39
  privileges, 178-185
  protecting from risky programs, 140-142
  restricted shell, 47
  restricting remote root access, 70-71
  role-based access control (RBAC), 46, 169-171
  root access restrictions, 51 70-71
  saving failed login attempts, 61
  special logins, 42
  su command monitoring, 46 70
task map, 140
system state audit monitoring, 46 70
system-wide administration audit class, 565
System V IPC
  ipc audit class, 565
  ipc audit token, 578
  ipc_perm audit token, 579
  privileges, 179
system variables
  See also variables
  CRYPT_DEFAULT, 66
  KEYBOARD_ABORT, 73
  noexec_user_stack, 141
  noexec_user_stack_log, 142
  rstchown, 131
  SYSLOG_FAILED_LOGINS, 62
system-wide administration audit class, 565
systems, protecting from risky programs, 140-142

T
tables, gsscred, 499
tail command, example of use, 521
tape drives
  allocating, 85
  cleaning of data, 94
tape drives (Continued)

device-clean scripts, 93


task maps

administering cryptographic framework, 263
administering policies (Kerberos), 449
administering principals (Kerberos), 437
administering Secure RPC, 281-282
allocating devices, 84-85
ASET, 160-163
auditing, 523
changing default algorithm for password encryption, 65
configuring audit files, 524
configuring auditing service, 532-533
configuring device policy, 76
configuring devices, 75-76
configuring Kerberos NFS servers, 376
configuring RBAC, 188
configuring Solaris Secure Shell, 307
controlling access to system hardware, 72
cryptographic framework, 255
device allocation, 79-80
device policy, 76
devices, 75-76
enabling auditing service, 532-533
Kerberos configuration, 361-362
Kerberos maintenance, 362
managing and using privileges, 229
managing audit records, 542-543
managing device allocation, 79-80
managing device policy, 76
managing RBAC, 203-204
monitoring and restricting superuser, 69
PAM, 292
planning auditing, 513
protecting against programs with security risk, 140
protecting files, 128
protecting files with ACLs, 134
protecting files with cryptographic mechanisms, 256
protecting files with UNIX permissions, 129
protecting system hardware, 72
running ASET, 160-163
securing logins and passwords, 58
securing systems, 57-58
Solaris Secure Shell, 307
system access, 57-58
task maps (Continued)

Using BART task map, 100-101
using device allocation, 84-85
using RBAC, 187
using roles, 200
using Solaris Secure Shell, 311-312
using the cryptographic framework, 255
TASKS variable (ASET), 152 157
tasks tat command (ASET), 145, 148
TCP
addresses, 579
Solaris Secure Shell and, 310 325
telnet command
description, 487
Kerberos and, 477-480
telnetd daemon, Kerberos and, 487-488
terminal ID, audit, 568
terminating, signal received during auditing shutdown, 563
terminology
authentication-specific, 488-489
Kerberos, 488-493
Kerberos-specific, 488
test manifests, 99
text audit token, format, 585
TGS, getting credential for, 494-495
TGT, in Kerberos, 341-342
third-party password algorithms, adding, 68-69
ticket file, See credential cache
ticket-granting service, See TGS
ticket-granting ticket, See TGT
tickets
creating, 469-470
creating with kinit, 470
definition, 340
definition in Kerberos, 489
destroying, 472
-F option or -f option, 479
file
See credential cache
forwardable, 340, 470 480-481 490
initial, 490
invalid, 490
-k option, 479
klist command, 470-472
lifetime, 491-492
tickets (Continued)
  maximum renewable lifetime, 492
  obtaining, 469-470
  or credentials, 341
  postdatable, 490
  postdated, 340
  proxiable, 490
  proxy, 490
  renewable, 490
  requesting for specific realm, 479
  types of, 489-493
  viewing, 470-472
  warning about expiration, 389
TIMEOUT in Solaris Secure Shell, 330
timestamps
  ASET reports, 149
  audit files, 569
  /tmp/krb5cc_uid file, description, 486
  /tmp/ovsec_adm_xxxxx file, description, 486
  tmpfile string, audit_warn script, 563
  TMPFS file system, security, 123
token, definition in cryptographic framework, 251
trail audit policy
  and trailer token, 519
  description, 519
trailer audit token
  format, 586
  order in audit record, 586
  praudit display, 586
transparency, definition in Kerberos, 340
Trojan horse, 47
troubleshooting
  allocating a device, 85
  ASET errors, 164
  encrypt command, 262
  Kerberos, 429
  lack of privilege, 232-233
  list_devices command, 82
  mounting a device, 87
  privilege requirements, 232-233
  rights profiles, 208
  role capabilities, 193
truss command, for privilege debugging, 232
tune files (ASET) (Continued)
  examples, 158, 159
  modifying, 154
  rules, 159
  tune_rpt file, 145, 150
  types of tickets, 489-493
  TZ in Solaris Secure Shell, 330
U
  -U option
    allocate command, 91
    list_devices command, 90
  uauth audit token, 511, 586
UDP
  addresses, 579
  port forwarding and, 310
  Solaris Secure Shell and, 310
  using for remote audit logs, 508
uid_aliases file (ASET), 151, 153
UID_ALIASES variable (ASET), 151, 153, 157
umask value
  and file creation, 123
  typical settings, 123
umount command, with security attributes, 80
uninstalling, cryptographic providers, 269
UNIX file permissions, See files, permissions
unmounting, allocated devices, 88
update_drv command
  description, 89
  using, 77-78
updating, auditing service, 541-542
URL for online help, SEAM Tool, 359
use_authid option, SASL and, 301
UseLogin keyword, sshd_config file, 329
user accounts
  See also users
    ASET check, 146
    displaying login status, 58-59, 59
User Accounts tool, description, 208-210
user ACL entries
  default entries for directories, 127
  description, 126-127
  setting, 135-137
user administration audit class, 565
Index

user_attr database
  defaultpriv keyword, 244
  description, 219, 220-221
  limitpriv keyword, 244
  RBAC relationships, 219-220
user audit fields, audit_user database, 561
user classes of files, 120
user database (RBAC), See user_attr database
user ID
  audit ID and, 503-504 567
  in NFS services, 379
User keyword, ssh_config file, 329
user principal, description, 344
user procedures
  allocating devices, 84-85
  assuming a role, 200
  chkey command, 286
  computing digest of a file, 258-259
  computing MAC of a file, 259-260
  decrypting files, 261-263
  encrypting files, 256
  encrypting NIS user’s private key, 285
  generating a symmetric key, 256-258
  protecting files, 129-134
  using ACLs, 134-140
  using an assigned role, 200
  using Solaris Secure Shell, 311-312
user scripts, configuring for ssh_agent daemon, 317
useradd command
  adding local user, 198
  description, 226
userdel command, description, 226
UserKnownHostsFile keyword, ssh_config file, 329
UserKnownHostsFile2 keyword, See UserKnownHostsFile keyword
usermod command
  changing user’s RBAC properties, 209
  description, 226
  using to assign role, 196-197
users
  adding local user, 198
  allocating devices, 85
  assigning allocate authorization to, 80-81
  assigning privileges to, 234-235
  assigning RBAC defaults, 224-225
  basic privilege set, 182
users (Continued)
  changing properties from command line, 210
  computing digest of files, 258-259
  computing MAC of files, 259-260
  creating local user, 198
  deallocating devices, 88
  determining directly assigned privileges, 238-239
  determining own privileged commands, 239-240
  disabling login, 60
  displaying login status, 58-59
  encrypting files, 261-263
  having no passwords, 59-60
  initial inheritable privileges, 182
  modifying audit preselection mask of, 528-530
  modifying properties (RBAC), 208-210
  mounting allocated devices, 86-87
  restricting basic privileges, 235
  unmouting allocated devices, 88
using
  ACLs, 135-137
  allocate command, 85
  ASET, 160-163
  BART, 101
  cryptographic framework task map, 255
  deallocate command, 88
  device allocation, 84-85 85
  digest command, 258-259
  encrypt command, 261-263
  file permissions, 128-129
  mac command, 259-260
  mount command, 86
  new password algorithm, 66
  ppriv command, 230
  privileges, 237
  privileges task map, 237
  RBAC task map, 187
  roles, 200
  roles task map, 200
  smrole command, 235
  Solaris Secure Shell task map, 311-312
  ssh-add command, 316-317
  ssh-agent daemon, 316-317
  truss command, 232
  umount command, 88
  usermod command, 234
/usr/aset/asetenv file, 152
/usr/aset directory, 144
/usr/aset/masters/tune files
description, 151
modifying, 154
rules, 159
/usr/aset/masters/uid_aliases file, 151
/usr/lib/kprop command, description, 487
/usr/lib/krb5/kadmind daemon, Kerberos and, 487
/usr/lib/krb5/kpropd daemon, Kerberos and, 488
/usr/lib/krb5/k5kdc daemon, Kerberos and, 488
/usr/lib/krb5/ktkt Warnd daemon, Kerberos and, 488
/usr/lib/krb5/krb5k5dc library, overview, 299
/usr/sbin/gkadmin command, description, 487
/usr/sbin/kadmin command, description, 487
/usr/sbin/kadmin.local command, description, 487
/usr/sbin/kprop command, description, 487
/usr/sbin/kpropd command, description, 487
/usr/share/lib/xml directory, 557
/usrgrp.rpt file
description, 146, 150
example, 150
uucico command, login program, 64

/var/krb5/principal.kadm5.lock file, description, 486
/var/krb5/principal.ok file, description, 486
/var/krb5/principal.ularg file, description, 486
/var/krb5/slave_datatrans file, description, 486
/var/krb5/slave_datatrans slave file, description, 486
/var/log/authlog file, failed logins, 62-63
/var/run/sshd.pid file, description, 332

variables
adding to audit record, 518-575
ASET environment variables
ASETDIR, 156
ASETSECLEVEL, 156
CKLISTPATH_level, 151, 153, 158
PERIODIC_SCHEDULE, 153, 157
summary, 155
TASKS, 152, 157
UID_ALIASES, 151, 153, 157
YPCHECK, 154, 158

auditing those associated with a command, 574
for proxy servers and ports, 321
KEYBOARD_ABORT, 73
login and Solaris Secure Shell, 330-331
noexec_user_stack, 128
noexec_user_stack_log, 128
rstchown, 131
setting in Solaris Secure Shell, 330

verifiers
description, 280
returned to NFS client, 281
window, 280

VerifyReverseMapping keyword, ssh_config file, 329
viewing
ACL entries, 139-140
audit record formats, 543-544
available cryptographic mechanisms, 265, 269
binary audit files, 548-550
contents of rights profiles, 217
cryptographic mechanisms
available, 265, 269
existing, 264, 265, 269
device allocation information, 81-82
device policy, 76-77
digest of a file, 258
directly assigned privileges, 238
existing cryptographic mechanisms, 265, 269
file permissions, 129-130
viewing (Continued)
- keylist buffer with list command, 465, 466
- list of policies, 449-451
- list of principals, 438-440
- MAC of a file, 260
- policy attributes, 451-453
- principal’s attributes, 440-442
- privileges in a shell, 231 238-239
- privileges on a process, 230
- tickets, 470-472
- user’s login status, 58-59
- users with no passwords, 59-60
- XML audit records, 549 557

viruses
- denial of service attack, 49
- Trojan horse, 47

vnode audit token, format, 574

voTd daemon, turned off by device allocation, 81

W
- warn.conf file, description, 486
- warning about ticket expiration, 389
- wildcard characters
  - for hosts in Solaris Secure Shell, 320
  - in ASET files, 157
  - in ASET tune files, 159
  - in RBAC authorizations, 218
- window verifier, 280
- write permissions, symbolic mode, 124

X
- -x option
  - Kerberized commands, 479
  - praudit command, 557
- -X option, Kerberized commands, 479
- X Window system, and SEAM Administration Tool, 432-433
- X11 forwarding
  - configuring in ssh_config file, 327
  - in Solaris Secure Shell, 325
- X11DisplayOffset keyword, sshd_config file, 330
- X11Forwarding keyword, sshd_config file, 329
- X1UseLocalHost keyword, sshd_config file, 330
- xauth command, X11 forwarding, 330
- XAuthLocation keyword, Solaris Secure Shell port forwarding, 330
- XML format, audit records, 550
- XML option, praudit command, 557
- Xylogics tape drive device-cleanup script, 93

Y
- YPCHECK variable (ASET), 154, 158

Z
- zonename audit policy
  - description, 519
  - using, 564
- zonename audit token, 511, 586
- zones
  - auditing and, 564
  - configuring auditing in global zone, 537
  - cryptographic framework and, 254
  - cryptographic services and, 274
  - devices and, 44
  - per zone audit policy, 564
  - planning auditing in, 514-515
  - zonename audit policy, 564