

# Securing Users and Processes in Oracle® Solaris 11.3

ORACLE®

Part No: E54830  
September 2018



**Part No: E54830**

Copyright © 2002, 2018, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

**Référence: E54830**

Copyright © 2002, 2018, Oracle et/ou ses affiliés. Tous droits réservés.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou ce matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

**Accès aux services de support Oracle**

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> ou le site <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> si vous êtes malentendant.

# Contents

---

|   |    |
|---|----|
| <b>Using This Documentation</b> .....                                       | 13 |
| <b>1 About Using Rights to Control Users and Processes</b> .....            | 15 |
| What's New in Rights in Oracle Solaris 11.3 .....                           | 15 |
| User Rights Management .....  | 16 |
| User and Process Rights Provide an Alternative to the Superuser Model ..... | 16 |
| Basics of User and Process Rights .....                                     | 20 |
| More About User Rights .....  | 23 |
| More About User Authorizations .....  | 23 |
| More About Rights Profiles .....  | 23 |
| More About Roles .....  | 24 |
| About Qualified User Attributes .....                                       | 25 |
| Process Rights Management .....   | 25 |
| Privileges Protecting Kernel Processes .....                                | 26 |
| Privilege Descriptions .....  | 27 |
| Administrative Differences on a System With Privileges .....                | 28 |
| More About Privileges .....   | 29 |
| How Privileges Are Implemented .....  | 29 |
| How Privileges Are Used .....   | 30 |
| Privilege Assignment .....  | 33 |
| Privilege Escalation and User Rights .....                                  | 35 |
| Privilege Escalation and Kernel Privileges .....                            | 36 |
| Rights Verification .....   | 37 |
| Profile Shells and Rights Verification .....                                | 37 |
| Name Service Scope and Rights Verification .....                            | 37 |
| Order of Search for Assigned Rights .....                                   | 38 |
| Applications That Check for Rights .....                                    | 39 |
| Considerations When Assigning Rights .....                                  | 40 |

|   |            |
|---|------------|
| Security Considerations When Assigning Rights .....                     | 40         |
| Usability Considerations When Assigning Rights .....                    | 41         |
| <b>2 Planning Your Administrative Rights Configuration .....</b>        | <b>43</b>  |
| Deciding Which Rights Model to Use for Administration .....             | 43         |
| Following Your Chosen Rights Model .....                                | 44         |
| <b>3 Assigning Rights in Oracle Solaris .....</b>                       | <b>47</b>  |
| Assigning Rights to Users .....   | 47         |
| Who Can Assign Rights .....   | 48         |
| Determining Which Rights to Assign to Administrators .....              | 48         |
| Assigning Rights to Users and Roles .....                               | 52         |
| Expanding Users' Rights .....   | 59         |
| Restricting Users' Rights .....   | 66         |
| Setting General User Restrictions .....                                 | 67         |
| Setting Remote Login Restrictions .....                                 | 71         |
| Removing Privileges From Users .....                                    | 72         |
| Removing Rights From Many Users by Using a Rights Profile .....         | 77         |
| Removing Rights System-Wide .....                                       | 77         |
| Qualifying Accounts to Assume Rights on Specific Systems .....          | 79         |
| <b>4 Assigning Rights to Applications, Scripts, and Resources .....</b> | <b>81</b>  |
| Limiting Applications, Scripts, and Resources to Specific Rights .....  | 81         |
| Assigning Rights to Applications and Scripts .....                      | 81         |
| Locking Down Resources by Using Extended Privileges .....               | 85         |
| Users Locking Down the Applications That They Run .....                 | 92         |
| Administering Immutable Zones .....                                     | 95         |
| <b>5 Managing the Use of Rights .....</b>                               | <b>97</b>  |
| Managing the Use of Rights .....  | 97         |
| Using Your Assigned Administrative Rights .....                         | 98         |
| Auditing Administrative Actions .....                                   | 102        |
| Creating Rights Profiles and Authorizations .....                       | 103        |
| Changing Whether root Is a User or a Role .....                         | 109        |
| <b>6 Listing Rights in Oracle Solaris .....</b>                         | <b>113</b> |

|   |            |
|---|------------|
| Listing Rights and Their Definitions .....                            | 113        |
| Listing Authorizations .....  | 114        |
| Listing Rights Profiles .....   | 115        |
| Listing Roles .....   | 117        |
| Listing Privileges .....  | 118        |
| Listing Qualified Attributes .....                                    | 121        |
| <b>7 Troubleshooting Rights in Oracle Solaris .....</b>               | <b>123</b> |
| Troubleshooting RBAC and Privileges .....                             | 123        |
| ▼ How to Troubleshoot Rights Assignments .....                        | 124        |
| ▼ How to Reorder Assigned Rights .....                                | 128        |
| ▼ How to Determine Which Privileges a Program Requires .....          | 129        |
| Troubleshooting Passwords .....                                       | 132        |
| <b>8 Reference for Oracle Solaris Rights .....</b>                    | <b>135</b> |
| Rights Profiles Reference .....                                       | 135        |
| Viewing the Contents of Rights Profiles .....                         | 136        |
| Authorizations Reference .....  | 137        |
| Authorization Naming Conventions .....                                | 137        |
| Delegation Authority in Authorizations .....                          | 137        |
| Rights Databases .....  | 138        |
| Rights Databases and the Naming Services .....                        | 138        |
| user_attr Database .....  | 139        |
| auth_attr Database .....  | 140        |
| prof_attr Database .....  | 140        |
| exec_attr Database .....  | 141        |
| policy.conf File .....  | 141        |
| Commands for Administering Rights .....                               | 142        |
| Commands That Manage Authorizations, Rights Profiles, and Roles ..... | 142        |
| Selected Commands That Require Authorizations .....                   | 143        |
| Privileges Reference .....  | 144        |
| Commands for Handling Privileges .....                                | 144        |
| Files That Contain Privilege Information .....                        | 145        |
| Privileged Actions in the Audit Record .....                          | 145        |
| <b>Glossary .....</b>   | <b>147</b> |

**Index** ..... 151



## Examples

---

|                   |   |    |
|-------------------|---|----|
| <b>EXAMPLE 1</b>  | Determining Which Rights a Command Requires .....                                       | 50 |
| <b>EXAMPLE 2</b>  | Using ARMOR Roles .....   | 53 |
| <b>EXAMPLE 3</b>  | Creating a Role for an Application Administrator .....                                  | 54 |
| <b>EXAMPLE 4</b>  | Creating a User Administrator Role in the LDAP Repository .....                         | 55 |
| <b>EXAMPLE 5</b>  | Creating Roles for Separation of Duty .....   | 55 |
| <b>EXAMPLE 6</b>  | Creating and Assigning a Role to Administer Cryptographic Services .....                | 55 |
| <b>EXAMPLE 7</b>  | Adding a Role to a User .....   | 57 |
| <b>EXAMPLE 8</b>  | Adding a Rights Profile as the Role's First Rights Profile .....                        | 57 |
| <b>EXAMPLE 9</b>  | Replacing a Local Role's Assigned Profiles .....  | 57 |
| <b>EXAMPLE 10</b> | Assigning Privileges Directly to a Role .....   | 58 |
| <b>EXAMPLE 11</b> | Changing the Password of a Role in a Specific Repository .....                          | 59 |
| <b>EXAMPLE 12</b> | Creating a Trusted User to Administer DHCP .....  | 60 |
| <b>EXAMPLE 13</b> | Requiring a User to Type Password Before Administering DHCP .....                       | 61 |
| <b>EXAMPLE 14</b> | Assigning Authorizations Directly to a User .....                                       | 61 |
| <b>EXAMPLE 15</b> | Assigning Authorizations to a Role .....  | 61 |
| <b>EXAMPLE 16</b> | Assigning Privileges Directly to a User .....   | 61 |
| <b>EXAMPLE 17</b> | Adding to a Role's Basic Privileges .....   | 62 |
| <b>EXAMPLE 18</b> | Enabling a User to Use Own Password for Role Password .....                             | 62 |
| <b>EXAMPLE 19</b> | Creating a Rights Profile for Administrators of a Third-Party Application .....         | 63 |
| <b>EXAMPLE 20</b> | Modifying a Rights Profile to Enable a User to Use Own Password for Role Password ..... | 63 |
| <b>EXAMPLE 21</b> | Changing the Value of roleauth for a Role in the LDAP Repository .....                  | 64 |
| <b>EXAMPLE 22</b> | Enabling a Trusted User to Read Extended Accounting Files .....                         | 64 |
| <b>EXAMPLE 23</b> | Enabling a Non-root Account to Read a root-Owned File .....                             | 65 |
| <b>EXAMPLE 24</b> | Removing Privileges From a User's Limit Set .....                                       | 74 |
| <b>EXAMPLE 25</b> | Removing a Basic Privilege From Yourself .....  | 74 |
| <b>EXAMPLE 26</b> | Preventing Guests From Spawning Editor Subprocesses .....                               | 75 |

|            |   |     |
|------------|---|-----|
| EXAMPLE 27 | Assigning the Editor Restrictions Rights Profile to All Users .....                             | 76  |
| EXAMPLE 28 | Modifying the <code>policy.conf</code> File to Limit the Rights Available to System Users ..... | 77  |
| EXAMPLE 29 | Creating a Remote Users Rights Profile .....  | 77  |
| EXAMPLE 30 | Removing Basic Privileges From a Rights Profile .....   | 78  |
| EXAMPLE 31 | Restricting an Administrator to Explicitly Assigned Rights .....                                | 78  |
| EXAMPLE 32 | Preventing Selected Applications From Spawning New Processes .....                              | 79  |
| EXAMPLE 33 | Qualifying Where and When LDAP Users and Roles Can Use Their Rights .....                       | 79  |
| EXAMPLE 34 | Qualifying the Systems Where Users and Roles Have Administrative Rights .....                   | 80  |
| EXAMPLE 35 | Assigning Security Attributes to a Legacy Application .....                                     | 83  |
| EXAMPLE 36 | Running an Application With Assigned Rights .....   | 83  |
| EXAMPLE 37 | Checking for Authorizations in a Script or Program .....  | 84  |
| EXAMPLE 38 | Scripting the Batch Editing of Files in a Directory .....                                       | 84  |
| EXAMPLE 39 | Running a Browser in a Protected Environment .....  | 93  |
| EXAMPLE 40 | Protecting Directories on Your System From Application Processes .....                          | 94  |
| EXAMPLE 41 | Editing a System File .....   | 100 |
| EXAMPLE 42 | Caching Authentication for Ease of Role Use .....   | 100 |
| EXAMPLE 43 | Assuming the <code>root</code> Role .....   | 101 |
| EXAMPLE 44 | Assuming an <code>ARMOR</code> Role .....   | 101 |
| EXAMPLE 45 | Using Two Roles to Configure Auditing .....   | 102 |
| EXAMPLE 46 | Creating a Sun Ray Users Rights Profile .....   | 104 |
| EXAMPLE 47 | Creating a Rights Profile That Includes Privileged Commands .....                               | 104 |
| EXAMPLE 48 | Cloning and Enhancing the Network IPsec Management Rights Profile ....                          | 105 |
| EXAMPLE 49 | Cloning and Removing Selected Rights From a Rights Profile .....                                | 106 |
| EXAMPLE 50 | Testing a New Authorization .....   | 108 |
| EXAMPLE 51 | Adding Authorizations to a Rights Profile .....   | 108 |
| EXAMPLE 52 | Changing the <code>root</code> User Into the <code>root</code> Role .....                       | 110 |
| EXAMPLE 53 | Preventing the <code>root</code> Role From Being Used to Maintain a System .....                | 111 |
| EXAMPLE 54 | Listing All Authorizations .....  | 114 |
| EXAMPLE 55 | Listing the Content of the Authorizations Database .....  | 114 |
| EXAMPLE 56 | Listing the Default Authorizations of Users .....   | 114 |
| EXAMPLE 57 | Listing the Names of All Rights Profiles .....  | 115 |
| EXAMPLE 58 | Listing the Contents of the Rights Profiles Database .....                                      | 115 |
| EXAMPLE 59 | Listing the Default Rights Profiles of Users .....  | 115 |
| EXAMPLE 60 | Listing the Rights Profiles of the Initial User .....   | 116 |

---

|                   |   |     |
|-------------------|---|-----|
| <b>EXAMPLE 61</b> | Listing the Contents of an Assigned Rights Profile .....                              | 116 |
| <b>EXAMPLE 62</b> | Listing the Security Attributes of a Command in a Rights Profile .....                | 117 |
| <b>EXAMPLE 63</b> | Listing the Contents of Rights Profiles That Are Recently Created .....               | 117 |
| <b>EXAMPLE 64</b> | Listing Your Assigned Roles .....   | 118 |
| <b>EXAMPLE 65</b> | Listing All Privileges and Their Definitions .....                                    | 118 |
| <b>EXAMPLE 66</b> | Listing Privileges That Are Used in Privilege Assignment .....                        | 119 |
| <b>EXAMPLE 67</b> | Listing the Privileges in Your Current Shell .....                                    | 119 |
| <b>EXAMPLE 68</b> | Listing the Basic Privileges and Their Definitions .....                              | 120 |
| <b>EXAMPLE 69</b> | Listing the Commands With Security Attributes in Your Rights Profiles ....            | 120 |
| <b>EXAMPLE 70</b> | Listing a User's Qualified Attributes on This System .....                            | 121 |
| <b>EXAMPLE 71</b> | Listing All Qualified Attributes for a User in LDAP .....                             | 121 |
| <b>EXAMPLE 72</b> | Determining Whether You Are Using a Profile Shell .....                               | 127 |
| <b>EXAMPLE 73</b> | Determining the Privileged Commands of a Role .....                                   | 127 |
| <b>EXAMPLE 74</b> | Running the Privileged Commands in Your Role .....                                    | 128 |
| <b>EXAMPLE 75</b> | Assigning Rights Profiles in a Specific Order .....                                   | 129 |
| <b>EXAMPLE 76</b> | Using the <code>truss</code> Command to Examine Privilege Use .....                   | 130 |
| <b>EXAMPLE 77</b> | Using the <code>ppriv</code> Command to Examine Privilege Use in a Profile Shell .... | 130 |
| <b>EXAMPLE 78</b> | Changing a File Owned by the <code>root</code> User .....                             | 131 |
| <b>EXAMPLE 79</b> | Using the <code>openldap</code> System Account to Run a cron Job .....                | 132 |
| <b>EXAMPLE 80</b> | Creating a Role That Requires the User's Password .....                               | 132 |
| <b>EXAMPLE 81</b> | Overriding the Password Requirements for an Account .....                             | 133 |



## Using This Documentation

---

- **Overview** – Describes how to assign additional rights to users, create and use roles, and assign rights to programs and specific resources on Oracle Solaris systems.
- **Audience** – Security administrators.
- **Required knowledge** – Site security requirements.

## Product Documentation Library

Documentation and resources for this product and related products are available at <http://www.oracle.com/pls/topic/lookup?ctx=E53394-01>.

## Feedback

Provide feedback about this documentation at <http://www.oracle.com/goto/docfeedback>.



# About Using Rights to Control Users and Processes

---

Oracle Solaris provides [rights](#) that can be assigned to users, [roles](#), processes, and selected resources. These rights provide a more secure administrative alternative to the [superuser model](#).

This chapter provides information about the elements that support user and process rights management and discusses ways to expand users' rights, limit users' rights, add privileges to commands, and limit applications to just the privileges that they require:

- [“What's New in Rights in Oracle Solaris 11.3” on page 15](#)
- [“User Rights Management” on page 16](#)
- [“Process Rights Management” on page 25](#)

## What's New in Rights in Oracle Solaris 11.3

This section highlights information for existing customers about important new features in user rights, also called role-based access control (RBAC) and new features in process rights, also called privileges.

- The `dax_access` privilege enables data analytics acceleration on the DAX co-processors on SPARC M7 servers and SPARC T7-Series systems for Oracle Database 12c. A database given this privilege can offload parts of query processing to the hardware.
- You can generate a password hash for a user and use the `passwd -p` command to assign it. This functionality is useful for automated installation (AI) and scripting. For more information, see the [pwhash\(1\)](#) and [passwd\(1\)](#) man pages.
- Oracle Solaris provides the `pam_otp_auth` PAM module for processing one-time passwords (OTP). OTPs provide a second authentication step before login. The package that installs the module also installs two PAM stacks in the `/etc/security/pam_policy` directory. For more information, see [“Task Map: Using OTP in Oracle Solaris” in \*Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3\*](#).

- Oracle Solaris provides the `pam_pkcs11` PAM module for managing smart card authentication. Smart cards enable users to log in only if they 1) possess a smart card that is recognized by the login server and 2) can supply the correct PIN. For more information, see [Chapter 7, “Using Smart Cards for Multifactor Authentication in Oracle Solaris”](#) in *Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3*.

## User Rights Management

User rights management is a security feature for controlling user access to tasks that would normally be restricted to the `root` role. By applying [security attributes](#), or *rights*, to processes and to users, the site can divide superuser privileges among several administrators. Process rights management is implemented through *privileges*. User rights management is implemented through *rights profiles*, which collect rights that are then assigned to users or to [roles](#). User rights can also be restricted, such as for kiosks or guest users.

- For a discussion of rights on kernel processes, see [“Process Rights Management”](#) on page 25.
- For procedures to manage rights, see [Chapter 3, “Assigning Rights in Oracle Solaris”](#), [Chapter 4, “Assigning Rights to Applications, Scripts, and Resources”](#), and [Chapter 5, “Managing the Use of Rights”](#).
- For troubleshooting information, see [Chapter 7, “Troubleshooting Rights in Oracle Solaris”](#).
- For reference information, see [Chapter 6, “Listing Rights in Oracle Solaris”](#) and [Chapter 8, “Reference for Oracle Solaris Rights”](#).

## User and Process Rights Provide 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`, as do many `setuid` programs, are also 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 root` program that is hijacked can do anything on the system.

Assigning rights to users, resources, and processes provides a more secure alternative to the all-or-nothing [superuser model](#). With rights, you can enforce security [policy](#) at a more fine-grained level. Rights follows 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. Regular users have enough privilege to use their applications, check the status of their jobs, print files, create new files, and so on. Rights beyond regular user rights are grouped into rights profiles. Users who are expected to do jobs that require some of the rights of superuser can be assigned a rights profile.

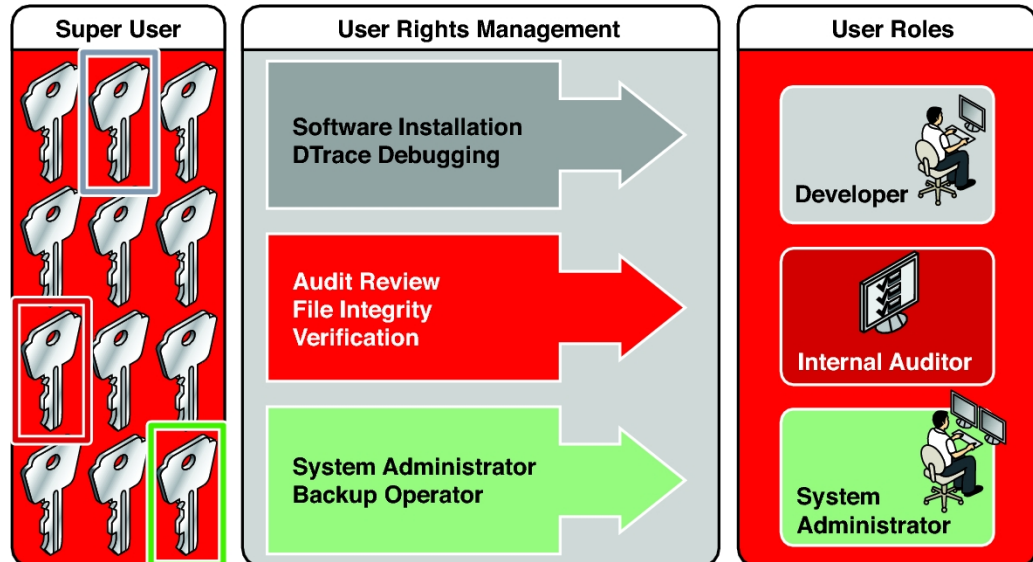
Rights that are grouped into a profile can be assigned directly to users. They can also be indirectly assigned by creating special accounts that are called *roles*. A user can then assume a role to do a job that requires some administrative privileges. Oracle Solaris supplies many predefined rights profiles. You create the roles and assign the profiles.

The ARMOR package provides a set of standardized roles. By auto-installing this package and assigning the roles to users, you can create a system that provides [separation of duty](#) at boot. For more information, see [UNIX Authorization Roles Managed On RBAC \(O-ARMOR\) Reference: C125](#). In this guide, see “[Following Your Chosen Rights Model](#)” on page 44, and [Example 2, “Using ARMOR Roles,”](#) on page 53.

Rights profiles can provide broad administrative rights. For example, the System Administrator rights profile enables an account to perform tasks that are not related to security, such as printer management and cron job management. Rights profiles can also be narrowly defined. For example, the Cron Management rights profile manages `at` and `cron` jobs. When you create roles, the roles can be assigned broad administrative rights or narrow rights.

The following figure illustrates how Oracle Solaris can distribute rights to [trusted users](#) by creating roles. Superuser can also distribute rights by assigning rights profiles directly to trusted users.

FIGURE 1 Distribution of Rights



In the illustrated rights model, superuser creates three 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 names. After login, users assume roles that can run administrative commands and graphical user interface (GUI) tools.

The flexibility in setting up roles enables a variety of security policies. Although few roles are shipped with Oracle Solaris, roles are easily configured. [Example 2, "Using ARMOR Roles," on page 53](#) shows how to use roles that are based on the ARMOR standard. In addition to or in place of ARMOR roles, you can create your own roles based on the rights profiles that Oracle Solaris provides.

- **root** – A powerful role that is equivalent to the root user. However, like all roles, the root role cannot log in. A regular user must log in, then assume the assigned root role. This role is configured and assigned to the initial user by default.
- **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.

---

**Note** - The Media Backup rights profile provides access to the entire root file system. Therefore, while the Media Backup and Operator rights profiles are designed for a junior administrator, you must ensure that the user can be trusted.

---

You might also want to configure one or more security roles. Three rights profiles and their supplementary profiles handle security: Information Security, User Security, and Zone Security. Network security is a supplementary profile in the Information Security rights profile.

Note that roles do not have to be implemented. Roles are a function of an organization's security needs. One strategy is to set up roles 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. You can also assign rights profiles directly to users and not create roles at all.

The [superuser model](#) and the [rights model](#) can co-exist. The following table summarizes the gradations from superuser to restricted regular user that are possible in the rights model. The table includes the administrative actions that can be tracked in both models. For a summary of the effect of process rights, that is, *privileges*, see [Table 2, “Visible Differences Between a System With Privileges and a System Without Privileges,”](#) on page 28.

**TABLE 1** Superuser Model Contrasted With Rights Model

| User Capabilities on a System   | Superuser Model                                  | Rights Model   |
|---|--|--|
| Can become superuser with full superuser privileges                                   | Can  | Can  |
| Can log in as a user with full user rights  | Can  | Can  |
| Can become superuser with limited rights  | Cannot   | Can  |
| Can log in as a user, and have superuser privileges sporadically                      | Can, with <code>setuid</code> root programs only | Can, with <code>setuid</code> root programs and with rights  |
| Can log in as a user with administrative rights but without full superuser privileges | Cannot   | Can, with rights profiles, roles, and with directly assigned privileges and authorizations   |
| Can log in as a user with fewer rights than a regular user                            | Cannot   | Can, by removing rights  |
| Can track superuser actions   | Can, by auditing the <code>su</code> command     | Can, by auditing calls to <code>pfexec()</code><br><br>Also, the name of the user who has assumed the <code>root</code> role is in the audit trail |

## Basics of User and Process Rights

The terms *unprivileged* or *without rights* do not apply in Oracle Solaris. Every process in Oracle Solaris, including regular user processes, has at least some privileges or other user rights, such as authorizations. To learn about the basic set of privileges that Oracle Solaris grants to all UNIX processes, see [“Process Rights Management” on page 25](#).

The following elements enforce user rights in Oracle Solaris. These rights can be configured to enforce permissive security policies or restrictive security policies.

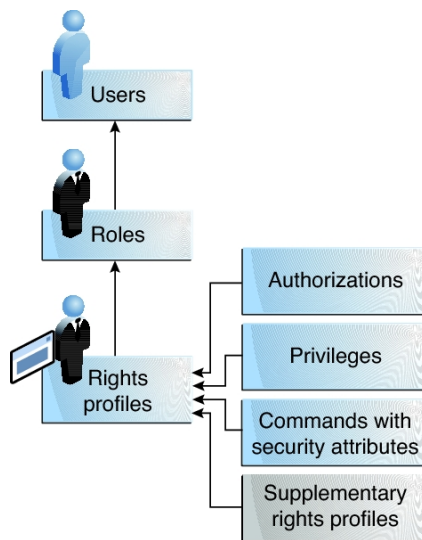
- **Authorization** – A permission that enables a user or role to perform a class of actions that require additional rights. For example, the default security policy gives console users the `solaris.device.cdrw` authorization. This authorization enables users to read and write to a CD-ROM device. For a list of authorizations, use the `auths list` command. Authorizations are enforced at the user application level, not in the kernel. See [“More About User Authorizations” on page 23](#).
- **Privilege** – A right that can be granted to a command, a user, a role, or a specific resources, such as a port or SMF method. Privileges are implemented in the kernel. For example, the `proc_exec` privilege allows a process to call `execve()`. Regular users have basic privileges. To see your basic privileges, run the `ppriv -vl basic` command. For more information, see [“Process Rights Management” on page 25](#).
- **Security attributes** – An attribute that enables a process to perform an operation, or the implementation of a right. In a typical UNIX environment, a security attribute enables a process to perform an operation that is otherwise forbidden to regular users. For example, `setuid` and `setgid` programs have security attributes. In the rights model, authorizations and privileges are [security attributes](#) in addition to `setuid` and `setgid` programs. These attributes, or rights, can be assigned to a user. For example, a user with the `solaris.device.allocate` authorization can allocate a device for exclusive use. Privileges can be placed on a process. For example, a process with the `file_flag_set` privilege can set immutable, no-unlink, or append-only file attributes.  
  
Security attributes can also limit rights. For example, the `access_times` and `access_tz` security attributes set the days and times and optionally the timezone when specific security-relevant operations are permitted. You can limit users directly or by assigning them an [authenticated rights profile](#) that contains these keywords. For more information, see the `user_attr(4)` man page.
- **Privileged application** – An application or command that can override system controls by checking for rights. For more information, see [“Applications That Check for Rights” on page 39](#) and [Developer’s Guide to Oracle Solaris 11.3 Security](#).
- **Rights profile** – A collection of rights that can be assigned to a role or to a user. A rights profile can include authorizations, directly assigned privileges, commands with [security attributes](#), and other rights profiles. Profiles that are within another profile are called

*supplementary rights profiles*. Rights profiles offer a convenient way to group rights. They can be directly assigned to users or to special accounts called *roles*. You can use the commands in a rights profile only if your process recognizes rights. Additionally, you can be required to supply a password. Alternatively, password [authentication](#) can be supplied by default. See [“More About Rights Profiles” on page 23](#).

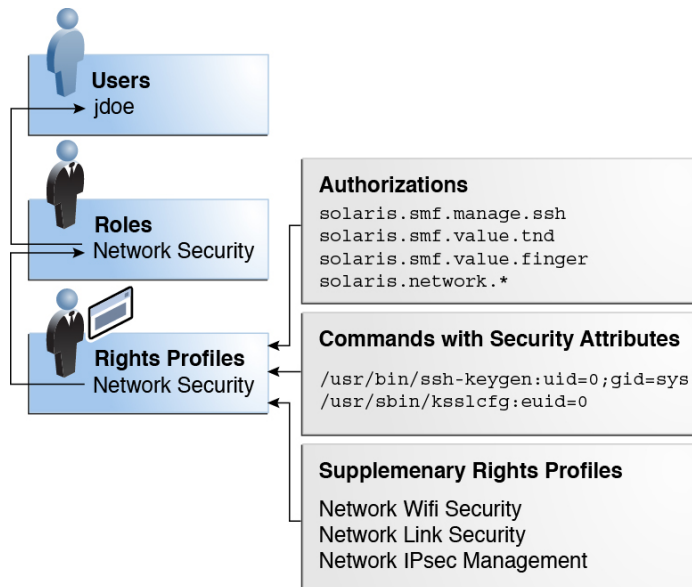
- **Role** – A special identity for running *privileged applications*. The special identity can be assumed by assigned users only. In a system that is run by roles, superuser can be unnecessary after initial configuration. See [“More About Roles” on page 24](#).
- **Qualified user attribute** – A security attribute that can be applied to user and role accounts in LDAP and therefore can be centrally managed. For example, you can limit a user to specified access times or assign a user a role or a rights profile on designated systems only. See [“About Qualified User Attributes” on page 25](#).

The following figure shows how user rights and process rights work together.

**FIGURE 2** User Rights and Process Rights Working Together



The following figure uses the Network Security role and the Network Security rights profile to demonstrate how assigned rights work.

**FIGURE 3** Example of a User Rights and Process Rights Assignment

The Network Security role is used to manage IPsec, wifi, and network links. 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 administrator can enable the role to authenticate by using the user password rather than a role password.

In the figure, the Network Security rights profile is assigned to the Network Security role. The Network Security rights profile contains supplementary profiles that are evaluated in order, Network Wifi Security, Network Link Security, and Network IPsec Management. These supplementary profiles contain rights that complete the role's primary tasks.

The Network Security rights profile has three directly assigned authorizations, no directly assigned privileges, and two commands with security attributes. The supplementary rights profiles have directly assigned authorizations, and two of them have commands with security attributes.

When jdoe assumes the Network Security role, the shell changes to a [profile shell](#). The profile shell process can evaluate the use of rights, so jdoe can administer network security.

## More About User Rights

This section provides more details about the implementation and use of rights at the user level.

### More About User Authorizations

An *authorization* is a right that can be granted to a role, a program, a zone, or a user. Authorizations enforce policy at the user application level. Like privileges, mistaken assignments of authorizations can result in more rights being granted than originally intended. For more information, see [“Privilege Escalation and User Rights” on page 35](#).

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 can be prevented from using a [privileged application](#) or from performing security-sensitive operations within a privileged application. For a fuller discussion of privileges, see [“Process Rights Management” on page 25](#).

Rights-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 about authorizations, see the following sections:

- [“Authorizations Reference” on page 137](#)
- [“auth\\_attr Database” on page 140](#)
- [“Selected Commands That Require Authorizations” on page 143](#)

### More About Rights Profiles

A *rights profile* is a collection of rights that can be assigned to a role or user to perform tasks that require administrative rights. A rights profile can include authorizations, privileges, commands with assigned [security attributes](#), and other rights profiles. Rights profiles can also contain entries to reduce or extend the initial inheritable set of privileges and to reduce the limit set.

An *authenticated rights profile* is a rights profiles that requires the user to supply a password, or to *reauthenticate*. The administrator decides which profiles can be used without user reauthentication. A good example of a profile that would not require reauthentication is the

Basic Solaris User rights profile. Depending on site security requirements, rights profiles for security-sensitive tasks might require reauthentication.

For reference information about rights profiles, see the following sections:

- [“Rights Profiles Reference” on page 135](#)
- [“prof\\_attr Database” on page 140](#)
- [“exec\\_attr Database” on page 141](#)

## More About 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 rights. Roles cannot inherit rights from other roles or from the user who assumes the role. Roles distribute superuser privileges, and thus enable more secure administrative practices.

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 at the command line by running the `su` command and supplying the role name and the role's password. The administrator can configure the system to enable a user to authenticate by supplying the user's password. See [Example 18, “Enabling a User to Use Own Password for Role Password,” on page 62](#).

A role cannot log in directly. A user logs in, and then assumes a role. Once you have assumed a role, you cannot assume another role without first exiting your current role.

Also, while a rights profile adds rights to the user's environment, a role gives the user a clean execution environment that is shared with other users who can assume that role. When a user switches to a role, none of the user's authorizations or rights profiles applies to the role.

The `passwd`, `shadow`, and `user_attr` databases store static role information. You can and should audit the actions of roles.

For detailed information about setting up roles, see the following sections:

- [“Following Your Chosen Rights Model” on page 44](#)
- [“Assigning Rights to Users” on page 47](#)

The fact that `root` is a role in Oracle Solaris prevents anonymous `root` login. If the profile shell command, `pfexec`, is being audited, the audit trail contains the login user's real UID, any roles that the user has assumed, and the privileged operations that were performed. To audit the system for privileged operations, see [“Auditing Administrative Actions” on page 102](#).



## About Qualified User Attributes

Qualified user attributes are attributes that can be assigned to users and roles, and to hosts and groups of hosts called netgroups. Netgroups simplify administration of a set of systems, such as a lab network. These qualifiers apply only to LDAP accounts, not to the files naming service.

Qualified and unqualified user attributes are maintained independently, and cannot be combined. This independence allows administrators to assign both qualified and unqualified extended policy attributes to a single user or role. At runtime, the system first determines the hostname where the execution is occurring, then applies the appropriate set of policy attributes.

The `usermod` and `rolemod` commands accept a qualifier option, `-q` to indicate the host or netgroup where the security attributes apply. Separate `usermod` and `rolemod` commands are required to manage each set of qualified or unqualified attributes. The `userdel` and `roledel` commands can remove a complete set of qualified attributes without affecting other qualified or unqualified attributes.

The policy for applying the appropriate set of user attributes follows the search order specified by the `name-service/switch` service and is cached by the `name-service/cache` service. The order is:

1. A local entry matching the named user or role
2. One or more LDAP entries of the named user or role's qualified attributes
3. An LDAP entry whose hostname matches the current host
4. A netgroup (in LDAP) that has the current host as a member
5. An unqualified entry for the named LDAP user or role
6. In the absence of an assigned Stop rights profile, default attributes specified in the `/etc/policy.conf` file

If a match is found, it is cached to optimize subsequent queries. For examples, see [Example 33, “Qualifying Where and When LDAP Users and Roles Can Use Their Rights,”](#) on page 79.

## Process Rights Management

Process rights management in Oracle Solaris is implemented by *privileges*. Privileges enable processes to be restricted at the level of command, user, role, and specific system resource. Privileges decrease the security risk that is associated with one user or one process having full superuser powers on a system. Process rights and user rights provide a compelling alternative model to the traditional [superuser model](#).

Traditionally, privileges are used to add rights. However, privileges can also be used to restrict rights, for example, changing a `setuid root` program to a program that is [privilege-aware](#). Also, with an *extended privilege policy*, administrators can allow only specified privileges to be used with a file object, user ID, or port. This fine-grained privilege assignment denies all other privileges except basic privileges to these resources.

- For information about extended privilege policy and restrictive privileges, see [“Using Extended Privilege Policy to Restrict Privilege Use”](#) on page 35.
- For information about user rights, see [“User Rights Management”](#) on page 16.
- For information about how to administer privileges, see [Chapter 3, “Assigning Rights in Oracle Solaris”](#).
- For reference information about privileges, see [“Privileges Reference”](#) on page 144.

## Privileges Protecting Kernel Processes

A privilege is a 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 *basic set* of privileges operates within the bounds of the system security policy. `setuid root` 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 root`.

Privileges 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 manipulates files might require the `file_dac_write` and `file_flag_set` privileges. These privileges on the process eliminate the need to run the program as `root`.

Historically, systems have not followed the [privilege model](#), or rights model, as introduced in [“Basics of User and Process Rights”](#) on page 20. Rather, systems used the [superuser model](#). In the superuser model, processes were 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 discretionary access control (DAC) to protect system files. Device nodes were protected by DAC. For example, devices owned by the group `sys` could be opened only by members of that group.

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 root` 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 provides a gradation between user rights and root rights. A user can be granted privileges to perform activities that are beyond the rights of regular users, and root can be limited to fewer privileges than root currently possesses. With rights, a command that runs with privileges can be isolated in a rights profile and assigned to one user or role. [Table 1, “Superuser Model Contrasted With Rights Model,” on page 19](#) summarizes the gradation between user rights and root privileges that the rights model provides.

The rights model provides greater security than the superuser model. Privileges that have been removed from a process cannot be exploited. Process privileges can provide an additional safeguard for sensitive files and devices in contrast to DAC protections alone, which can be exploited to gain access.

Privileges, then, can restrict programs and processes to just the rights that the program requires. On a system that implements [least privilege](#), an intruder who captures a process can access 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.

Other logical groups include CONTRACT, CPC, DAX, DTRACE, GRAPHICS, VIRT, and WIN.

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 `net_rawaccess` privilege indicates its broad effect:

```
net_rawaccess
    Allows a process to have direct access to the network layer.
```

The [privileges\(5\)](#) man page provides descriptions of every privilege. See also “[Listing Privileges](#)” on page 118.

## 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 2** Visible Differences Between a System With Privileges and a System Without Privileges

| Feature            | No Privileges  | Privileges  |
|--------------------|--|---|
| Daemons            | Daemons run as root.   | Daemons run as the user daemon.<br><br>For example, these daemons are assigned limited privileges and run as daemon: <code>lockd</code> and <code>rpcbind</code> .  |
| Log file ownership | Log files are owned by root.   | Log files are owned by daemon, who creates the log file. The root user does not own the file.   |
| Error messages     | Error messages refer to superuser.<br><br>For example, <code>chroot: not superuser</code> .                                  | Error messages reflect the use of privileges.<br><br>For example, the equivalent error message for <code>chroot</code> failure is <code>chroot: exec failed</code> .  |
| setuid programs    | Programs use <code>setuid root</code> to complete tasks that regular users are not allowed to perform.                       | Many <code>setuid root</code> programs run with just the privileges they need.<br><br>For example, the following commands use privileges: <code>audit</code> , <code>ikeadm</code> , <code>ipadm</code> , <code>ipsecconf</code> , <code>ping</code> , <code>traceroute</code> , and <code>newtask</code> . |
| File permissions   | Device permissions are controlled by DAC. For example, members of the group <code>sys</code> can open <code>/dev/ip</code> . | File permissions (DAC) do not predict who can open a device. Devices are protected with DAC <i>and</i> device policy.<br><br>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.                                |
| Audit events       | Auditing the use of the <code>su</code> command covers many administrative functions.  | Auditing the use of privileges covers most administrative functions. The <code>cusa</code> audit class includes audit events that monitor administrative functions.   |

| Feature   | No Privileges   | Privileges   |
|-----------|---|--|
| Processes | Processes are protected by the rights of the process owner. | Processes are protected by privileges. Process privileges and process flags are visible as a new entry in the <code>/proc/&lt;pid&gt;/priv</code> directory.   |
| Debugging | No reference to privileges in core dumps.                   | 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.<br><br>The <code>ppriv</code> command and other commands show the proper number of properly sized sets. The commands correctly map the bits in the bit sets to privilege names. |

## More About Privileges

This section covers privilege implementation, use, and assignment details.

### 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** – 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** – 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 not added. 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 about [privilege-aware](#) programs, see [Chapter 2, “Developing Privileged Applications”](#) in *Developer’s Guide to Oracle Solaris 11.3 Security*.

- **Inheritable privilege set, or I** – The set of privileges that a process can inherit across a call to `exec`. After the call to `exec`, the inherited privileges are placed in the permitted set and the effective set, thus making these sets 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** – The set that defines the outside limit of which 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](#).

The kernel recognizes a basic privilege set. On an unmodified system, each user's initial inheritable set equals the basic set at login. While you cannot modify the basic set, you can modify which privileges a user inherits from 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
```

At login, all users would have the basic set in their inheritable set, their permitted set, and their effective set. A user's limit set is equivalent to the default limit set for the zone, global or non-global.

You can assign additional privileges directly to a user, or more precisely to a user's login process, indirectly to many users through a rights profile, and indirectly by assigning a privileged command to a user. You can also remove privileges from a user's basic set. For procedures and examples, see [Chapter 3, “Assigning Rights in Oracle Solaris”](#).

## How Privileges Are Used

Privileges are built into Oracle Solaris. This section describes how Oracle Solaris uses privileges with devices, in resource management, and with legacy applications.

## How Processes Get Privileges

Processes can inherit privileges or be assigned privileges. A process inherits privileges from its parent process. At login, the user's initial inheritable set of privileges determines which 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, roles, and specific resources. 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 opened, 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, and therefore can change the ownership of a file that the role does not own.

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 privileges 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 enables and disables the use of privilege during program execution. To succeed in a production environment, the program must be assigned the privileges that the program enables and disables. Before you make a privilege-aware program available, you assign to the executable only 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.

For examples of privilege-aware code, see [Chapter 2, “Developing Privileged Applications” in \*Developer’s Guide to Oracle Solaris 11.3 Security\*](#). To assign privileges to a program that requires privileges, see [Example 35, “Assigning Security Attributes to a Legacy Application,” on page 83](#) and [Example 47, “Creating a Rights Profile That Includes Privileged Commands,” on page 104](#).

## Privileges and Devices

In the rights model, privileges protect system interfaces that in the superuser model are protected by file permissions alone. 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 gain full access to the system.

Therefore, in Oracle Solaris, 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/ip` device. The file permissions on `/dev/ip` are `0666`, but the `net_rawaccess` privilege is also required to open the device.

Because device policy is controlled by privileges, you have more flexibility in granting permission to open devices. Privilege requirements are configurable for the device policy and for the driver proper. You can configure the privilege requirements when installing, adding, or updating a device driver.

For more information, see the [add\\_drv\(1M\)](#), [devfsadm\(1M\)](#), [getdevpolicy\(1M\)](#), and [update\\_drv\(1M\)](#) man pages.

## Privileges and Resource Management

In Oracle Solaris, you can use the `project.max-locked-memory` and `zone.max-locked-memory` resource controls to limit the memory consumption of processes that are assigned the `PRIV_PROC_LOCK_MEMORY` privilege. This privilege allows a process to lock pages in physical memory.

If you assign the `PRIV_PROC_LOCK_MEMORY` privilege to a rights profile, you can give the processes that have this privilege the ability to lock all memory. As a safeguard, set a resource control to prevent the user of the privilege from locking all memory. For privileged processes that run in a non-global zone, set the `zone.max-locked-memory` resource control. For privileged processes that run on a system, create a project and set the `project.max-locked-memory` resource control. For information about these resource controls, see [Chapter 6, “About Resource Controls” in \*Administering Resource Management in Oracle Solaris 11.3\*](#) and [Chapter 1, “Configuration Resources for Non-Global Zones” in \*Oracle Solaris Zones Configuration Resources\*](#).

## Legacy Applications and the Use of Privileges

To accommodate legacy applications, the implementation of privileges works with both the superuser and the rights 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 rights. 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.



## Debugging Use of Privilege

Oracle Solaris 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 examples, see “[Troubleshooting RBAC and Privileges](#)” on page 123. You can also use the `dttrace` command. For more information, see the [dttrace\(1M\)](#) man page and [Oracle Solaris 11.3 DTrace \(Dynamic Tracing\) Guide](#).

## Privilege Assignment

The term “[privilege](#)” traditionally indicates an increase in rights. Because every process on an Oracle Solaris system runs with some rights, you can decrease the rights on a process by removing privileges. In this release, you can also use an *extended privilege policy* to remove most privileges except the ones that are given to certain resources by default.

## Assigning Privileges to Users and Processes

In your capacity as security administrator, you are responsible for assigning privileges. Existing rights profiles have privileges already assigned to commands in the profile. You then assign the rights profile to a role or user.

Privileges can also be assigned directly to a user, a role, or a rights profile. 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 broader effects, such as `file_dac_write`. For a fuller discussion, see “[Security Considerations When Assigning Rights](#)” on page 40.

Privileges can also be denied to a user, role, or process. Care must be taken when removing privileges from the initial inheritable set or the limit set of a user or role.

## Expanding a User or Role's Privileges

Users and roles have an inheritable set of privileges. The limit set can only be reduced because the limit set is initially all privileges. The initial inheritable set can be expanded for users, roles, and processes by assigning a privilege that is not in the inheritable set.

You can expand the privileges that are available in three ways:

- A privilege that is not in the initial inheritable set but is in the limit set can be assigned to users and roles. The assignment can be indirect, through a privileged command in a rights profile, or it can be direct.
- A privilege that is not in the inheritable set can be explicitly assigned to a process, such as adding privileges to a script or application.
- A privilege that is not in the inheritable set but is in the limit set can be explicitly assigned to a network port, UID, or file object. This use of privilege is called an *extended privilege policy* and is also a means of restricting available privileges. For more information, see [“Using Extended Privilege Policy to Restrict Privilege Use” on page 35](#).

The assignment of a privilege to just the administrative task that requires the privilege is the most precise way to expand a user or role's privileges. You create a rights profile that includes the command or script with its required privileges. Then, you assign this rights profile to a user or role. Such assignment enables the user or role to run that privileged command. The privilege is otherwise unavailable to the user.

Expanding the initial inheritable set of privileges for users or roles is a less desirable 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. For a fuller discussion, see [“Security Considerations When Assigning Rights” on page 40](#).

To reduce unnecessary privilege availability, you can assign *extended privileges* to network ports, UIDs, and file objects. Such assignment removes privileges that are not in the extended privilege assignment from the effective set. For a discussion, see [“Using Extended Privilege Policy to Restrict Privilege Use” on page 35](#).

## Restricting Privileges for a User or Role

Privileges and rights profiles can also be applied to untrusted users to restrict their rights. 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 root` program might fail because the program requires a privilege that was removed. For examples of privilege removal, see [Example 24, “Removing Privileges From a User's Limit Set,” on page 74](#) and [Example 46, “Creating a Sun Ray Users Rights Profile,” on page 104](#).

To limit the privileges that are available to a user ID, port, or file object, see [“Using Extended Privilege Policy to Restrict Privilege Use” on page 35](#).

## 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 rights profile executes the script in a profile shell. If the script contains commands that require privileges, the commands with added privileges must also be in an assigned rights profile. For examples, see [“Assigning Rights to Applications and Scripts” on page 81](#).

## Using Extended Privilege Policy to Restrict Privilege Use

Extended privilege policy can restrict access to ports, user IDs, or file objects except for the basic privileges and the privileges that you explicitly grant. With so few privileges, the resource cannot easily be used to attack the system. In fact, users can protect files and directories that they own from access by potentially malicious processes. For examples of extended privilege policy, see [“Limiting Applications, Scripts, and Resources to Specific Rights” on page 81](#).

## Privilege Escalation and User Rights

Oracle Solaris provides administrators with a great deal of flexibility when configuring security. As installed, the software prevents [privilege escalation](#). *Privilege escalation* occurs when a user or process gains more administrative rights than you intended to grant. In this sense, "privilege" means all rights, not just kernel privileges. See [“Privilege Escalation and Kernel Privileges” on page 36](#).

Oracle Solaris software includes rights that are assigned to the root role only. With other security protections in place, an administrator might assign attributes that are designed for the root role to other accounts, but such assignment must be made with care.

The following rights profile and set of authorizations can escalate the privileges of a non-root account:

- **Media Restore rights profile** – This profile is not part of any other rights profile. Because Media Restore provides access to the entire root file system, its use is a possible escalation of privilege. Deliberately altered files or substitute media could be restored. By default, the root role includes this rights profile.
- **solaris.\*.assign authorizations** – These authorizations are not assigned to any rights profile. An account with a `solaris.*.assign` authorization could assign rights to others

that the account itself is not assigned. For example, a role with the `solaris.profile` assign authorization can assign rights profiles to other accounts that the role itself is not assigned. By default, only the root role has `solaris.*.assign` authorizations.

Assign `solaris.*.delegate` authorizations, rather than `solaris.*.assign` authorizations. A `solaris.*.delegate` authorization enables the delegater to assign other accounts only those rights that the delegater possesses. For example, a role that is assigned the `solaris.profile.delegate` authorization can assign rights profiles that the role itself is assigned to other users and roles.

For the prevention of escalation of kernel privileges, see [“Privilege Escalation and Kernel Privileges” on page 36](#).

## Privilege Escalation and Kernel Privileges

The kernel prevents [privilege escalation](#). To prevent a process from gaining more privileges than the process should have, the kernel checks that vulnerable system modifications have the full set of privileges. For example, a file or process that is owned by root (UID=0) can be changed only by a process with the full set of privileges. The root account 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. Specifically, 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.

The `file_chown_self` privilege is limited by the `rstchown` system variable. When the `rstchown` variable is set to 0, the `file_chown_self` privilege is removed from the initial inheritable set of all users of the system image. For more information about the `rstchown` system variable, see the [chown\(1\)](#) man page.

The `file_chown_self` privilege is most safely assigned to a particular command, the command placed in a rights profile, and the profile assigned to a role or a trusted user.

- 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, the command placed in a profile, and the profile assigned to a role.



---

**Caution** - You can configure a user's account to include the `file_chown_self` privilege or the `proc_owner` privilege in the user's initial inheritable set. However, you should have overriding security reasons for placing such powerful privileges in any user or role's inheritable set.

---

For information about how privilege escalation is prevented for devices, see [“Privileges and Devices” on page 31](#). For a general discussion, see the `privileges(5)` man page.

## Rights Verification

The shell that a process runs in, the scope of the naming service, and the order of search can affect whether assigned rights are evaluated. Processes whose rights cannot be evaluated fail. For assistance in checking for rights assignments, see [“Troubleshooting RBAC and Privileges” on page 123](#).

## Profile Shells and Rights Verification

Users and roles can run privileged applications from a profile shell. A *profile shell* is a special shell that recognizes rights. Administrators can assign a profile shell to users as a login shell, or the profile shell is started when a user runs the `pfexec` command or the `su` command to assume a role. In Oracle Solaris, every shell has a profile shell counterpart. For a list of profile shells, see the `pfexec(1)` man page.

Users who are directly assigned a rights profile and whose login shell is not a profile shell must open a profile shell to run the privileged commands that they are assigned. Users and roles who are assigned an authenticated rights profile are prompted to authenticate, that is, to provide a password before the command can execute. For usability and security considerations, see [“Considerations When Assigning Rights” on page 40](#).

## Name Service Scope and Rights Verification

Name service scope affects when assigned rights are available. The scope of a role might be limited to an individual host. Alternatively, the scope might include all hosts that are served by a naming service such as LDAP. The name service scope for a system is specified in the name switch service, `svc:/system/name-service/switch`. 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. For information about naming services, see the `nsswitch.conf(4)` man page, [Working With Oracle Solaris 11.3 Directory and Naming Services: DNS and NIS](#), and [Working With Oracle Solaris 11.3 Directory and Naming Services: LDAP](#).

## Order of Search for Assigned Rights

A user or role can be assigned [security attributes](#) directly or through a rights profile. The order of search affects which security attribute value is used. The value of the first found instance of the attribute is used.

---

**Note** - The order of authorizations is not important. Authorizations are cumulative.

---

When a user logs in, rights are assigned in the following search order:

- **Rights** that are assigned directly to the user with the `useradd` and `usermod` commands. For a list of possible rights assignments, see [“user\\_attr Database” on page 139](#).
- **Rights profiles** that are assigned to the user with the `useradd` and `usermod` commands. These assignments are searched in order.
  - First, the authenticated rights profiles are searched.  
The order is the first profile in the authenticated profiles list and then its supplementary profiles, the second profile in the authenticated profiles list and then its supplementary profiles, and so on. The first instance of a value is the one that the system uses, except for `auths` values, which are cumulative. The attributes that can be assigned to rights profiles include all the rights that can be assigned to users, plus supplementary profiles. For the list, see [“user\\_attr Database” on page 139](#).
  - Then, the rights profiles that do not require reauthentication are searched in the same fashion.
- **Console User rights profile** value. For a description, see [“Rights Profiles Reference” on page 135](#).
- If the **Stop rights profile** is assigned, the evaluation of security attributes stops. No attributes are assigned after the Stop profile is assigned. The Stop profile is evaluated after the Console User rights profile and before the other security attributes in the `policy.conf` file, including `AUTHS_GRANTED`. For a description, see [“Rights Profiles Reference” on page 135](#).
- **Basic Solaris User rights profile** value in the `policy.conf` file.
- `AUTHS_GRANTED` value in the `policy.conf` file.

- AUTH\_PROFS\_GRANTED value in the `policy.conf` file.
- PROFS\_GRANTED value in the `policy.conf` file.
- PRIV\_DEFAULT value in the `policy.conf` file.
- PRIV\_LIMIT value in the `policy.conf` file.

## Applications That Check for Rights

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 an assigned UID in a rights profile. A user or role with that rights profile can then run the program as that UID 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 `reboot` command requires a real rather than an effective UID.

---

**Tip** - If an effective ID is not sufficient to run a command, assign the real ID to the command.

---

### Applications That Check for Privileges

Privileged applications can check for the use of privileges. The rights profile mechanism enables you to specify the privileges for specific commands that require security attributes. Then, you can isolate the command with assigned 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.

Commands that check for privileges include the following:

- Kerberos commands, such as `kadmin`, `kprop`, and `kdb5_util`

- Network commands, such as `ipadm`, `routeadm`, and `snoop`
- File and file system commands, such as `chmod`, `chgrp`, and `mount`
- Commands that control processes, such as `kill`, `pcrd`, and `rcapadm`

To add commands with privileges to a rights profile, see [“How to Create a Rights Profile” on page 103](#) and the `profiles(1)` man page. To determine which commands check for privileges in a particular profile, see [Chapter 6, “Listing Rights in Oracle Solaris”](#).

## Applications That Check Authorizations

Some Oracle Solaris commands check authorizations, including the following:

- Audit administration commands, such as `auditconfig` and `auditreduce`
- Printer administration commands, such as `cupsenable` and `lpadmin`
- Batch job commands, such as `at`, `atq`, `batch`, and `crontab`
- Device-oriented commands, such as `allocate`, `deallocate`, `list_devices`, and `cdrw`.

For guidance about checking a script or program for authorizations, see [Example 37, “Checking for Authorizations in a Script or Program,” on page 84](#). To write a program that requires authorizations, see [“About Authorizations” in \*Developer’s Guide to Oracle Solaris 11.3 Security\*](#).

## Considerations When Assigning Rights

Security and usability issues can affect how administrators assign rights.

## Security Considerations When Assigning Rights

Typically, users or roles obtain administrative rights through a rights profile, but direct assignment of rights is also possible.

- Privileges can be assigned directly to users and roles.  
Direct assignment of privileges is not a secure practice. Users and roles with a directly assigned privilege can override security policy wherever this privilege is required by the kernel. Also, malicious processes that compromise a user or role’s process can use this privilege wherever it is required by the kernel.



A more secure practice is to assign the privilege as a security attribute of a command in a rights profile. Then, that privilege is available only for that command by someone who has that rights profile.

- Authorizations can be assigned directly to users and roles.

Because authorizations are evaluated 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 assigning audit flags. For greater security, assign authorizations in an authenticated rights profile where the user must supply a password before the command can execute.

## Usability Considerations When Assigning Rights

Direct assignment of rights can affect usability.

- Directly assigned authorizations and the commands and authorizations in a user's rights profile must be interpreted by a profile shell to be effective. By default, users are not assigned a profile shell. Therefore, users must remember to open a profile shell and execute the commands in that shell.
- Singly assigning authorizations is not scalable. Also, directly assigned authorizations might not be sufficient to perform a task. The task might require privileged commands. Rights profiles are designed to bundle authorizations and privileged commands together. They also scale well to groups of users.



## Planning Your Administrative Rights Configuration

---

This chapter provides information to help you decide whether to use a traditional rights model or to fully take advantage of the Oracle Solaris rights model when administering your system. The chapter covers the following topics:

- [“Deciding Which Rights Model to Use for Administration” on page 43](#)
- [“Following Your Chosen Rights Model” on page 44](#)

For an overview of rights, see [“User Rights Management” on page 16](#). For reference information, see [Chapter 8, “Reference for Oracle Solaris Rights”](#).

### Deciding Which Rights Model to Use for Administration

Rights in Oracle Solaris include rights profiles, authorizations, and privileges. Oracle Solaris offers several ways to configure administrative rights on a system.

The following list is ordered from most secure to the less secure traditional [superuser model](#).

1. Divide administrative tasks among several [trusted users](#), each of whom has limited rights. This approach is the Oracle Solaris rights model.

For information about how to follow this approach, see [“Following Your Chosen Rights Model” on page 44](#).

For a discussion of the benefits of this approach, see [Chapter 1, “About Using Rights to Control Users and Processes”](#).

2. Use the default rights configuration. This approach uses the rights model but does not customize it to your site.

By default, the initial user has some administrative rights and can assume the root role. Optionally, the root role could assign the root role to another trusted user. For greater security, the root role would enable the auditing of administrative commands.

Tasks that are useful to administrators who use this model are the following:

- [“Using Your Assigned Administrative Rights” on page 98](#)
- [“Assigning Rights to Users” on page 47](#)
- [“Auditing Administrative Actions” on page 102](#)
- [“Changing a Role Password” on page 58](#)
- [Chapter 6, “Listing Rights in Oracle Solaris”](#)

3. Use the `sudo` command.

Administrators who are familiar with the `sudo` command can configure `sudo` and use it. Optionally, they can configure the `/etc/sudoers` file to enable `sudo` users to run administrative commands without [reauthentication](#) for a set period of time.

Tasks that are useful to `sudo` users are the following:

- [“Using Your Assigned Administrative Rights” on page 98](#)
- [“Auditing Administrative Actions” on page 102](#)
- [Example 42, “Caching Authentication for Ease of Role Use,” on page 100](#)

The `sudo` command is the Linux equivalent of the Oracle Solaris RBAC commands. Unlike the RBAC commands, `sudo` cannot reference rights profiles. It runs as `root` with all privileges so that it can grant the rights that are specified for each program in the `/etc/sudoers` file for the current user. For more information, see the [`sudo\(1m\)`](#) and [`sudoers\(4\)`](#) man pages.

4. Use the [superuser model](#) by changing the `root` role into a user.

Administrators who use the traditional UNIX model must complete [“How to Change the root Role Into a User” on page 109](#). Optionally, the `root` user can configure auditing.

## Following Your Chosen Rights Model

User and process rights management can be an integral part of managing your systems deployment. Planning requires a thorough knowledge of the security requirements of your organization as well as an understanding of rights in Oracle Solaris. This section describes the general process for planning your site's use of rights.

1. Learn the basic concepts about rights.

Read [Chapter 1, “About Using Rights to Control Users and Processes”](#). Using rights to administer a system is very different from using conventional UNIX administrative practices.

2. Examine your security [policy](#).

Your organization's security policy details the potential threats to your system, measures the risk of each threat, and provides strategies to counter these threats. Isolating the security-relevant tasks through rights can be a part of the strategy.

For example, your site might require that you separate security administration from non-security administration. To implement [separation of duty](#), see [Example 5, "Creating Roles for Separation of Duty," on page 55](#).

If your security policy relies on Authorization Rules Managed On RBAC (ARMOR), you must install and use the ARMOR package. For its use in Oracle Solaris, see [Example 2, "Using ARMOR Roles," on page 53](#).

3. Review the default rights profiles.

The default rights profiles collect the rights that are required to complete a task. To review available rights profiles, see ["Listing Rights Profiles" on page 115](#)

4. Decide whether you are going to use [roles](#) or assign rights profiles to users directly.

Roles can ease the administration of rights. The role name identifies the tasks that the role can perform and isolates role rights from user rights. If you are going to use roles, you have three options:

- You can install the ARMOR package, which installs the seven roles that the Authorization Roles Managed on RBAC (ARMOR) standard defines. See [Example 2, "Using ARMOR Roles," on page 53](#).
- You can define your own roles and also use ARMOR roles. See ["Creating a Role" on page 52](#) and [Example 2, "Using ARMOR Roles," on page 53](#).
- You can define your own roles and not use ARMOR roles. See ["Creating a Role" on page 52](#).

If roles are not required at your site, you can directly assign rights profiles to users. To require a password when users perform an administrative task from their rights profiles, use authenticated rights profiles. See [Example 13, "Requiring a User to Type Password Before Administering DHCP," on page 61](#).

5. Decide whether you need to create additional rights profiles.

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 using rights. For example, users of Sun Ray systems do not require all basic privileges. For an example of a rights profile that limits users, see [Example 30, "Removing Basic Privileges From a Rights Profile," on page 78](#).

- a. Determine which rights are needed for the new task.
- b. Decide whether an existing rights profile is appropriate for this task.
- c. Order the rights profile so that commands execute with their required privileges.

For information about ordering, see ["Order of Search for Assigned Rights" on page 38](#).

6. Decide which users should be assigned which rights.

According to the [principle of least privilege](#), you assign users to roles that are appropriate to the user's level of trust. When you prevent users from performing tasks that the users do not need to perform, you reduce potential problems.

---

**Note** - Rights that apply to all users of a system image are specified in the `/etc/security/policy.conf` file.

---

Once you have a plan, create logins for [trusted users](#) who can be assigned rights profiles or roles. For details on creating users, see [“Task Map for Setting Up and Managing User Accounts by Using the CLI”](#) in *Managing User Accounts and User Environments in Oracle Solaris 11.3*.

To assign rights, start with the procedures in [“Assigning Rights to Users”](#) on page 47. The sections that follow provide examples of expanding rights, limiting rights, assigning rights to resources, and troubleshooting rights assignments.

## Assigning Rights in Oracle Solaris

---

This chapter describes tasks for assigning rights to users and roles. The chapter covers the following topics:

- [“Assigning Rights to Users” on page 47](#)
- [“Expanding Users' Rights” on page 59](#)
- [“Restricting Users' Rights” on page 66](#)

For an overview of rights, see [“User Rights Management” on page 16](#). For reference information, see [Chapter 8, “Reference for Oracle Solaris Rights”](#).

---

**Note** - This chapter assumes that you protect your user passwords and role passwords as described in [“Passwords and Password Policy” in Oracle Solaris 11.3 Security and Hardening Guidelines](#).

---

### Assigning Rights to Users

Rights in Oracle Solaris exist on every process. You can add rights to users and [roles](#), and remove rights. Rights include privileges on the user's process, privileges or special IDs on a command that the user runs, and authorizations to perform a particular action. To ease the administrative burden of assigning rights, Oracle Solaris collects rights for services and administrative actions into *rights profiles*. Rather than assign individual rights to users and roles, you can assign a [rights profile](#) that includes all the authorizations and privileges that the administrative task requires.

Roles give a name to the administrative task that a user can perform, such as `audi.tadm`. To perform an administrative action, the user assumes an assigned role to perform the action. Roles can be required by security [policy](#) and they can simply be convenient. You can create roles or you can install the `armor` package which creates seven roles and their local home directories. For more information about roles, see [“User and Process Rights Provide an Alternative to the Superuser Model” on page 16](#).

## Who Can Assign Rights

Initially, you must be in the root role to assign rights.

If the root role has distributed administrative tasks to you as a trusted user or by assigning a role to you, the following rights profiles assignments enable you to create users and roles or assign rights to them:

- To create a user or role, you must become an administrator who is assigned the User Management rights profile.
- To assign most rights to a user or role, you must become an administrator who is assigned the User Security rights profile.

You cannot assign audit flags. Only the root role can assign audit flags to a user or role.

You cannot change the password of a role. Only the root role can change a role's password.

If you are assigned administrative rights, review [“Using Your Assigned Administrative Rights” on page 98](#) before you try to run administrative commands.

## Determining Which Rights to Assign to Administrators

Administrators require rights to run privileged commands, and often require authorization to run the commands. Rights profiles supply privileged commands, authorizations, and sometimes supplementary rights profiles in a convenient bundle.

You have several ways to determine which rights profile is best to assign. The names of rights profiles indicate their function, so you can list and search the profile names for functional areas. You can also start with a command name, and determine which rights profiles include that command.

When you know the name of the rights profile that contains the commands you are interested in, and you review the rights in that rights profile, then you can determine whether to assign that particular profile to an administrator. You should not assign individual privileges or authorizations to administrators. For more information, see [“Considerations When Assigning Rights” on page 40](#).

After you assign administrative rights, ask your administrators to review [“Using Your Assigned Administrative Rights” on page 98](#) before they run administrative commands.



## ▼ How to Determine Which Rights to Assign

You can search for which rights to assign by starting with rights profiles or with command names. This procedure shows how to search by rights profile. [Example 1, “Determining Which Rights a Command Requires,” on page 50](#) shows how to search by command.

### 1. List the available rights profiles.

```
$ profiles -a | more
...
Administrative Command History
Administrator Message Edit
Audit Configuration
...
```

### 2. Search for a functional area.

In the following example, you search for rights profiles about administering zones.

```
$ profiles -a | grep -i zone
Zone Security
Zone Configuration
Zone Management
Zone Migration
```

### 3. Review the contents of the rights profile that best describes the rights you plan to assign.

Continuing with the zones example, you are going to assign rights to secure zones.

```
$ profiles -p "Zone Security" info
name=Zone Security
desc=Zones Virtual Application Environment Security
auths=solaris.zone.*,solaris.auth.delegate
cmd=/usr/sbin/tzxonemgr
cmd=/usr/sbin/zoncfg
cmd=/usr/lib/rad/module/mod_zonemgr.so.1
```

The output indicates that the assignee will have all authorizations that begin with the string `solaris.zone`, and the `solaris.auth.delegate` authorization. The assignee can run the `txzonemgr` and `zoncfg` commands, and use the RAD command `mod_zonemgr.so.1` module.

For details about the rights that are assigned to the commands, continue with the following step. For descriptions of the `solaris.zone` authorizations, see [Step 5](#).

### 4. Search for the commands in the privileged commands database.

```
$ getent exec_attr | grep "^Zone Security"
```

```
Zone Security:solaris:cmd:R0::/usr/sbin/txzonemgr:uid=0
Zone Security:solaris:cmd:R0::/usr/sbin/zonecfg:uid=0
Zone Security:solaris:cmd:R0::/usr/lib/rad/module/mod_zonemgr.so.1:uid=0
```

The output indicates that the commands will run with a UID of 0, not with the assignee's UID. R0 indicates that this rights profile is read-only.

**5. (Optional) Review the definitions of the authorizations that are in your chosen rights profile.**

```
$ getent auth_attr | grep solaris.zone
solaris.zone.:R0::Zone Management::
solaris.zone.clonefrom:R0::Clone another Zone::
solaris.zone.login:R0::Zone Login::
solaris.zone.manage:R0::Zone Deployment::
solaris.zone.config:R0::Modify the Persistent Zone Configuration::
solaris.zone.liveconfig:R0::Inspect and Modify the Live Zone Configuration::
solaris.zone.migrate:R0::Zone Migration::
solaris.zone.migrate.cold:R0::Zone Cold Migration::
$ getent auth_attr | grep solaris.auth.delegate
solaris.auth.delegate:R0::Assign owned authorizations::
```

**Example 1** Determining Which Rights a Command Requires

In this example, the administrator wants to assign the `pfctl` command to a network administrator, but does not know what other rights the assignee might need to handle the Packet Filter (PF) firewall.

1. The administrator searches the privileged commands database, `exec_attr`, for the `pfctl` command.

```
$ getent exec_attr | grep pfctl
Network Firewall Management:solaris:cmd:R0::/usr/sbin/pfctl:privs=sys_ip_config
```

The output indicates that the `pfctl` command is part of the Network Firewall Management rights profile and runs with the `sys_ip_config` privilege.

2. The administrator reviews the content of the rights profile.

```
$ profiles -p "Network Firewall Management" info
name=Network Firewall Management
desc=Firewall Administration
auths=solaris.smf.value.network.firewall,solaris.smf.manage.network.firewall
cmd=/usr/sbin/pfconf
cmd=/usr/sbin/pfctl
```

The output indicates that the Network Firewall Management profile authorizes the assignee to modify the SMF properties of the firewall, and also contains the `pfconf` command.

3. The administrator looks up the `pfconf` command in the privileged commands database.

```
$ getent exec_attr | grep pfconf
Network Firewall Management:solaris:cmd:RO::/usr/sbin/pfconf:privs=sys_ip_config
```

4. The administrator reviews the definitions of the authorizations that are in the chosen profile.

```
$ getent auth_attr | grep firewall
solaris.smf.manage.network.firewall:RO::Manage Network Firewall::
solaris.smf.value.network.firewall:RO::Change Network Firewall Configuration::
solaris.smf.manage.firewall:RO::Manage Firewall Service::
solaris.smf.value.firewall.config:RO::Change Service Firewall Config::
```

5. If the rights profile includes all the functions the assignee needs, the administrator assigns it to the user, or creates a role and assigns the role to the user. For examples, see [“Creating a Role” on page 52](#) and [Example 12, “Creating a Trusted User to Administer DHCP,” on page 60](#).
6. If the assignee needs more network capabilities, the administrator continues to investigate.

The administrator lists all network rights profiles, chooses another one, and repeats the search.

```
$ profiles -a | grep ^Network
Network Autoconf Admin
Network Autoconf User
Network ILB
Network Dot1x Management
Network LLDP
Network VRRP
Network DLMP
Network Management
Network Observability
Network TCP Key Management
Network Security
Network Wifi Management
Network Wifi Security
Network Link Security
Network IPsec Management
Network Firewall Management
```

The administrator can also create a custom networking rights profile by following the instructions in [“Creating Rights Profiles and Authorizations” on page 103](#).

## Assigning Rights to Users and Roles

This section describes the commands that create and modify roles and users. To create or modify rights profiles, see [“How to Create a Rights Profile” on page 103](#) and [“How to Clone and Modify a System Rights Profile” on page 105](#).

For information about roles, see [“Basics of User and Process Rights” on page 20](#).

The main actions in creating and modifying roles and users are as follows:

- Creating a role
- Creating a user who is trusted with additional rights
- Modifying the rights of a role
- Modifying the rights of a user
- Enabling users to use their own password to assume a role
- Changing a role password
- Deleting a role

### Creating a Role

If you are going to use roles, you have several options. You can install the predefined roles from ARMOR and use them exclusively. You can also create roles. You can also combine the use of ARMOR roles with the roles that you create.

To use ARMOR roles, see [Example 2, “Using ARMOR Roles,” on page 53](#).

To create your own roles, you use the `roleadd` command. For a full list of the arguments to this command, see the [`roleadd\(1M\)` man page](#).



---

**Caution** - Do not configure a role with both the `roleauth=user` and `auth_profiles=profiles` keywords. The authentication will fail because the `auth_profiles` keyword authenticates against the current process owner's password (role), which is not the `roleauth` password (user).

---

For example, the following commands create a local User Administrator role with a home directory and a `pfbash` login shell, and create a password for the role:

```
# roleadd -c "User Administrator role, local" \  
-m -K profiles="User Security,User Management" accountadm  
80 blocks
```

```
# ls /export/home/accountadm
local.bash_profile  local.login  local.profile
# passwd accountadm
Password: xxxxxxxx
Confirm Password: xxxxxxxx
```

where:

|                     |   |
|---------------------|---|
| -c <i>comment</i>   | Describes the role.   |
| -m                  | Creates a home directory.   |
| -K <i>profiles=</i> | Assigns one or more rights profiles to the role. For the list of rights profiles, see <a href="#">“Listing Rights Profiles” on page 115</a> . |
| <i>rolename</i>     | The name of the role. For restrictions on acceptable strings, see the <a href="#">roleadd(1M)</a> man page.                                   |

---

**Note** - A role account can be assigned to more than one user. Therefore, an administrator typically creates a role password and provides the users with the role password out of band. For an alternative to the role password, see [“Enabling Users to Use Own Password for Role Password” on page 58, Example 18](#), [“Enabling a User to Use Own Password for Role Password,” on page 62](#), and [Example 20, “Modifying a Rights Profile to Enable a User to Use Own Password for Role Password,” on page 63](#).

---

#### EXAMPLE 2 Using ARMOR Roles

In this example, the security administrator installs roles that are defined by the ARMOR standard. The administrator first verifies that the role names do not conflict with any existing accounts, then installs the package, views the role definitions, and assigns the roles to [trusted users](#).

First, the administrator ensures that the following UIDs and names do not exist in the naming service:

- 57 auditadm
- 55 fsadm
- 58 pkgadm
- 53 secadm
- 56 svcadm
- 59 sysop

■ 54 useradm

After verifying that the UIDs and names are not in use, the administrator installs the package.

```
# pkg install system/security/armor
```

The package creates seven roles and local home directories in the /export/home directory.

To view the rights of each role, the administrator can list the profiles that are assigned to each role.

```
# profiles auditadm
# profiles fsadm
# profiles pkgadm
# profiles secadm
# profiles svcadm
# profiles sysop
# profiles useradm
```

The rights that are assigned to ARMOR roles cannot be modified. To create a different configuration of rights, see [“How to Clone and Modify a System Rights Profile” on page 105](#).

Finally, the administrator assigns the roles to trusted users. The users' own passwords are used to authenticate to the role. Some users are assigned more than one role. Roles whose tasks are time-critical are assigned to more than one trusted user.

```
# usermod -R=auditadm adoe
# usermod -R=fsadm, pkgadm bdoe
# usermod -R=secadm, useradm cdoe
# usermod -R=svcadm ddoe
# usermod -R=svcadm edoe
# usermod -R=sysop fdoe
# usermod -R=sysop gdoe
```

**EXAMPLE 3** Creating a Role for an Application Administrator

The administrator creates a role for the Oracle database administrator (DBA) to ensure that the administrator's login name is in the audit trail. Because the oracle account is a role, users must log in to their own account and then switch user to the oracle role. This site has three DBAs.

```
# usermod -K type=role oracle
# usermod -R oracle adoe
# usermod -R oracle bdoe
# usermod -R oracle cdoe
```

**EXAMPLE 4** Creating a User Administrator Role in the LDAP Repository

The administrator creates a User Administrator role in LDAP. The user provides a password when assuming the role, then does not need to supply a password for individual commands.

```
# roleadd -c "User Administrator role, LDAP" -m -S ldap \
-K profiles="User Security,User Management" accountadm
```

**EXAMPLE 5** Creating Roles for Separation of Duty

The administrator creates two roles. The usermgt role can create users, give them home directories, and perform other non-security tasks. The usersec role cannot create users, but can assign passwords and change other rights assignments. Neither role can set audit flags for users or roles, or change a role's password. The root role must perform those actions.

```
# roleadd -c "User Management role, LDAP" -s /usr/bin/pfksh \
-m -S ldap -K profiles="User Management" usermgt
# roleadd -c "User Security role, LDAP" -s /usr/bin/pfksh \
-m -S ldap -K profiles="User Security" usersec
```

The administrator ensures that two people are necessary to create every regular user in [Example 7, “Adding a Role to a User,” on page 57](#).

**EXAMPLE 6** Creating and Assigning a Role to Administer Cryptographic Services

In this example, the administrator on an LDAP network creates a role to administer the Cryptographic Framework, and assigns the role to UID 1111.

```
# roleadd -c "Cryptographic Services manager" \
-g 14 -m -u 104 -S ldap -K profiles="Crypto Management" cryptomgt
# passwd cryptomgt
New Password: xxxxxxxx
Confirm password: xxxxxxxx
# usermod -u 1111 -R +cryptomgt
```

The user with UID 1111 logs in, then assumes the role and displays the assigned rights.

```
$ su - cryptomgt
Password: xxxxxxxx
$ profiles -l
    Crypto Management
    /usr/bin/kmfcfg          euid=0
    /usr/sbin/cryptoadm     euid=0
    /usr/sfw/bin/CA.pl      euid=0
    /usr/sfw/bin/openssl    euid=0
```

For information about the Cryptographic Framework, see [Managing Encryption and Certificates in Oracle Solaris 11.3](#).

## Creating a Login for a Trusted User

You use the `useradd` command to create a login. For a full list of the arguments to the `useradd` command, see the [useradd\(1M\)](#) man page. The rights-related arguments to the command are similar to the `roleadd` command, with the addition of the `-R rolename` option.

If you assign a role to a user, the user can use the role's rights after assuming the role. For example, the following command creates a trusted user who can assume the `accountadm` role that you created in [“Creating a Login for a Trusted User” on page 56](#).

```
# useradd -c "Trusted Assistant User Manager user" -m -R accountadm \  
        -s /usr/bin/pfbash jdoe  
80 blocks  
# ls /export/home/jdoe  
local.bash_profile  local.login  local.profile
```

where:

- `-m` Creates a home directory for the user.
- `-R rolename` Assigns the name of an existing role.
- `-s shell` Determines the login shell for `username`. This shell can be a [profile shell](#), such as `pfbash`. For reasons to assign a profile shell to a trusted user, see [“Usability Considerations When Assigning Rights” on page 41](#). For a list of profile shells, see the [pfexec\(1\)](#) man page.

For more examples, see [“Task Map for Setting Up and Managing User Accounts by Using the CLI” in Managing User Accounts and User Environments in Oracle Solaris 11.3](#).

## Modifying a User's Rights

You use the `usermod` command to modify a user account. For a full list of the arguments to the `usermod` command, see the [usermod\(1M\)](#) man page. The rights-related arguments to the command are similar to the `useradd` command.

If you assign a rights profile to a user, the user can use the rights after the user opens a profile shell. For example, assign a rights profile and a profile shell as the user's login shell:

```
# usermod -s /usr/bin/pfbash -K profiles="User Management" kdoe
```



The changes are in effect at the user's next login. For users to learn how to use their assigned rights, refer them to [“Using Your Assigned Administrative Rights” on page 98](#).

**EXAMPLE 7** Adding a Role to a User

In this example, the administrator ensures that two [trusted users](#) are necessary to create regular users. The roles were created in [Example 5, “Creating Roles for Separation of Duty,” on page 55](#).

```
# usermod -R +accountadm jdoe
# usermod -R +usersec mdoe
```

## Modifying a Role's Rights

You use the `rolemod` command to modify a role account. For a full list of the arguments to the `rolemod` command, see the [`rolemod\(1M\)` man page](#). The rights-related arguments to the command are similar to the `roleadd` command.

The values of *key=value* pairs, and the `-A`, `-P`, and `-R` options can be modified by a minus (-) or plus (+) sign. The - sign indicates to subtract the value from the currently assigned values. The + sign indicates to add the value to the currently assigned values. For rights profiles, the value is prepended to the current list of profiles. For the effects of being an earlier rights profile, see [“Order of Search for Assigned Rights” on page 38](#).

**EXAMPLE 8** Adding a Rights Profile as the Role's First Rights Profile

For example, prepend a rights profile to the `accountadm` role:

```
# rolemod -K profiles+="Device Management" accountadm
# profiles accountadm
accountadm:
Device Management
User Management
User Security
```

**EXAMPLE 9** Replacing a Local Role's Assigned Profiles

In this example, the security administrator modifies the `prtmgmt` role to include the `VSCAN` Management rights profile after the Printer Management profile.

```
# rolemod -c "Handles printers and virus scanning" \  
-K profiles="Printer Management,VSCAN Management,ALL" prtmgt
```

**EXAMPLE 10** Assigning Privileges Directly to a Role

In this example, the security administrator entrusts the `realtime` role with a very specific privilege that affects system time. To assign the privilege to a user, see [Example 16, “Assigning Privileges Directly to a User,”](#) on page 61.

```
# rolemod -K defaultpriv+=proc_clock_highres realtime
```

The values for the `defaultpriv` keyword are in the list of privileges in the role's processes at all times.

## Enabling Users to Use Own Password for Role Password

To enable users to use their own password rather than a role password when assuming a role, modify the role.

The following command enables all users who are assigned the `accountadm` role to use their own password when assuming any assigned role, including the `accountadm` role.

```
# rolemod -K roleauth=user accountadm
```



---

**Caution** - Do not configure a role with both the `roleauth=user` and `auth_profiles=profiles` keywords. The authentication will fail because the `auth_profiles` keyword authenticates against the current process owner's password (role), which is not the `roleauth` password (user).

---

The `auth_profiles` keyword is not designed for roles, though you can use it with roles. This keyword is designed for reauthenticating with a user's credentials without requiring the use of a separate role account.

## Changing a Role Password

Because a role can be assigned to many users, users who are assigned a role cannot change the role password. You must be in the `root` role to change a role password.

```
# passwd accountadm  
Enter accountadm's password: xxxxxxxx  
New: xxxxxxxx  
Confirm: xxxxxxxx
```

If you do not specify a repository, the password is changed in all repositories. For more command options, see the [passwd\(1\)](#) man page.

**EXAMPLE 11** Changing the Password of a Role in a Specific Repository

In the following example, the root role changes the password of the local devadmin role.

```
# passwd -r files devadmin
New password: xxxxxxxx
Confirm password: xxxxxxxx
```

In the following example, the root role changes the password of the devadmin role in the LDAP naming service.

```
# passwd -r ldap devadmin
New password: xxxxxxxx
Confirm password: xxxxxxxx
```

## Deleting a Role

When you delete a role, the role immediately becomes unusable.

```
# roledel accountadm
```

Users who are currently performing administrative tasks in the role are prevented from continuing. The `profiles` command shows the following output:

```
accountadm $ profiles
Unable to get user name
```

## Expanding Users' Rights

The tasks and examples in this section add rights to the rights that users receive by default. For information about rights, see [Chapter 1, “About Using Rights to Control Users and Processes”](#).

- Assign a role to a trusted user – [Example 2, “Using ARMOR Roles,” on page 53](#), [Example 6, “Creating and Assigning a Role to Administer Cryptographic Services,” on page 55](#), [Example 7, “Adding a Role to a User,” on page 57](#)
- Assign a rights profile to a trusted user – [Example 12, “Creating a Trusted User to Administer DHCP,” on page 60](#), [Example 22, “Enabling a Trusted User to Read Extended Accounting Files,” on page 64](#), [Example 35, “Assigning Security Attributes to a Legacy Application,” on page 83](#)

- Assign an [authenticated rights profile](#) to a trusted user – [Example 13, “Requiring a User to Type Password Before Administering DHCP,” on page 61](#), [Example 36, “Running an Application With Assigned Rights,” on page 83](#)
- Assign an authorization to a trusted user or role – [Example 14, “Assigning Authorizations Directly to a User,” on page 61](#), [Example 15, “Assigning Authorizations to a Role,” on page 61](#)
- Assign privileges directly to a user or role – [Example 10, “Assigning Privileges Directly to a Role,” on page 58](#), [Example 16, “Assigning Privileges Directly to a User,” on page 61](#), [Example 17, “Adding to a Role’s Basic Privileges,” on page 62](#)



---

**Caution** - Inappropriate use of directly assigned privileges and authorizations can result in unintentional breaches of security. For a discussion, see [“Security Considerations When Assigning Rights” on page 40](#).

---

- Enable a user to use own password when assuming a role – [Example 18, “Enabling a User to Use Own Password for Role Password,” on page 62](#), [Example 20, “Modifying a Rights Profile to Enable a User to Use Own Password for Role Password,” on page 63](#)
- Create a rights profile for the administrator of a third-party application – [Example 19, “Creating a Rights Profile for Administrators of a Third-Party Application,” on page 63](#)
- Modify a rights profile – [“Removing Privileges From Users” on page 72](#)
- Add security attribute to a command in a rights profile – [Example 32, “Preventing Selected Applications From Spawning New Processes,” on page 79](#), [Example 26, “Preventing Guests From Spawning Editor Subprocesses,” on page 75](#), [Example 47, “Creating a Rights Profile That Includes Privileged Commands,” on page 104](#)
- Enable a user to read a root-owned file – [Example 22, “Enabling a Trusted User to Read Extended Accounting Files,” on page 64](#), [Example 23, “Enabling a Non-root Account to Read a root-Owned File,” on page 65](#)
- Enable a user or role to edit a root-owned file – [Example 49, “Cloning and Removing Selected Rights From a Rights Profile,” on page 106](#)
- Assign a rights profile that contains a new authorization – [Example 51, “Adding Authorizations to a Rights Profile,” on page 108](#)

**EXAMPLE 12**    Creating a Trusted User to Administer DHCP

The security administrator creates a user who can administer DHCP.

```
# useradd -K profiles="DHCP Management" -s /usr/bin/pfbash -S ldap jdoe
```

Because the user is assigned the pfbash login shell, the rights in the DHCP Management rights profile are always evaluated, so the DHCP administrative commands succeed.

**EXAMPLE 13** Requiring a User to Type Password Before Administering DHCP

In this example, the security administrator requires `jdoe` to provide a password before managing DHCP.

```
# usermod -K auth_profiles="DHCP Management" profiles="Edit Administrative Files" jdoe
```

When `jdoe` types a DHCP command, the password prompt appears. After authenticating `jdoe`, the DHCP command completes. In search order, authenticated rights profiles are processed before regular profiles.

```
jdoe$ dhcpconfig -R 120.30.33.7,120.30.42.132
Password: xxxxxxxx
    /** Command completes **/
```

**EXAMPLE 14** Assigning Authorizations Directly to a User

In this example, the security administrator creates a local user who can control screen brightness.

```
# useradd -c "Screened KDoE, local" -s /usr/bin/pfbash \
-K auths=solaris.system.power.brightness kdoe
```

This authorization is added to the user's existing authorization assignments.

**EXAMPLE 15** Assigning Authorizations to a Role

In this example, the security administrator creates a role that can change the configuration information for the DNS server service.

```
# roleadd -c "DNS administrator role" -m -K auths="solaris.smf.manage.bind" dnsadmin
```

**EXAMPLE 16** Assigning Privileges Directly to a User

In this example, the security administrator trusts the user `kdoe` with a very specific privilege that affects system time. To assign the privilege to a role, see [Example 10, “Assigning Privileges Directly to a Role,”](#) on page 58.

```
# usermod -K defaultpriv='basic,proc_clock_highres' kdoe
```

The values for the `defaultpriv` keyword replace the existing values. Therefore, for the user to retain the basic privileges, the value `basic` is specified. In the default configuration, all users have basic privileges. For the list of basic privileges, see [“Listing Privileges”](#) on page 118.

The user can view the added privilege and its definition.

```
kdoe$ ppriv -v $$
1800: pfksh
flags = <none>
E: file_link_any,...,proc_clock_highres,sys_ib_info
I: file_link_any,...,proc_clock_highres,sys_ib_info
P: file_link_any,...,proc_clock_highres,sys_ib_info
L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,win_upgrade_sl
$ ppriv -vl proc_clock_highres
Allows a process to use high resolution timers.
```

**EXAMPLE 17** Adding to a Role's Basic Privileges

In the following example, the role `realtime` is directly assigned privileges to handle date and time programs. You assigned the `proc_clock_highres` to `realtime` in [Example 10, “Assigning Privileges Directly to a Role,”](#) on page 58.

```
# rolemod -K defaultpriv='basic,sys_time' realtime

$ su - realtime
Password: xxxxxxxx
$ ppriv -v $$
1600: pfksh
flags = <none>
E: file_link_any,...,proc_clock_highres,sys_ib_info,sys_time
I: file_link_any,...,proc_clock_highres,sys_ib_info,sys_time
P: file_link_any,...,proc_clock_highres,sys_ib_info,sys_time
L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

**EXAMPLE 18** Enabling a User to Use Own Password for Role Password

By default, users must type the role's password to assume a role. By requiring a user password, the administrator makes assuming a role in Oracle Solaris similar to assuming a role in a Linux environment.

```
# rolemod -K roleauth=user auditrev
```

To assume this role, the assigned users can now use their own passwords, not the password that was created specifically for the role.

If the user has been assigned other roles, the user's password authenticates to those roles, too.

**EXAMPLE 19** Creating a Rights Profile for Administrators of a Third-Party Application

Typically, administrators of third-party applications are not given root privileges. If the application uses commands that are not included in an Oracle Solaris rights profile, the root administrator should create a rights profile for the application administrators. This example shows a rights profile for an application whose user management commands rely on Oracle Solaris commands in the User Management and User Security rights profiles. The application also uses the mv, rm, and cp commands in ways that require privilege.

To determine which privileges the cp, mv, and rm commands required, the administrators tested the application as described in [“How to Determine Which Privileges a Program Requires” on page 129](#).

```
# profiles -p "Application User Management"
profiles:Application User Management> set desc="Application User Management Profile"
profiles:Application User Management> add profiles="User Management,User Security"
profiles:Application User Management> add cmd=/usr/bin/cp
profiles:Application User Management:cp> set privs=zone
profiles:Application User Management:cp> end
profiles:Application User Management> add cmd=/usr/bin/mv
profiles:Application User Management:mv> set privs=zone
profiles:Application User Management:mv> end
profiles:Application User Management> add cmd=/usr/bin/rm
profiles:Application User Management:rm> set privs=zone
profiles:Application User Management:rm> end
profiles:Application User Management> add cmd=/opt/ Appl/bin/addusr
profiles:Application User Management:addusr> set privs=zone
profiles:Application User Management:addusr> end
profiles:Application User Management> add cmd=/opt/ Appl/bin/addgrp
profiles:Application User Management:addgrp> set privs=zone
profiles:Application User Management:addgrp> end
profiles:Application User Management> commit
profiles:Application User Management> info
    name=Application User Management
    desc=Application User Management Profile
    profiles=User Management,User Security
    cmd=/usr/bin/cp
    ...
profiles:Application Administrator> exit
```

**EXAMPLE 20** Modifying a Rights Profile to Enable a User to Use Own Password for Role Password

```
# profiles -p "Local System Administrator"
profiles:Local System Administrator> set roleauth="user"
profiles:Local System Administrator> end
```

```
profiles:Local System Administrator> exit
```

When a user who is assigned the Local System Administrator rights profile wants to assume the role, the user is prompted for a password. In the following sequence, the role name is admin:

```
$ su - admin
Password: xxxxxxxx
$    /** You are now in a profile shell with administrative rights**/
```

**EXAMPLE 21** Changing the Value of roleauth for a Role in the LDAP Repository

In this example, the root role enables all users who can assume the role secadmin to use their own password when assuming a role. This ability is granted to these users for all systems that are managed by the LDAP server.

```
# rolemod -S ldap -K roleauth=user secadmin
```

**EXAMPLE 22** Enabling a Trusted User to Read Extended Accounting Files

You can enable a trusted user or group of users to read a file that is owned by the root account. This right can be useful to users who can run an administrative application that includes a root-owned file. This example adds one or more Perl scripts to the Extended Accounting Net Management rights profile.

After assuming the root role, the administrator creates a rights profile that adds the ability to read accounting files whose names begin with network.

The following profile uses extended privilege policy to grant the file\_dac\_read privilege to a script which can then access /var/adm/exacct/network\* files only. This profile adds the existing Extended Accounting Net Management rights profile as a supplementary profile.

```
# profiles -p "Extended Accounting Perl Scripts"
profiles:Extended Accounting Perl Scripts >
set desc="Perl Scripts for Extended Accounting"
... Scripts> add profiles="Extended Accounting Net Management"
... Scripts> add cmd=/usr/local/bin/exacctdisp.pl
... Scripts:exacctdisp.pl> set privs={file_dac_read}:/var/adm/exacct/network*
... Scripts:exacctdisp.pl> end
... Scripts> commit
... Scripts> exit
```

For sample scripts, see [“Using the Perl Interface to libexacct” in Administering Resource Management in Oracle Solaris 11.3](#).



After reviewing the rights profile entries for errors such as typographical errors, omissions, or repetition, the administrator assigns the Extended Accounting Perl Scripts rights profile to a role or a user.

```
# profiles -p "Extended Accounting Perl Scripts" info
Found profile in files repository.
name=Extended Accounting Perl Scripts
desc=Perl Scripts for Extended Accounting
profiles=Extended Accounting Net Management
cmd=/usr/local/bin/exacctdisp.pl
privs={file_dac_read}:/var/adm/exacct/network*

# rolemod -K profiles+="Extended Accounting Perl Scripts" rolename
# usermod -s /usr/bin/pfbash -K profiles+="Extended Accounting Perl Scripts" username
```

**EXAMPLE 23** Enabling a Non-root Account to Read a root-Owned File

In this example, the administrator creates a rights profile that uses extended privilege policy to enable authorized users and roles to read the `/var/adm/sulog` file that root owns. The administrator adds the commands that the user can use to read the file. Unlisted commands cannot be used, such as the `head` command.

```
# profiles -p "Read sulog File"
profiles:Read sulog File
set desc="Read sulog File"
... File> add profiles="Read Log Files"
... File> add cmd=/usr/bin/cat
... File:cat> set privs={file_dac_read}:/var/adm/sulog
... File:cat> end
... File> add cmd=/usr/bin/less
... File:less> set privs={file_dac_read}:/var/adm/sulog
... File:less> end
... File> add cmd=/usr/bin/more
... File:more> set privs={file_dac_read}:/var/adm/sulog
... File:more> end
... File> add cmd=/usr/bin/page
... File:page> set privs={file_dac_read}:/var/adm/sulog
... File:page> end
... File> add cmd=/usr/bin/tail
... File:tail> set privs={file_dac_read}:/var/adm/sulog
... File:tail> end
... File> add cmd=/usr/bin/view
... File:head> set privs={file_dac_read}:/var/adm/sulog
... File:head> end
... File> commit
... File> exit
```

The `view` command enables the user to read a file but not to edit it.

## Restricting Users' Rights

The procedures and examples in this section restrict login attempts, limit the rights of regular users, or remove some administrative rights from an administrator. They show how to modify users, roles, and rights profiles. For information about rights, see [Chapter 1, “About Using Rights to Control Users and Processes”](#).

- Require users to supply a one-time password (OTP) – [“Task Map: Using OTP in Oracle Solaris” in \*Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3\*](#)
- Provide stronger default file permissions for a user – [“How to Set a More Restrictive `umask` Value for Regular Users” on page 67](#)
- Limit consecutive unsuccessful login attempts – [“How to Set Account Locking for Regular Users” on page 68](#)
- Prevent user processes from spawning subprocesses – [Example 26, “Preventing Guests From Spawning Editor Subprocesses,” on page 75](#)
- Remove limit privileges from a user – [Example 24, “Removing Privileges From a User's Limit Set,” on page 74](#)
- Remove basic privileges from your own shell process – [Example 25, “Removing a Basic Privilege From Yourself,” on page 74](#)
- Prevent user processes from spawning subprocesses – [Example 26, “Preventing Guests From Spawning Editor Subprocesses,” on page 75](#)
- Create a restricted editor for guests – [Example 26, “Preventing Guests From Spawning Editor Subprocesses,” on page 75](#)
- Assign the restricted editor to a public system – [Example 27, “Assigning the Editor Restrictions Rights Profile to All Users,” on page 76](#)
- Remove basic privileges from a user – [Example 29, “Creating a Remote Users Rights Profile,” on page 77](#)
- Create a rights profile for remote users – [Example 29, “Creating a Remote Users Rights Profile,” on page 77](#)
- Remove privileges from the limit set of a rights profile – [Example 30, “Removing Basic Privileges From a Rights Profile,” on page 78](#)
- Remove rights from a rights profile – [Example 30, “Removing Basic Privileges From a Rights Profile,” on page 78](#), [Example 49, “Cloning and Removing Selected Rights From a Rights Profile,” on page 106](#)
- Restrict an administrator to explicitly assigned rights – [Example 31, “Restricting an Administrator to Explicitly Assigned Rights,” on page 78](#)

- Prevent applications from creating subprocesses – [Example 32, “Preventing Selected Applications From Spawning New Processes,”](#) on page 79
- Remove rights from all users of a system – [Example 27, “Assigning the Editor Restrictions Rights Profile to All Users,”](#) on page 76, [Example 28, “Modifying the `policy.conf` File to Limit the Rights Available to System Users,”](#) on page 77
- Create a system for restricted use – [Example 28, “Modifying the `policy.conf` File to Limit the Rights Available to System Users,”](#) on page 77
- Qualify attributes in LDAP by user, role, system, or set of systems – [Example 33, “Qualifying Where and When LDAP Users and Roles Can Use Their Rights,”](#) on page 79 and the `user_attr(4)` man page
- Limit user access to system by time or location – [Example 33, “Qualifying Where and When LDAP Users and Roles Can Use Their Rights,”](#) on page 79
- Remove an authorization from a user – [Example 50, “Testing a New Authorization,”](#) on page 108
- Remove a role assignment from a user – [Example 53, “Preventing the `root` Role From Being Used to Maintain a System,”](#) on page 111

## Setting General User Restrictions

In this section, you change the default `umask` value for all users, prevent malicious login attempts by limiting failed logins, and remove the ability of console users to shut down the system. You can limit failed login attempts per system, per user, or through a rights profile. For a discussion of password constraints, see [“Passwords and Password Policy”](#) in *Oracle Solaris 11.3 Security and Hardening Guidelines*.

### ▼ How to Set a More Restrictive `umask` Value for Regular Users

The `umask` utility sets the file permission bits of user-created files. If the default `umask` value, `022`, is not restrictive enough, set a more restrictive mask by using this procedure.

**Before You Begin** You must become an administrator who is authorized to edit the skeleton files. The `root` role is assigned these authorizations. For more information, see [“Using Your Assigned Administrative Rights”](#) on page 98.

1. **View the sample files that Oracle Solaris provides for user shell defaults.**

```
# ls -la /etc/skel
.bashrc
.profile
```

```
local.cshrc
local.login
local.profile
```

**2. Set the `umask` value in the `/etc/skel` files that you are going to assign to users.**

Choose one of the following values:

- `umask 026` – Provides moderate file protection  
(751) – r for group, x for others
- `umask 027` – Provides strict file protection  
(750) – r for group, no access for others
- `umask 077` – Provides complete file protection  
(700) – No access for group or others

**See Also** For more information, see the following:

- [“Default umask Value” in \*Securing Files and Verifying File Integrity in Oracle Solaris 11.3\*](#)
- Selected man pages include [`useradd\(1M\)`](#) and [`umask\(1\)`](#).

## ▼ How to Set Account Locking for Regular Users

Use this procedure to lock regular user accounts after a certain number of failed login attempts. You can set account locking per user, per system, or for selected users through a rights profile.

---

**Note** - Roles are shared accounts. Do not set account locking for users who can assume roles or roles because one locked user can lock out the role.

---

**Before You Begin** Do not set this protection system-wide on a system that you use for administrative activities. Rather, monitor the administrative system for unusual use and keep it available for administrators.

You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

**1. Set the `LOCK_AFTER_RETRIES` security attribute to YES.**

Choose the scope of the attribute value.

- **Set system-wide.**

This protection applies to any user who attempts to use the system.

```
# pfedit /etc/security/policy.conf
...
#LOCK_AFTER_RETRIES=NO
LOCK_AFTER_RETRIES=YES
...
```

- **Set per user.**

This protection applies only to the user for whom you run this command. If you have many users, this is not a scalable solution.

```
# usermod -K lock_after_retries=yes username
```

- **Create and assign a rights profile.**

This protection applies to any user or system where you assign this rights profile.

- a. **Create the rights profile.**

```
# profiles -p shared-profile -S ldap
shared-profile: set lock_after_retries=yes
...
```

For more information about creating rights profiles, see [“Creating Rights Profiles and Authorizations” on page 103](#).

- b. **Assign the rights profile to users or system-wide.**

If you have many users that share a rights profile, setting this value in a rights profile can be a scalable solution.

```
# usermod -P shared-profile username
```

You can also assign the profile per system in the `policy.conf` file.

```
# pfedit /etc/security/policy.conf
...
#PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=shared-profile,Basic Solaris User
```

2. **Set the RETRIES security attribute to 3.**

Choose the scope of the attribute value.

- **Set system-wide.**

```
# pfedit /etc/default/login
...
```

```
#RETRIES=5
RETRIES=3
...
```

- **Set per user.**

```
# usermod -K lock_after_retries=3 username
```

- **Create and assign a rights profile.**

Review [“Creating Rights Profiles and Authorizations” on page 103](#) and create a rights profile that includes `lock_after_retries=3`.

- 3. To unlock a locked user, use the `passwd` command.**

```
# passwd -u username
```

A user who is locked out cannot log in without administrative intervention. You can unlock user accounts in both the `files` and `ldap` naming services.

- See Also**
- For a discussion of user and role security attributes, see [Chapter 8, “Reference for Oracle Solaris Rights”](#).
  - Selected man pages include [`passwd\(1\)`](#), [`policy.conf\(4\)`](#), [`profiles\(1\)`](#), [`user\_attr\(4\)`](#), and [`usermod\(1M\)`](#).

## ▼ How to Remove Power Management Capability From Users

Use this procedure to prevent users on the console of a system from suspending the system or powering it down. This software solution is not effective if the system hardware can be unplugged by the console user.

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

- 1. Review the contents of the Console User rights profile.**

```
$ profiles -p "Console User" info
name=Console User
desc=Manage System as the Console User
auths=solaris.system.shutdown,solaris.device.cdrw,
      solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
profiles=Suspend To RAM,Suspend To Disk,Brightness,CPU Power Management,
      Network Autoconf User
```

2. **Create a rights profile that includes any rights in the Console User profile that you want users to retain.**

For instructions, see [“How to Create a Rights Profile” on page 103](#).

The following is a sample replacement profile:

```
$ profiles -p "Site Console User" info
name=Site Console User
desc=Read devices and control brightness on the console
auths=solaris.device.cdrw,
profiles=Brightness,CPU Power Management,Network Autoconf User
```

3. **Comment out the Console User rights profile in the `/etc/security/policy.conf` file.**

```
#CONSOLE_USER=Console User
```

4. **Assign the rights profile that you created in [Step 2](#).**

- Assign this rights profile to users.

```
# usermod -P shared-profile username
```

- You can also assign the profile per system in the `policy.conf` file.

```
# pfedit /etc/security/policy.conf...
#PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=shared-profile,Basic Solaris User
```

**See Also** For more information, see [“policy.conf File” on page 141](#) and the [policy.conf\(4\)](#) and [usermod\(1M\)](#) man pages.

## Setting Remote Login Restrictions

This section describes several ways to prevent remote login for specific non-root users. To prevent remote login by root, review the CONSOLE variable value description in the `/etc/default/login` file and [“Securing Logins and Passwords” in \*Securing Systems and Attached Devices in Oracle Solaris 11.3\*](#).

Oracle Solaris provides several ways for you to prevent remote logins for specific non-root users.

- You can specify the days and times when users can access a PAM service such as ssh. For example, use the `access_times` security attribute to prevent `jdoe` from using Secure Shell:

```
# usermod -K access_times={ssh}:A10000-0000 jdoe
```

The value indicates that for All Days (A1) no access times are available. For more information, see the [user\\_attr\(4\)](#) man page.

One advantage of using RBAC security attributes is that you can use the `-S ldap` option to the user and role configuration commands to maintain the restriction in LDAP. Another advantage is that you can specify the service names, days, and times when access is allowed.

- You can add specific users to the DenyUsers property in Secure Shell. For more information, see the [ssh\\_config\(5\)](#) man page.
- You can customize a PAM stack to include the `pam_deny` module, and assign the customized PAM stack to specific users by using the `pam_policy` security attribute to the user and role configuration commands. For more information, see “[Configuring PAM](#)” in *Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3* and the [pam\\_deny\(5\)](#) and [pam\\_user\\_policy\(5\)](#) man pages.

## Removing Privileges From Users

Under particular circumstances, privileges can be removed from a regular or guest user. For example, you might prevent remote users from examining the status of processes that they do not own, individual users might do the same to highlight their own processes, and you might prevent guests from using too many resources.

### ▼ How to Remove Unneeded Basic Privileges From Users

Under particular circumstances, some basic privileges can be removed from a regular or guest user's basic set. For example, remote users might be prevented from examining the status of processes that they do not own.

**Before You Begin** You must assume the root role. For more information, see “[Using Your Assigned Administrative Rights](#)” on page 98.

#### 1. List a full definition of the basic privilege set.

The following three basic privileges are likely candidates for removal.

```
$ ppriv -lv basic
file_link_any
  Allows a process to create hardlinks to files owned by a uid
  different from the process' effective uid.
...
proc_info
```



```

Allows a process to examine the status of processes other
than those it can send signals to. Processes which cannot
be examined cannot be seen in /proc and appear not to exist.
proc_session
Allows a process to send signals or trace processes outside its
session.
...

```

## 2. Choose the scope of the privilege removal.

### ■ Set system-wide.

Any user who attempts to use the system is denied these privileges. This method of privilege removal might be appropriate for a publicly available computer.

```

# pfedit /etc/security/policy.conf
...
#PRIV_DEFAULT=basic
PRIV_DEFAULT=basic,!file_link_any,!proc_info,!proc_session

```

### ■ Remove privileges from individual users.

#### ■ Prevent a user from linking to a file that the user does not own.

```
# usermod -K 'defaultpriv=basic,!file_link_any' user
```

#### ■ Prevent a user from examining processes that the user does not own.

```
# usermod -K 'defaultpriv=basic,!proc_info' user
```

#### ■ Prevent a user from starting a second session, such as starting an ssh session from the user's current session.

```
# usermod -K 'defaultpriv=basic,!proc_session' user
```

#### ■ Remove all three privileges from a user's basic set.

```
# usermod -K 'defaultpriv=basic,!file_link_any,!proc_info,!proc_session' user
```

### ■ Create and assign a rights profile.

This protection applies to any user or system where you assign this rights profile.

#### a. Create the rights profile.

```

# profiles -p shared-profile -S ldap
shared-profile: set defaultpriv=basic,!file_link_any,!proc_info,!proc_session

```

...

For more information about creating rights profiles, see [“Creating Rights Profiles and Authorizations” on page 103](#).

**b. Assign the rights profile to users or system-wide.**

If you have many users that share a rights profile, such as remote users, setting this value in a rights profile can be a scalable solution.

```
# usermod -P shared-profile username
```

You can also assign the profile per system in the `policy.conf` file.

```
# pfedit /etc/security/policy.conf
...
#PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=shared-profile,Basic Solaris User
```

**Example 24** Removing Privileges From a User's Limit Set

In the following example, all sessions that originate from `jdoue`'s initial login are prevented from using the `sys_linkdir` privilege. The user cannot make hard links to directories or unlink directories even after running the `su` command.

```
# usermod -K 'limitpriv=all,!sys_linkdir' jdoue
# userattr limitpriv jdoue
all,!sys_linkdir
```

**Example 25** Removing a Basic Privilege From Yourself

In the following example, a regular user modifies `.bash_profile` to remove the `proc_info` basic privilege. The output of programs like `ps` and `prstat` contain only the user's own processes, which can highlight useful information.

```
## .bash_profile
## Remove proc_info privilege from my shell
##
ppriv -s EI-proc_info $$
```

The `ppriv` line removes the `proc_info` privilege from the user's effective and inheritable privilege sets (`EI-`) in the current shell process (`$$`).

In the following `prstat` output, the totals shrink from 74 to three processes:

```
## With all basic privileges
```

```
Total: 74 processes, 527 lwps, load averages: 0.01, 0.00, 0.00
```

```
## With proc_info removed from the effective and inheritable set
Total: 3 processes, 3 lwps, load averages: 0.00, 0.00, 0.00
```

### Example 26 Preventing Guests From Spawning Editor Subprocesses

In this example, the administrator prevents users from creating subshells from one or more editors by removing the `proc_exec` basic privilege from the editor command.

1. The administrator creates a rights profile that removes `proc_exec` from the limit privilege set of the `vim` editor.

```
# profiles -p -S ldap "Editor Restrictions"
profiles:Editor Restrictions> set desc="Site Editor Restrictions"
... Restrictions> add cmd=/usr/bin/vim
... Restrictions:vim> set limitprivs=all,!proc_exec
... Restrictions:vim> end
... Restrictions> commit
... Restrictions> exit
```

2. The administrator adds other popular editors to the rights profile.

```
# profiles -p "Editor Restrictions"
profiles:Editor Restrictions> add cmd=/usr/bin/gedit
... Restrictions:gedit> set limitprivs=all,!proc_exec
... Restrictions:gedit> end
... Restrictions> add cmd=/usr/bin/vim
... Restrictions:vim> set limitprivs=all,!proc_exec
... Restrictions:vim> end
... Restrictions> add cmd=/usr/xpg4/ed
... Restrictions:ed> set limitprivs=all,!proc_exec
... Restrictions:ed> end
... Restrictions> add cmd=/usr/xpg4/ex
... Restrictions:ex> set limitprivs=all,!proc_exec
... Restrictions:ex> end
... Restrictions> add cmd=/usr/xpg4/vi
... Restrictions:vi> set limitprivs=all,!proc_exec
... Restrictions:vi> end
... Restrictions> commit
... Restrictions> exit
```

3. The administrator reviews the rights profile entries for errors such as typographical errors, omissions, or repetition.

```
# profiles -p "Editor Restrictions" info
```

```
Found profile in files repository.  
name=Editor Restrictions  
desc=Site Editor Restrictions  
cmd=/usr/bin/vim  
limitprivs=all,!proc_exec  
...
```

4. The administrator assigns the Editor Restrictions rights profile to the guest user.

```
# usermod -K profiles+="Editor Restrictions" guest
```

By using `profiles+`, the administrator adds this rights profile to the account's current rights profiles.

5. To verify that the editor privileges are limited, the administrator opens the editor and in a separate window, examines the privileges on the editor process.

```
# ppriv -S $(pgrep vim)  
2805:  vim .bash_profile  
flags = PRIV_PFEEXEC      User is running a profile shell  
E: basic,!proc_info      proc_info is removed from basic set  
I: basic,!proc_info  
P: basic,!proc_info  
L: all,!proc_exec        proc_exec is removed from limit set
```

**Example 27** Assigning the Editor Restrictions Rights Profile to All Users

In this example, the administrator adds the Editor Restrictions rights profile to the `policy.conf` file. The administrator ensures that this file is distributed to all public systems where guests can log in.

```
# cd /etc/security; cp policy.conf policy.conf.orig  
# pfedit /etc/security/policy.conf  
...  
AUTHS_GRANTED=  
AUTH_PROFS_GRANTED=  
#PROFS_GRANTED=Basic Solaris User  
PROFS_GRANTED=Editor Restrictions,Basic Solaris User
```

The User Security administrator has assigned a [profile shell](#) to every user. For the reasons and the procedure, see [“Assigning Rights to Users” on page 47](#).

**See Also** For more information, see the [privileges\(5\)](#) man page.

## Removing Rights From Many Users by Using a Rights Profile

Rights profiles can limit rights for a large number of users. They are easily maintained because after you modify the rights profile, users who are assigned the profile get the modified rights at next login.

## Removing Rights System-Wide

**EXAMPLE 28** Modifying the `policy.conf` File to Limit the Rights Available to System Users

In this example, the administrator creates a system that is useful only to administer the network. The administrator removes the Basic Solaris User rights profile and any authorizations from the `/etc/security/policy.conf` file. The Console User rights profile is not removed. The affected lines in the resulting `policy.conf` file are the following:

```
...
##AUTHS_GRANTED=
##AUTH_PROFS_GRANTED=
##PROFS_GRANTED=Basic Solaris User
CONSOLE_USER=Console User
...
```

Only a user who has been explicitly assigned authorizations, commands, or rights profiles is able to use this system. After login, the authorized user can perform administrative duties. If the authorized user is sitting at the system console, the user has the rights of the Console User.

**EXAMPLE 29** Creating a Remote Users Rights Profile

In this example, the administrator creates a rights profile in the LDAP repository for users who log in remotely.

```
# profiles -p -S LDAP "Remote Users"
profiles:Remote Users> set desc="For all logins from remote systems"
... Remote Users> set defaultpriv="basic,!proc_info"
... Remote Users> set limitpriv="basic,!proc_info"
... Remote Users> end
... Remote Users> exit
```

The administrator verifies the contents.

```
# profiles -p "Remote Users" info
Found profile in LDAP repository.
    name=Remote Users
    desc=For all logins from remote systems
    defaultpriv=basic,!proc_info,
    limitpriv=basic,!proc_info
```

**EXAMPLE 30** Removing Basic Privileges From a Rights Profile

In the following example, after thorough testing, the security administrator removes another basic privilege from the Remote Users rights profile created in [Example 29, “Creating a Remote Users Rights Profile,” on page 77](#). Users who are assigned this profile cannot examine any processes outside their current session, and they cannot add another session.

```
# profiles -p "Remote Users"
profiles:Remote Users> set defaultpriv="basic,!proc_info,!proc_session"
profiles:Remote Users> end
profiles:Remote Users> exit
```

**EXAMPLE 31** Restricting an Administrator to Explicitly Assigned Rights

You can restrict a role or user to a limited number of administrative actions in two ways.

- Assign the Stop rights profile as the last profile in the user's list of profiles. The Stop rights profile is the simplest way to create a restricted shell. The authorizations and rights profiles in the `policy.conf` file are not assigned to the user or role.
- Modify the `policy.conf` file on a system, and require the role or user to use that system for administrative tasks. See [Example 28, “Modifying the `policy.conf` File to Limit the Rights Available to System Users,” on page 77](#).

The following command limits the `auditrev` role to performing only audit reviews.

```
# rolemod -K profiles="Audit Review,Stop" auditrev
```

Because the `auditrev` role does not have the Console User rights profile, the auditor cannot shut down the system. Because this role does not have the `solaris.device.cdrw` authorization, the auditor cannot read from or write to the CD-ROM drive. Because this role does not have the Basic Solaris User rights profile, no commands from that profile can be run in this role. Because the All rights profile is not assigned, the `ls` command will not run. The role uses the File Browser to select the audit files for review.

For more information, see [“Order of Search for Assigned Rights” on page 38](#) and [“Rights Profiles Reference” on page 135](#).

**EXAMPLE 32** Preventing Selected Applications From Spawning New Processes

In this example, the administrator creates a rights profile for applications that do not require subprocesses for correct operation. For convenience, the administrator creates a directory to hold these executables. When new applications are added that do not require subprocesses, the executables can be added to this directory. Or, if the executable is required to be in a specific directory, the administrator can link to it from `/opt/local/noex/app-executable`.

```
# profiles -p "Prevent App Subprocess"
profiles:Prevent App Subprocess> set desc="Keep apps from execing processes"
profiles:Prevent App Subprocess> add cmd=/opt/local/noex/mkmod
... Subprocess:mkmod> set limitprivs=all,!proc_exec
... Subprocess:mkmod> end
... Subprocess> add cmd=/opt/local/noex/gomap
... Subprocess:gomap> set limitprivs=all,!proc_exec
... Subprocess:gomap> end
... Subprocess> commit
... Subprocess> exit
```

## Qualifying Accounts to Assume Rights on Specific Systems

This set of examples illustrates how to centrally manage the assignment of security attributes to users and roles. These commands work only in the LDAP naming service, not in the files naming service.

**EXAMPLE 33** Qualifying Where and When LDAP Users and Roles Can Use Their Rights

The following example enables the user `jdoe` to administer the systems `labsys1` and `labsys2`. `jdoe` is an LDAP account.

```
# usermod -q labsys1 -K profiles="System Administrator" jdoe
# usermod -q labsys2 -K profiles="System Administrator" jdoe
```

The following example limits administrative access to the role `admin` on `system1` to weekdays from 5am to 3pm. `admin` is an LDAP account. The system's local time zone is used.

```
# rolemod -q system1 -k access_times="(*):Wk0500-1500" \
-K profiles="System Administrator" admin
```

**EXAMPLE 34** Qualifying the Systems Where Users and Roles Have Administrative Rights

This set of examples illustrates how to qualify the assignment of security attributes by hostname or by group of hosts called *netgroups*. See the [netgroup\(4\)](#) man page.

The following example enables the user `jdoe` to administer a set of systems defined as the `lab1` netgroup. `jdoe` and the `lab1` netgroup are managed in the LDAP directory.

```
# usermod -q @lab1 -K profiles="System Administrator" jdoe
```

The following example limits the user `jdoe` to administering the `lab1` netgroup to weekdays from 5am to 3pm.

```
# usermod -q @lab1 -k access_times="(*):Wk0500-1500" -K profiles="System Administrator" jdoe
```



# ◆◆◆ CHAPTER 4

## Assigning Rights to Applications, Scripts, and Resources

---

This chapter covers tasks that apply privileges, extended privilege policy, and other rights to users, ports, and applications:

- [“Assigning Rights to Applications and Scripts” on page 81](#)
- [“Locking Down Resources by Using Extended Privileges” on page 85](#)
- [“Users Locking Down the Applications That They Run” on page 92](#)
- [“Administering Immutable Zones” on page 95](#)

For an overview of rights, see [“User Rights Management” on page 16](#).

### Limiting Applications, Scripts, and Resources to Specific Rights

The tasks and examples in this section assign privileges to executables and system resources. Typically, you assign a privilege to an executable to enable a trusted user to run that executable. In [“Assigning Rights to Applications and Scripts” on page 81](#), the privilege assignment enables the application or script to be run by a trusted user in a [profile shell](#). In [“Locking Down Resources by Using Extended Privileges” on page 85](#), extended privilege policy limits a user ID, port, or file object, to a smaller set of privileges than the default effective set. Privileges that are unspecified are denied to that user's process, port, or object. Such an assignment approximates [least privilege](#) policy.

### Assigning Rights to Applications and Scripts

Applications and scripts execute one command or a series of commands. To assign rights, you set the [security attributes](#), such as set IDs or privileges, for each command in a [rights profile](#). Applications can check for authorizations, if appropriate.

---

**Note** - 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. When the script is executed in a profile shell, the command runs with the security attributes.

---

- Run a script that needs rights – [“How to Run a Shell Script With Privileged Commands” on page 82](#)
- Batch edit sensitive files – [Example 38, “Scripting the Batch Editing of Files in a Directory,” on page 84](#)
- Enable [privilege-aware](#) applications to be run by non-root users – [Example 35, “Assigning Security Attributes to a Legacy Application,” on page 83](#)
- Enable root-owned applications to be run by non-root users – [Example 36, “Running an Application With Assigned Rights,” on page 83](#)
- Check for authorizations in a script – [Example 37, “Checking for Authorizations in a Script or Program,” on page 84](#)

## ▼ How to Run a Shell Script With Privileged Commands

To run a privileged shell script, you add privileges to the script and to the commands in the script. Then, the appropriate rights profile must contain the commands with privileges assigned to them.

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

1. **Create the script with `/bin/pfsh`, or any other profile shell, on the first line.**

```
#!/bin/pfsh
# Copyright (c) 2015 by Oracle
```

2. **As a regular user, determine the privileges that the commands in the script need.**

By running the script with no privileges, the `debug` option to the `ppriv` command lists the missing privileges.

```
$ ppriv -eD script-full-path
```

For more information, see [“How to Determine Which Privileges a Program Requires” on page 129](#).

3. **Create or modify a rights profile for the script.**

Add the shell script, and the commands in the shell script, with their required security attributes to the rights profile. See [“How to Create a Rights Profile” on page 103](#).

**4. Assign the rights profile to a trusted user or role.**

For examples, see [“Assigning Rights to Users” on page 47](#).

**5. To run the script, do one of the following:**

- **If you are assigned the script as a user, open a profile shell and run the script.**

```
$ pfexec script-full-path
```

- **If you are assigned the script as a role, assume the role and run the script.**

```
$ su - rolename
Password: xxxxxxxx
# script-full-path
```

**Example 35** Assigning Security Attributes to a Legacy Application

Because a legacy application is not privilege-aware, the administrator assigns the `eid=0` security attribute to the application executable in a rights profile. Then, the administrator assigns it to a trusted user.

```
# profiles -p LegacyApp
profiles:LegacyApp> set desc="Legacy application"
profiles:LegacyApp> add cmd=/opt/legacy-app/bin/legacy-cmd
profiles:LegacyApp:legacy-cmd> set eid=0
profiles:LegacyApp:legacy-cmd> end
profiles:LegacyApp> exit
# profiles -p LegacyApp 'select cmd=/opt/legacy-app/bin/legacy-cmd;info;end'
  id=/opt/legacy-app/bin/legacy-cmd
  eid=0

# usermod -K profiles+="Legacy application" jdoe
```

**Example 36** Running an Application With Assigned Rights

In this example, the administrator assigns the rights profile from [Example 47, “Creating a Rights Profile That Includes Privileged Commands,” on page 104](#) to a trusted user. The user must provide a password when executing the script.

```
# usermod -K auth_profiles+="Site application" jdoe
```

**Example 37** Checking for Authorizations in a Script or Program

To check for authorizations, write 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 whether 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
```

A more complete test includes logic that checks for the use of wildcards. For example, to test whether the user has the `solaris.system.date` authorization, you would need to check for the following strings:

- `solaris.system.date`
- `solaris.system.*`
- `solaris.*`

If you are writing a program, use the function `getauthattr()` to test for the authorization.

**Example 38** Scripting the Batch Editing of Files in a Directory

This example shows how to batch edit sensitive files by using two scripts. The first script switches out the default editor and calls the second script. The second script is the editing script. By changing the second script, you keep a record of the changes to the files.

1. Create the first script.

```
# pfedit batchpfedit.sh
#!/usr/bin/bash
OLDEDITOR=$EDITOR
export EDITOR=~/.bin/key-edit.sh
pfedit /var/ITdemos/firstdemo-directory
export EDITOR=$OLDEDITOR
```

where:

`OLDEDITOR=$EDITOR` Defines the `OLDEDITOR` variable to hold the value of the current editor.

`~/bin/key-edit.sh` Is the script that contains the edits.

`pfedit /var/ITdemos/firstdemo-directory` Is the command that creates a temporary `pfedit` file for the script to use.

`export EDITOR=$OLDEDITOR` Replaces the original `$EDITOR` definition.

2. Create the editing script.

```
# pfedit batchpfedit.sh
#!/usr/bin/bash
OLDEDITOR=$EDITOR
export EDITOR=/bin/key-edit.sh
pfedit /var/ITdemos/firstdemo-directory
export EDITOR=$OLDEDITOR
```

The second script contains the editing changes. In this example, the `key-edit.sh` script substitutes the word `key` for the word `pey`.

```
#!/usr/bin/bash
perl -pi.pfedit -e 's/pey/key/g' $1
rm $1.pfedit
```

- The `$1` finds the temporary file.
- The `perl -pi.extension` creates a temporary file and writes back to the original file.
- The `rm` command removes this extra temporary file.

3. Call the first script.

You can create more editing scripts, then replace the `key-edit.sh` script in the `batchpfedit.sh` with the new script name.

## Locking Down Resources by Using Extended Privileges

Extended privilege policy can limit attacker access to a system when an attack on an application is successful. An extended policy rule limits the scope of the effect of a privilege assignment to just the resource that is in the rule. Extended policy rules are expressed by enclosing the privileges between curly braces, followed by a colon and the associated resource. For more discussion, see [“Expanding a User or Role's Privileges” on page 33](#). For examples of the syntax, see the `ppriv(1)` and `privileges(5)` man pages.

Both administrators and regular users can lock down resources by using extended privileges. Administrators can create extended privilege rules for users, ports, and applications. Regular users can use the command line or write scripts that use the `ppriv -r` command to prevent applications from writing files outside of user-specified directories.

- Limit the access available to a malicious user who enters by a port – [“How to Apply Extended Privilege Policy to a Port” on page 86](#)
- Run a database as a non-root daemon – [“How to Lock Down the MySQL Service” on page 87](#)
- Run the Apache HTTP Server as a non-root daemon – [“How to Assign Specific Privileges to the Apache HTTP Server” on page 90](#)
- Verify that the Apache HTTP Server is running with privileges – [“How to Determine Which Privileges the Apache HTTP Server Is Using” on page 91](#)
- Prevent Firefox from writing to directories on your system – [Example 39, “Running a Browser in a Protected Environment,” on page 93](#)
- Limit your applications to specific directories on your system – [Example 40, “Protecting Directories on Your System From Application Processes,” on page 94](#)

## ▼ How to Apply Extended Privilege Policy to a Port

The service for the Network Time Protocol (NTP) uses the privileged port 123 for udp traffic. Privileges are required for this service to run. This example procedure modifies the service manifest to protect other ports from being accessed by a malicious user who might gain the privileges that are assigned to this port.

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

### 1. Read the default service manifest entry for the port.

From the following `/lib/svc/manifest/network/ntp.xml` start method entry, the `net_privaddr`, `proc_lock_memory`, and `sys_time` privileges could be used on other processes:

```
privileges='basic,!file_link_any,!proc_info,!proc_session,net_privaddr,  
proc_lock_memory,sys_time'
```

The removed privileges specified by `!file_link_any`, `!proc_info`, `!proc_session` prevent the service from signaling or observing any other processes, and from creating hard links as a way of renaming files. That is, the process that is started by the service is only able to bind to NTP's port 123, not to any of the other privileged ports.

If a hacker could exploit the service to start another process, that process would be similarly limited.

2. **Modify the start and restart methods to limit the `net_privaddr` privilege to this port only.**

```
# svccfg -s ntp editprop -a
```

- a. **Search for the string `net_privaddr`.**
- b. **Uncomment the entries that contain `net_privaddr`.**
- c. **In both entries, replace `net_privaddr` with `{net_privaddr}:123/udp`.**

The extended privilege policy removes all privileges from this service except the specified privileges plus the basic privileges that are not specified. Therefore, the set of over eighty potentially exploitable privileges is reduced to less than eight.

3. **Restart the service to use the extended privilege policy.**

```
# svcadm restart ntp
```

4. **Verify that the service is using extended privilege.**

```
# svcprop -s ntp | grep privileges
start/privileges    astring  basic,!file_link_any,!proc_info,!proc_session,
                   {net_privaddr}:123/udp,proc_lock_memory,sys_time
restart/privileges  astring  basic,!file_link_any,!proc_info,!proc_session,
                   {net_privaddr}:123/udp,proc_lock_memory,sys_time
```

## ▼ How to Lock Down the MySQL Service

At installation, the MySQL database is configured to run with the full privileges of root over an unprotected port. In this task, you assign extended privilege policy to the MySQL service in a rights profile. After the rights profile becomes the exec method of the service, MySQL runs over a protected port as the user `mysql` with limited database access by non-MySQL processes.

**Before You Begin** The initial user can install the package. The remaining steps must be performed by the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

1. **Install the MySQL package.**

```
# pkg search basename:mysql
...
basename ... pkg:/database/mysql-55@version
# pfexec pkg install mysql-55
```

---

**Note** - If you upgrade to a later version of the MySQL database, then modify all steps to use the later version number rather than 5.5 and 55.

---

## 2. Display the FMRI and state of the MySQL service.

```
# svcs mysql
STATE          STIME    FMRI
disabled      May_15   svc:/application/database/mysql:version_55
```

## 3. Create a rights profile that modifies the execution method of the service.

The service manifest for this service specifies that the execution method is a shell script wrapper, `/lib/svc/method/mysql_55`.

```
# svcprop -s mysql | grep manifest
... astring    /lib/svc/manifest/application/database/mysql_55.xml
# grep exec= /lib/svc/manifest/application/database/mysql_55.xml
      exec='/lib/svc/method/mysql_55 start'
      exec='/lib/svc/method/mysql_55 stop'
```

Use the `/lib/svc/method/mysql_55` wrapper for the command in the profile.

```
$ su -
Password: xxxxxxxx
# profiles -p "MySQL Service"
MySQL Service> set desc="Locking down the MySQL Service"
MySQL Service> add cmd=/lib/svc/method/mysql_55
MySQL Service:mysql_55> set privs=basic
MySQL Service:mysql_55> add privs={net_privaddr}:3306/tcp
MySQL Service:mysql_55> add privs={file_write}:/var/mysql/5.5/data/*
MySQL Service:mysql_55> add privs={file_write}:/tmp/mysql.sock
MySQL Service:mysql_55> add privs={file_write}:/var/tmp/ib*
MySQL Service:mysql_55> end
MySQL Service> set uid=mysql
MySQL Service> set gid=mysql
MySQL Service> exit
```

The `file_write` privilege is a basic privilege granted by default to all processes. By explicitly enumerating the writable paths, write access is restricted to just those paths. This constraint applies to the specified executable and its child processes.

## 4. Make the default port for MySQL a privileged port.

```
# ipadm set-prop -p extra_priv_ports+=3306 tcp
# ipadm show-prop -p extra_priv_ports tcp
PROTO PROPERTY          PERM CURRENT    PERSISTENT  DEFAULT  POSSIBLE
tcp  extra_priv_ports      rw   2049,4045, 3306          2049,4045  1-65535
```



3306

The `net_privaddr` privilege is required to bind to a privileged port. In the case of MySQL, binding to the default port number, 3306, does not normally require this privilege.

##### 5. Assign the rights profile to the MySQL service and tell the service to use it.

```
# svccfg -s mysql:version_55
...version_55> setprop method_context/profile="MySQL Service"
...version_55> setprop method_context/use_profile=true
...version_55> refresh
...version_55> exit
```

##### 6. Enable the service.

The last component of the FMRI, `mysql:version_55`, is sufficient to uniquely specify the service.

```
# svcadm enable mysql:version_55
```

##### 7. (Optional) Verify that the service is running with the rights that are specified in the MySQL Service rights profile.

```
# ppriv $(pgrep mysql)
103697:  /usr/mysql/5.5/bin/mysqld --basedir=/usr/mysql/5.5
                                     --datadir=/var/mysql/5.5/data
flags =  PRIV_XPOLICY
Extended policies:
          {net_privaddr}:3306/tcp
          {file_write}:/var/mysql/5.5/data/*
          {file_write}:/tmp/mysql.sock
          {file_write}:/var/tmp/ib*
E: basic,!file_write
I: basic,!file_write
P: basic,!file_write
L: all
103609:  /bin/sh /usr/mysql/5.5/bin/mysqld_safe --user=mysql
                                     --datadir=/var/mysql/5.5/data
flags =  PRIV_XPOLICY
Extended policies:
          {net_privaddr}:3306/tcp
          {file_write}:/var/mysql/5.5/data/*
          {file_write}:/tmp/mysql.sock
          {file_write}:/var/tmp/ib*
E: basic,!file_write
I: basic,!file_write
P: basic,!file_write
L: all
```

## ▼ How to Assign Specific Privileges to the Apache HTTP Server

This procedure locks down the web server daemon by assigning to it only the privileges it needs. The web server can only bind to port 80, and can only write to files that the webservd daemon owns. No apache22 service processes run as root.

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

### 1. Create the web server rights profile.

```
# profiles -p "Apache2"
profiles:Apache2> set desc="Apache HTTP Server Extended Privilege"
profiles:Apache2> add cmd=/lib/svc/method/http-apache2
profiles:Apache2:http-apache2> add privs={net_privaddr}:80/tcp
...http-apache2> add privs={zone}:/system/volatile/apache2
...http-apache2> add privs={zone}:/var/apache2/2.2/logs/*
...http-apache2> add privs={zone}:/var/user
...http-apache2> add privs={file_write}:/var/user/webserv*
...http-apache2> add privs={file_write}:/tmp/*
...http-apache2> add privs={file_write}:/system/volatile/apache*
...http-apache2> add privs={file_write}:/proc/*
...http-apache2> add privs=basic,proc_priocntl
...http-apache2> set uid=webservd
...http-apache2> set gid=webservd
...http-apache2> end
---Apache2> exit
```

### 2. (Optional) If you are using the SSL kernel proxy with Apache2, you must add the SSL ports to your webservd extended policy.

```
# profiles -p "Apache2"
profiles:Apache2> add privs={net_privaddr}:443/tcp
profiles:Apache2> add privs={net_privaddr}:8443/tcp
profiles:Apache2:http-apache2> end
```

The SSL kernel proxy procedure is described in [“How to Configure an Apache 2.2 Web Server to Use the SSL Kernel Proxy” in \*Securing the Network in Oracle Solaris 11.3\*](#).

### 3. Add the rights profile to the apache22 SMF start method.

```
# svccfg -s apache22
svc:/network/http:Apache2> listprop start/exec
start/exec astring "/lib/svc/method/http-apache22 start"
...
svc:/network/http:Apache2> setprop start/profile="Apache2"
```

```

svc:/network/http:Apache2> setprop start/use_profile=true
svc:/network/http:Apache2> refresh
svc:/network/http:Apache2> exit

```

When the apache22 service is enabled, the Apache2 profile will be used.

#### 4. Enable the apache22 service.

```
# svcadm enable apache22
```

#### 5. Verify that web server is working.

Open a browser and type localhost in the Firefox URL field.

**Next Steps** To verify that the privileges are applied correctly, continue with [“How to Determine Which Privileges the Apache HTTP Server Is Using” on page 91](#).

## ▼ How to Determine Which Privileges the Apache HTTP Server Is Using

In this task, you determine which privileges the web server is using by creating a debug version of the Apache2 rights profile.

**Before You Begin** You have completed [“How to Assign Specific Privileges to the Apache HTTP Server” on page 90](#). The apache22 service is disabled. You are in the root role.

#### 1. Clone the Apache2 profile to call a different command.

Debugging a command is simpler than debugging an SMF service. The apachectl command starts the Apache service interactively.

```

# profiles -p "Apache2"
profiles:Apache2> set name="Apache-debug"
profiles:Apache-debug> sel <Tab><Tab>
profiles:Apache-debug:http-apache22> set id=/usr/apache2/2.2/bin/apachectl
profiles:Apache-debug:apachectl> end
profiles:Apache-debug> exit

```

For more information, see the apachectl(8) man page.

#### 2. Assign the cloned profile to the webservd account.

```
# usermod -K profiles+=Apache-debug webservd
```

#### 3. Switch to the webservd identity.

```
# su - webservd
```

**4. (Optional) Verify the identity.**

```
# id
uid=80(webservd) gid=80(webservd)
```

**5. Start the web service in debug mode in a profile shell.**

Do not use SMF directly. Use the command in the Apache-debug rights profile.

```
$ pfbash
# ppriv -De /usr/apache2/2.2/bin/apachectl start
```

**6. In the root role, examine the privileges of the first http daemon.**

```
# ppriv $(pgrep httpd|head -1)
2999: httpd
flags = PRIV_DEBUG|PRIV_XPOLICY|PRIV_EXEC
 5      Extended policies:
 6          {net_privaddr}:80/tcp
 7          {zone}:/system/volatile/apache2
 8          {zone}:/var/apache2/2.2/logs/*
 9          {zone}:/var/user
10          {file_write}:/var/user/webserv*
11          {file_write}:/tmp/*
12          {file_write}:/system/volatile/apache*
13          {file_write}:/proc/*
14      E: basic,!file_write,!proc_info,proc_priocntl
15      I: basic,!file_write,!proc_info,proc_priocntl
16      P: basic,!file_write,!proc_info,proc_priocntl
17      L: all
```

## Users Locking Down the Applications That They Run

Users can remove basic privileges from applications by using extended privilege policy. The policy prevents access to directories that the applications should not access.

---

**Note** - Order is important. Broader privileges for directories such as `$HOME/Download*` must be assigned after narrower privileges for most `$HOME/.*` directories.

---

**EXAMPLE 39** Running a Browser in a Protected Environment

This example illustrates how users can run the Firefox browser in a protected environment. In this configuration, the user's Documents directory is hidden from Firefox.

By using the following command, the user removes basic privileges from the `/usr/bin/firefox` command. The extended privilege arguments to the `ppriv -r` command limit the browser to reading and writing in only the directories that the user specifies. The `-e` option and its arguments open the browser with the extended privilege policy.

```
$ ppriv -r "\
{file_read}:/dev/*,\
{file_read}:/etc/*,\
{file_read}:/lib/*,\
{file_read}:/usr/*,\
{file_read}:/var/*,\
{file_read}:/proc,\
{file_read}:/proc/*,\
{file_read}:/system/volatile/*,\
{file_write}:$HOME,\
{file_read}:$HOME/*,\
{file_read,file_write}:$HOME/.mozill*,\
{file_read,file_write}:$HOME/.gnome*,\
{file_read,file_write}:$HOME/Downloa*,\
{file_read,file_write}:/tmp,\
{file_read,file_write}:/tmp/*,\
{file_read,file_write}:/var/tmp,\
{file_read,file_write}:/var/tmp/*,\
{proc_exec}:/usr/*\
" -e /usr/bin/firefox file:/// $HOME/Desktop
```

When the `file_read` and `file_write` privileges are used in an extended policy, you must grant explicit access to every file that should be read or written. The use of the wildcard character, `*`, is essential in such policies.

To handle automounted home directories, the user would add an explicit entry for the automount path, for example:

```
{file_read,file_write}:/export/home/$USER
```

If the site is not using the automount facility, the initial list of protected directories is sufficient.

Users can automate this command-line protected browser by creating a shell script. Then, to launch a browser, the user calls the script, not the `/usr/bin/firefox` command.

**EXAMPLE 40** Protecting Directories on Your System From Application Processes

In this example, a regular user creates a sandbox for applications by using a shell script wrapper. The first part of the script limits applications to certain directories. Exceptions, such as Firefox, are handled later in the script. Comments about parts of the script follow the script.

```
1 #!/bin/bash
2
3 # Using bash because ksh misinterprets extended policy syntax
4
5 PATH=/usr/bin:/usr/sbin:/usr/gnu/bin
6
7 DENY=file_read,file_write,proc_exec,proc_info
8
9 SANDBOX=""
10 {file_read}:/dev/*,\
11 {file_read}:/etc/*,\
12 {file_read}:/lib/*,\
13 {file_read,file_write}:/usr/*,\
14 {file_read}:/proc,\
15 {file_read,file_write}:/proc/*,\
16 {file_read}:/system/volatile/*,\
17 {file_read,file_write}:/tmp,\
18 {file_read,file_write}:/tmp/*,\
19 {file_read,file_write}:/var/*,\
20 {file_write}:/home/*,\
21 {file_read}:/home/*,\
22 {file_read,file_write}:/home/*,\
23 {file_read,file_write}:/home/*/*,\
24 {proc_exec}:/usr/*\
25 "
26
27 # Default program is restricted bash shell
28
29 if [[ ! -n $1 ]]; then
30     program="/usr/bin/bash --login --noprofile
31     --restricted"
32 else
33     program="$@"
34 fi
35
36 # Firefox needs more file and network access
37 if [[ "$program" =~ firefox ]]; then
38     SANDBOX+="/home/*,\
39 {file_read,file_write}:/home/*/*,\
```

```

40 {file_read,file_write}:$HOME/.mozill*,\
41 {file_read,file_write}:$HOME/.dbu*,\
42 {file_read,file_write}:$HOME/.puls*\
43 "
44
45 else
46     DENY+=" ,net_access"
47 fi
48
49 echo Starting $program in sandbox
50 ppriv -s I-$DENY -r $SANDBOX -De $program

```

The policy can be adjusted to permit specific applications more or less access. One adjustment is in lines 38-42, where Firefox is granted write access to several dot files that maintain session information in the user's home directory. Also, Firefox is not subject to line 46, which removes network access. However, Firefox is still restricted from reading arbitrary files in the user's home directory, and can save files only in its current directory.

As an extra level of protection, the default program, at line 30, is a restricted Bash shell. A restricted shell cannot change its current directory or execute the user's dot files. Therefore, any commands that are started from this shell are similarly locked into the sandbox.

In the final line of the script the `ppriv` command is passed two privilege sets as shell variables, `$DENY` and `$SANDBOX`.

The first set, `$DENY`, prevents the process from reading or writing any file, executing any subprocess, observing other user's processes, and (conditionally) accessing the network. These restrictions are too severe, so in the second set, `$SANDBOX`, the policy is refined by enumerating the directories which are available for reading, writing, and executing.

Also, in line 50 the debug option, `-D`, is specified. Access failures display in the terminal window in real time and include the named object and the corresponding privilege that is required for success. This debugging information can help the user customize the policy for other applications.

## Administering Immutable Zones

Immutable zones ensure the integrity of the files on the system. The policy of the zone specifies which files can be modified. If you have access to a terminal window in the global zone, you can change the zone to mutable, administer, then return the zone to immutable. Use the `zlogin -U` or `zoneadm` command, see [“Administering an Immutable Zone by Making It Writable” in \*Creating and Using Oracle Solaris Zones\*](#) and the `mwac(5)` man page.





## Managing the Use of Rights

---

This chapter covers tasks that maintain systems that use the rights model for administration. Several tasks extend the rights that Oracle Solaris provides by creating new rights profiles and authorizations.

The chapter covers the following topics:

- [“Using Your Assigned Administrative Rights” on page 98](#)
- [“Auditing Administrative Actions” on page 102](#)
- [“Creating Rights Profiles and Authorizations” on page 103](#)
- [“Changing Whether root Is a User or a Role” on page 109](#)

For information about rights, see [Chapter 1, “About Using Rights to Control Users and Processes”](#). For information about maintaining the assigned rights of users and roles, see [Chapter 3, “Assigning Rights in Oracle Solaris”](#).

## Managing the Use of Rights

The tasks and examples in this section describe how to use the rights that you have been assigned, and how to change the rights configuration that is provided by default.

---

**Note** - For troubleshooting assistance, see [“Troubleshooting RBAC and Privileges” on page 123](#).

---

- Use your assigned rights – [“Using Your Assigned Administrative Rights” on page 98](#)
- Audit administrative actions – [Example 45, “Using Two Roles to Configure Auditing,” on page 102](#)
- Add rights profiles and authorizations – [“Creating Rights Profiles and Authorizations” on page 103](#)
- Configure root to be a user – [“How to Change the root Role Into a User” on page 109](#)

- Change root back into a role – [Example 52, “Changing the root User Into the root Role,” on page 110](#)
- Prevent root from administering a system – [Example 53, “Preventing the root Role From Being Used to Maintain a System,” on page 111](#)

## Using Your Assigned Administrative Rights

In the root role, the initial user has all administrative rights. As root, this user can assign administrative rights, such as a role, a [rights profile](#), or specific privileges and authorizations to [trusted users](#). This section describes how these users can use their assigned rights.

---

**Note** - Oracle Solaris provides a special editor for administrative files. When editing administrative files, use the `pfedit` command. [Example 41, “Editing a System File,” on page 100](#) shows how to enable non-root users to edit specified system files.

---

To perform your administrative tasks, open a terminal window and choose from the following options:

- If you are using `sudo`, type the `sudo` command.  
For administrators who are familiar with the `sudo` command, run the command with the name of an administrative command that you are assigned in the `sudoers` file. For more information, see the [sudo\(1m\)](#) and `sudoers(4)` man pages.
- If your task requires superuser privileges, become root.

```
$ su -  
Password: xxxxxxxx  
#
```

---

**Note** - This command works whether `root` is a user or a role. The pound sign (#) prompt indicates that you are now root.

---

- If your task is assigned to a role, assume the role that can perform that task.  
In the following example, you assume an audit configuration role. This role includes the Audit Configuration rights profile. You received the role password from your administrator.

```
$ su - audadmin  
Password: xxxxxxxx  
#
```

---

**Tip** - If you did not receive a role password, your administrator has configured the role to require your user password. Type your user password to assume the role. For more information about this option, see [Example 18, “Enabling a User to Use Own Password for Role Password,”](#) on page 62.

---

The shell in which you typed this command is now a [profile shell](#). In this shell, you can run the `auditconfig` command. For more about profile shells, see [“Profile Shells and Rights Verification”](#) on page 37.

---

**Tip** - To view the rights of your role, see [“Listing Rights Profiles”](#) on page 115.

---

- If your task is assigned directly to you as a user and you are not running a profile shell as described in [Example 72, “Determining Whether You Are Using a Profile Shell,”](#) on page 127, create a profile shell in one of the following ways:

- Use the `pfbash` command to create a shell that evaluates administrative rights.

In the following example, you have been directly assigned the Audit Configuration rights profile. The following set of commands enables you to view audit preselection values and audit policy in the `pfbash` profile shell:

```
$ pfbash
$ auditconfig -getflags
active user default audit flags = ua,ap,lo(0x45000,0x45000)
configured user default audit flags = ua,ap,lo(0x45000,0x45000)
$ auditconfig -getpolicy
configured audit policies = cnt
active audit policies = cnt
```

- Use the `pfexec` command to run one administrative command.

In the following example, you have been directly assigned the Audit Configuration rights profile as an [authenticated rights profile](#). You can run a privileged command from this profile by using the `pfexec` command with the name of that command. For example, you can view the user's preselected audit flags:

```
$ pfexec auditconfig -getflags
Enter password:      Type your user password
active user default audit flags = ua,ap,lo(0x45000,0x45000)
configured user default audit flags = ua,ap,lo(0x45000,0x45000)
```

Typically, to run another privileged command that is included in your rights, you must type `pfexec` again before you type the privileged command. For more information, see

the `pfexec(1)` man page. If you are configured with password caching, you can run subsequent commands within a configurable interval without providing a password, as shown in [Example 42, “Caching Authentication for Ease of Role Use,”](#) on page 100.

**EXAMPLE 41** Editing a System File

If you are not root with the UID of 0, by default you cannot edit system files. However, if you are assigned the `solaris.admin.edit/path-to-system-file` authorization, you can edit *system-file*. For example, if you are assigned the `solaris.admin.edit/etc/security/audit_warn` authorization, you can edit the `audit_warn` file by using the `pfedit` command.

```
# pfedit /etc/security/audit_warn
```

For more information, see the `pfedit(4)` man page. This command is for use by all administrators.

**EXAMPLE 42** Caching Authentication for Ease of Role Use

In this example, the administrator configures a role to manage audit configuration, but provides ease of use by caching the user's authentication. First, the administrator creates and assigns the role.

```
# roleadd -K roleauth=user -K profiles="Audit Configuration" audadmin
# usermod -R +audadmin jdoe
```

When `jdoe` uses the `-c` option when switching to the role, a password is required before the `auditconfig` output is displayed:

```
$ su - audadmin -c auditconfig option
Password: xxxxxxxx
    auditconfig output
```

If authentication is not being cached, when `jdoe` runs the command again, a password prompt appears.

The administrator creates a file in the `pam.d` directory to hold an `su` stack that enables the caching of authentication. When authentication is cached, a password is initially required but not thereafter until a certain amount of time has passed.

```
# pfedit /etc/pam.d/su
## Cache authentication for switched user
#
auth required          pam_unix_cred.so.1
auth sufficient        pam_tty_tickets.so.1
```

```

auth requisite      pam_authtok_get.so.1
auth required       pam_dhkeys.so.1
auth required       pam_unix_auth.so.1

```

After creating the file, the administrator checks the entries for typos, omissions, or repetitions.

The administrator must provide the entire preceding su stack. The `pam_tty_tickets.so.1` module implements the cache. For more about PAM, see the [pam\\_tty\\_tickets\(5\)](#) and [pam.conf\(4\)](#) man pages and [Chapter 1, “Using Pluggable Authentication Modules” in \*Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3\*](#).

After the administrator adds the su PAM file and reboots the system, all roles including the `audadmin` role are prompted only once for a password when running a series of commands.

```

$ su - audadmin -c auditconfig option
Password: xxxxxxxx
    auditconfig output
$ su - audadmin -c auditconfig option
    auditconfig output
...

```

#### EXAMPLE 43 Assuming the root Role

In the following example, the initial user assumes the root role and lists the privileges in the role's shell.

```

$ roles
root
$ su - root
Password: xxxxxxxx
#      Prompt changes to root prompt
# ppriv $$
1200:  pfksh
flags = <none>
      E: all
      I: basic
      P: all
      L: all

```

For information about privileges, see [“Process Rights Management” on page 25](#) and the [ppriv\(1\)](#) man page.

#### EXAMPLE 44 Assuming an ARMOR Role

In this example, the user assumes an ARMOR role that the administrator assigned.

In a terminal window, the user determines which roles are assigned.

```
$ roles
fsadm
sysop
```

The user then assumes the `fsadm` role and supplies the user's password.

```
$ su - fsadm
Password: xxxxxxxx
#
```

The `su - rolename` command changes the terminal's shell to a profile shell. The user is now the `fsadm` role in this terminal window.

To determine which commands can be run in this role, the user follows the instructions in [“Listing Rights Profiles” on page 115](#).

## Auditing Administrative Actions

Site security [policy](#) often requires that you audit administrative actions. The `116:AUE_PFEEXEC:execve(2)` with `pfexec enabled:ps,ex,ua,as` audit event captures these actions. The `cosa` metaclass, which provides a group of events that is appropriate for use with roles, is another option when auditing administrative actions. For more information, review the comments in the `/etc/security/audit_class` file.

### EXAMPLE 45 Using Two Roles to Configure Auditing

In this example, two administrators implement the audit configuration plan of their site security officer. The plan is to use the `pf` class for all users, and specify the `cosa` metaclass for individual roles. The `root` role will assign the audit flags to the roles. The first administrator configures auditing and the second enables the new configuration.

The first administrator is assigned the Audit Configuration rights profile. This administrator views the current audit configuration:

```
# auditconfig -getflags
active user default audit flags = lo(0x1000,0x1000)
configured user default audit flags = lo(0x1000,0x1000)
```

Because the `pf` class does not include the `lo` class, the administrator adds the class to the system configuration.

```
# auditconfig -setflags lo,pf
```

To read the new audit configuration into the kernel, the administrator who is assigned the Audit Control rights profile refreshes the audit service.

```
# audit -s
```

## Creating Rights Profiles and Authorizations

You can create or change a rights profile when the provided rights profiles do not contain the collection of rights that you need. You might create a rights profile for users with limited rights, for a new application, or various other reasons.

The rights profiles that Oracle Solaris provides are read-only. You can clone a provided rights profile for modification if its collection of rights is insufficient. For example, you might want to add the `solaris.admin.edit/path-to-system-file` authorization to a provided rights profile. For background, see [“More About Rights Profiles” on page 23](#).

You can create an authorization when the provided authorizations do not include the authorizations that are coded in your privileged applications. You cannot change an existing authorization. For background, see [“More About User Authorizations” on page 23](#).

For examples of limiting privileges in a rights profile, see [Example 29, “Creating a Remote Users Rights Profile,” on page 77](#) and [Example 30, “Removing Basic Privileges From a Rights Profile,” on page 78](#).

### ▼ How to Create a Rights Profile

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

#### 1. Create a rights profile.

```
# profiles -p [-S repository] profile-name
```

You are prompted for a description.

#### 2. Add contents to the rights profile.

Use the `set` subcommand for profile properties that have a single value, such as `set desc`. Use the `add` subcommand for properties that can have more than one value, such as `add cmd`.

The following command creates the custom PAM rights profile in [“How to Assign a Modified PAM Policy” in \*Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3\*](#). The name is shortened for display purposes.

```
# profiles -p -S LDAP "Site PAM LDAP"
profiles:Site PAM LDAP> set desc="Profile which sets pam_policy=ldap"
...LDAP> set pam_policy=ldap
...LDAP> commit
...LDAP> end
...LDAP> exit
```

**Example 46** Creating a Sun Ray Users Rights Profile

In this example, the administrator creates a rights profile for Sun Ray users in the LDAP repository. The administrator has already created a Sun Ray version of the Basic Solaris User rights profile, and has removed all rights profiles from the `policy.conf` file on the Sun Ray server.

```
# profiles -p -S LDAP "Sun Ray Users"
profiles:Sun Ray Users> set desc="For all users of Sun Rays"
... Ray Users> add profiles="Sun Ray Basic User"
... Ray Users> set defaultpriv="basic,!proc_info"
... Ray Users> set limitpriv="basic,!proc_info"
... Ray Users> end
... Ray Users> exit
```

The administrator verifies the contents.

```
# profiles -p "Sun Ray Users" info
Found profile in LDAP repository.
    name=Sun Ray Users
    desc=For all users of Sun Rays
    defaultpriv=basic,!proc_info,
    limitpriv=basic,!proc_info,
    profiles=Sun Ray Basic User
```

**Example 47** Creating a Rights Profile That Includes Privileged Commands

In this example, the security administrator adds privileges to an application in a rights profile that the administrator creates. The application is privilege-aware.

```
# profiles -p SiteApp
profiles:SiteApp> set desc="Site application"
profiles:SiteApp> add cmd="/opt/site-app/bin/site-cmd"
profiles:SiteApp:site-cmd> add privs="proc_fork,proc_taskid"
profiles:SiteApp:site-cmd> end
profiles:SiteApp> exit
```

To verify, the administrator selects the `site-cmd`.



```
# profiles -p SiteApp "select cmd=/opt/site-app/bin/site-cmd; info;end"
Found profile in files repository.
  id=/opt/site-app/bin/site-cmd
  privs=proc_fork,proc_taskid
```

**Next Steps** Assign the rights profile to a trusted user or role. For examples, see [Example 12, “Creating a Trusted User to Administer DHCP,”](#) on page 60 and [Example 22, “Enabling a Trusted User to Read Extended Accounting Files,”](#) on page 64.

**See Also** To troubleshoot rights assignment, see [“How to Troubleshoot Rights Assignments”](#) on page 124. For background, see [“Order of Search for Assigned Rights”](#) on page 38.

## ▼ How to Clone and Modify a System Rights Profile

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights”](#) on page 98.

### 1. Create a new rights profile from an existing profile.

```
# profiles -p [-S repository] existing-profile-name
```

- **To add content to an existing rights profile, create a new profile.**

Add the existing rights profile as a supplementary rights profile to the new profile, then add the enhancements. See [Example 48, “Cloning and Enhancing the Network IPsec Management Rights Profile,”](#) on page 105.

- **To remove content from an existing rights profile, clone the profile and then rename it and modify.**

See [Example 49, “Cloning and Removing Selected Rights From a Rights Profile,”](#) on page 106.

### 2. Modify the new rights profile by adding or removing supplementary rights profiles, authorizations, and other rights.

**Example 48** Cloning and Enhancing the Network IPsec Management Rights Profile

In this example, the administrator adds a `solaris.admin.edit` authorization to a site IPsec Management rights profile so that the `root` role is not required. This rights profile will be assigned only to users who are trusted to modify the `/etc/hosts` file.

1. The administrator verifies that the Network IPsec Management rights profile cannot be modified.

```
# profiles -p "Network IPsec Management"
profiles:Network IPsec Management> add auths="solaris.admin.edit/etc/hosts"
Cannot add. Profile cannot be modified
```

2. The administrator creates a rights profile that includes the Network IPsec Management profile.

```
# profiles -p "Total IPsec Mgt"
... IPsec Mgt> set desc="Network IPsec Mgt plus /etc/hosts"
... IPsec Mgt> add profiles="Network IPsec Management"
... IPsec Mgt> add auths="solaris.admin.edit/etc/hosts"
... IPsec Mgt> end
... IPsec Mgt> exit
```

3. The administrator verifies the contents.

```
# profiles -p "Total IPsec Mgt" info
name=Total IPsec Mgt
desc=Network IPsec Mgt plus /etc/hosts
auths=solaris.admin.edit/etc/hosts
profiles=Network IPsec Management
```

#### Example 49 Cloning and Removing Selected Rights From a Rights Profile

In this example, the administrator separates managing the properties of the VSCAN service from the ability to enable and disable the service.

First, the administrator lists the contents of the rights profile that Oracle Solaris provides.

```
# profiles -p "VSCAN Management" info
name=VSCAN Management
desc=Manage the VSCAN service
auths=solaris.smf.manage.vscan,solaris.smf.value.vscan,
solaris.smf.modify.application
help=RtVscanMngmnt.html
```

Then, the administrator creates a rights profile that can enable and disable the service.

```
# profiles -p "VSCAN Management"
profiles:VSCAN Management> set name="VSCAN Control"
profiles:VSCAN Control> set desc="Start and stop the VSCAN service"
... VSCAN Control> remove auths="solaris.smf.value.vscan"
... VSCAN Control> remove auths="solaris.smf.modify.application"
... VSCAN Control> end
... VSCAN Control> exit
```

Then, the administrator creates a rights profile that can change the properties of the service.

```
# profiles -p "VSCAN Management"
profiles:VSCAN Management> set name="VSCAN Properties"
profiles:VSCAN Properties> set desc="Modify VSCAN service properties"
... VSCAN Properties> remove auths="solaris.smf.manage.vscan"
... VSCAN Properties> end
... VSCAN Properties> exit
```

The administrator verifies the contents of the new rights profiles.

```
# profiles -p "VSCAN Control" info
    name=VSCAN Control
    desc=Start and stop the VSCAN service
    auths=solaris.smf.manage.vscan
# profiles -p "VSCAN Properties" info
    name=VSCAN Properties
    desc=Modify VSCAN service properties
    auths=solaris.smf.value.vscan,solaris.smf.modify.application
```

**Next Steps** Assign the rights profile to a trusted user or role. For examples, see [Example 12, “Creating a Trusted User to Administer DHCP,” on page 60](#) and [Example 22, “Enabling a Trusted User to Read Extended Accounting Files,” on page 64](#).

**See Also** To troubleshoot rights assignment, see [“How to Troubleshoot Rights Assignments” on page 124](#). For background, see [“Order of Search for Assigned Rights” on page 38](#).

## ▼ How to Create an Authorization

**Before You Begin** Developers have defined and used the authorization in the applications that you are installing. For instructions, see [“About Authorizations” in \*Developer’s Guide to Oracle Solaris 11.3 Security\*](#).

### 1. (Optional) Create the help file for your new authorization.

For example, create the help file for an authorization to enable the user to modify the data in an application.

```
# pfedit /docs/helps/NewcoSiteAppModData.html
<HTML>
-- Copyright 2015 Newco. All rights reserved.
-- NewcoSiteAppModData.html
-->
<HEAD>
    <TITLE>NewCo Modify SiteApp Data Authorization</TITLE>
```

```
</HEAD>
<BODY>
The com.newco.siteapp.data.modify authorization authorizes you
to modify existing data in the application.
<p>
Only authorized accounts are permitted to modify data.
Use this authorization with care.
<p>
</BODY>
</HTML>
```

## 2. Create the authorization by using the `auths add` command.

For example, the following command creates the `com.newco.siteapp.data.modify` authorization on the local system.

```
# auths add -t "SiteApp Data Modify Authorized" \
-h /docs/helps/NewcoSiteAppModData.html com.newco.siteapp.data.modify
```

You can now test the authorization, then add it to a rights profile and assign the profile to a role or user.

### Example 50 Testing a New Authorization

In this example, the administrator tests the `com.newco.siteapp.data.modify` authorization with the SiteApp rights profile from [Example 47, “Creating a Rights Profile That Includes Privileged Commands,” on page 104](#).

```
# usermod -A com.newco.siteapp.data.modify -P SiteApp tester1
```

When the test succeeds, the administrator removes the authorization.

```
# rolemod -A=com.newco.siteapp.data.modify siteapptester
```

For ease of maintenance, the administrator adds the authorization to the SiteApp rights profile in [Example 51, “Adding Authorizations to a Rights Profile,” on page 108](#).

### Example 51 Adding Authorizations to a Rights Profile

After testing that the authorization works correctly, the security administrator adds the `com.newco.siteapp.data.modify` authorization to an existing rights profile. [Example 47, “Creating a Rights Profile That Includes Privileged Commands,” on page 104](#) shows how the administrator created the profile.

```
# profiles -p "SiteApp"
profiles:SiteApp> add auths="com.newco.siteapp.data.modify"
profiles:SiteApp> end
```

```
profiles:SiteApp> exit
```

To verify, the administrator lists the contents of the profile.

```
# profiles -p SiteApp
Found profile in files repository.
  id=/opt/site-app/bin/site-cmd
  auths=com.newco.siteapp.data.modify
```

**Next Steps** Assign the rights profile to a trusted user or role. For examples, see [Example 12, “Creating a Trusted User to Administer DHCP,”](#) on page 60 and [Example 22, “Enabling a Trusted User to Read Extended Accounting Files,”](#) on page 64.

**See Also** To troubleshoot rights assignment, see [“How to Troubleshoot Rights Assignments”](#) on page 124. For background, see [“Order of Search for Assigned Rights”](#) on page 38.

## Changing Whether root Is a User or a Role

By default, root is a role in Oracle Solaris. You have the option to change it to a user, change it back in to a role, or remove it from use.

You must change root to a user if you are using [Oracle Enterprise Manager](#) or are following the traditional [superuser model](#) of administration rather than the rights model. For background, see [“Deciding Which Rights Model to Use for Administration”](#) on page 43.

If you are following the rights model, you might change root to a user when decommissioning a system that has been removed from the network. In this scenario, logging in to the system as root simplifies the cleanup.

---

**Note** - If you administer remotely with the root role, see [“How to Remotely Administer ZFS With Secure Shell”](#) in *Managing Secure Shell Access in Oracle Solaris 11.3* for secure remote login instructions.

---

At some sites, root is not a legitimate account on production systems. To remove root from use, see [Example 53, “Preventing the root Role From Being Used to Maintain a System,”](#) on page 111.

### ▼ How to Change the root Role Into a User

This procedure is required on systems where root must be able to log in directly to the system.

**Before You Begin** You must assume the root role.

**1. Remove the root role assignment from local users.**

For example, remove the role assignment from two users.

```
$ su -
Password: xxxxxxxx
# roles jdoe
root
# roles kdoe
root
# roles ldoe
secadmin
# usermod -R "" jdoe
# usermod -R "" kdoe
#
```

**2. Change the root role into a user.**

```
# rolemod -K type=normal root
```

Users who are currently in the root role remain so, Other users who have root access can su to root or log in to the system as the root user.

**3. Verify the change.**

You can use one of the following commands.

■ **Examine the user\_attr entry for root.**

```
# getent user_attr root
root:::auths=solaris.*;profiles=All;audit_flags=lo\:no;lock_after_retries=no
```

If the type keyword is missing in the output or is equal to normal, the account is not a role.

■ **View the output from the userattr command.**

```
# userattr type root
```

If the output is empty or lists normal, the account is not a role.

**Example 52** Changing the root User Into the root Role

In this example, the root user turns the root user back into a role.

First, the root user changes the root account into a role and verifies the change.

```
# usermod -K type=role root
# getent user_attr root
root::::type=role...
```

Then, root assigns the root role to a local user.

```
# usermod -R root jdoe
```

### Example 53 Preventing the root Role From Being Used to Maintain a System

In this example, site security [policy](#) requires that the root account be prevented from maintaining the system. The administrator has created and tested the roles which maintain the system. These roles include every security profile and the System Administrator rights profile. A trusted user has been assigned a role that can restore a backup. No role can change the audit flags for a user, role, or a rights profile or change the password of a role.

To prevent the root account from being used to maintain the system, the security administrator removes the root role assignment. Because the root account must be able to log in to the system in single-user mode, the account retains a password.

```
# usermod -K roles= jdoe
# userattr roles jdoe
```

**Troubleshooting** In a desktop environment, you cannot directly log in as root when root is a role. A diagnostic message indicates that root is a role on your system.

If you do not have a local account that can assume the root role by performing the following steps:

- As root, log in to the system in single-user mode, create a local user account and password.
- Assign the root role to the new account.
- Log in as the new user and assume the root role.





## Listing Rights in Oracle Solaris

---

This chapter describes how to list all rights on the system, rights that are assigned to specific users, and your own rights:

- [“Listing Authorizations” on page 114](#)
- [“Listing Rights Profiles” on page 115](#)
- [“Listing Roles” on page 117](#)
- [“Listing Privileges” on page 118](#)
- [“Listing Qualified Attributes” on page 121](#)

For an overview of rights, see [“User Rights Management” on page 16](#). For reference information, see [Chapter 8, “Reference for Oracle Solaris Rights”](#).

### Listing Rights and Their Definitions

The commands in this section enable you to find rights that are defined on the system, and list the rights that are in effect on a user's process.

For a full description of the commands in this section, see the following man pages:

- [auths\(1\)](#)
- [getent\(1M\)](#)
- [ppriv\(1\)](#)
- [profiles\(1\)](#)
- [privileges\(5\)](#)
- [roles\(1\)](#)

## Listing Authorizations

- `auths` – Lists the current user's authorizations
- `auths list` – Lists the current user's authorizations
- `auths list -u username` – Lists the authorizations for *username*
- `auths list -x` – Lists the current user's authorizations that require authentication
- `auths list -xu username` – Lists the *username*'s authorizations that require authentication
- `auths info` – Lists all authorization names in the naming service
- `getent auth_attr` – Lists the full definition of all authorizations in the naming service

### EXAMPLE 54 Listing All Authorizations

```
$ auths info
solaris.account.activate
solaris.account.setpolicy
solaris.admin.edit
...
solaris.zone.login
solaris.zone.manage
```

### EXAMPLE 55 Listing the Content of the Authorizations Database

```
$ getent auth_attr | more
solaris.:::All Solaris Authorizations:::help=AllSolAuthsHeader.html
solaris.account.:::Account Management:::help=AccountHeader.html
...
solaris.zone.login:::Zone Login:::help=ZoneLogin.html
solaris.zone.manage:::Zone Deployment:::help=ZoneManage.html
```

### EXAMPLE 56 Listing the Default Authorizations of Users

The following authorizations are included in the rights profiles that are assigned to all users by default.

```
$ auths
solaris.device.cdrw,solaris.device.mount.removable,solaris.mail.mailq
solaris.network.autoconf.read,solaris.admin.wusb.read
solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
```

## Listing Rights Profiles

- `profiles` – Lists the current user's rights profiles
- `profiles -a` – Lists all rights profiles names
- `profiles -l` – Lists the full definition of the current user's rights profiles
- `profiles username` – Lists the rights profiles for *username*
- `profiles -x` – Lists the current user's rights profiles that require authentication
- `profiles -x username` – Lists the *username*'s rights profiles that require authentication
- `profiles -p profile-name info` – Pretty prints the contents of specified rights profile
- `getent prof_attr` – Lists the full definition of all rights profiles in the naming service

**EXAMPLE 57** Listing the Names of All Rights Profiles

```
$ profiles -a
    Console User
    CUPS Administration
    Desktop Removable Media User
...
    VSCAN Management
    WUSB Management
```

**EXAMPLE 58** Listing the Contents of the Rights Profiles Database

```
$ getent prof_attr | more
All::Execute any command as the user or role:help=RtAll.html
Audit Configuration:::Configure Solaris Audit:auths=solaris.smf.value.audit;
help=RtAuditCfg.html
...
Zone Management:::Zones Virtual Application Environment Administration:
help=RtZoneMngmnt.html
Zone Security:::Zones Virtual Application Environment Security:auths=solaris.zone.*,
solaris.auth.delegate;help=RtZoneSecurity.html ...
```

**EXAMPLE 59** Listing the Default Rights Profiles of Users

List your rights profiles. The following rights profiles are assigned to all users by default.

```
$ profiles
Basic Solaris User
All
```

**EXAMPLE 60** Listing the Rights Profiles of the Initial User

The initial user is assigned several rights profiles.

```
$ profiles Initial user
System Administrator
Audit Review
...
CPU Power Management
Basic Solaris User
All
```

To show all the [security attributes](#) that are assigned to the initial user's profiles, use the `-l` option.

```
$ profiles -l Initial user | more
Initial user:
System Administrator
  profiles=Install Service Management,Audit Review,Extended Accounting
  Flow Management,Extended Accounting Net Management,Extended Accounting Process
  Management,Extended Accounting Task Management,Printer Management,Cron Managem
  ent,Device Management,File System Management,Log Management,Mail Management,
  Maintenance and Repair,Media Catalog,Name Service Management,Network Management,
  Project Management,RAD Management,Service Operator,Shadow Migration Monitor,So
  Software Installation,System Configuration,User Management,ZFS Storage Management
  /usr/sbin/gparted      uid=0
Install Service Management
  auths=solaris.autoinstall.service
  profiles=Install Manifest Management,Install Profile Management,
  Install Client Management
...
```

**EXAMPLE 61** Listing the Contents of an Assigned Rights Profile

The initial user lists the rights that are granted by the Audit Review profile.

```
$ profiles -l
Audit Review
  solaris.audit.read

  /usr/sbin/auditreduce  euid=0
  /usr/sbin/auditstat    privs=proc_audit
  /usr/sbin/praudit      privs=file_dac_read
```

**EXAMPLE 62** Listing the Security Attributes of a Command in a Rights Profile

This variant of the `profiles` command is useful for viewing the security attributes of a command in a rights profile that is not assigned to you.

First, list the commands in the profile.

```
$ profiles -p "Audit Review" info
name=Audit Review
desc=Review Solaris Auditing logs
help=RtAuditReview.html
cmd=/usr/sbin/auditreduce
cmd=/usr/sbin/auditstat
cmd=/usr/sbin/praudit
```

Then, list the security attributes of one of the commands in the profile.

```
$ profiles -p "Audit Review" "select cmd=/usr/sbin/praudit ; info; end;"
select: command is read-only
  id=/usr/sbin/praudit
  privs=file_dac_read
end: command is read-only
```

**EXAMPLE 63** Listing the Contents of Rights Profiles That Are Recently Created

The `less` option displays the most recently added rights profiles first. This variant of the `profiles` command is useful when you create or modify rights profiles at your site. The following output shows the contents of the profile that was added in [Example 35, “Assigning Security Attributes to a Legacy Application,” on page 83](#). A regular user can run this command.

```
$ profiles -la | less
LegacyApp
      /opt/legacy-app/bin/legacy-cmd
                                euid=0
OpenLDAP...
```

## Listing Roles

- `roles` – Lists the current user's roles
- `roles username` – Lists the roles for *username*
- `logins -r` – Lists all available roles

**EXAMPLE 64** Listing Your Assigned Roles

The root role is assigned to the initial user by default. No `roles` indicates that you are not assigned a role.

```
$ roles
root
```

## Listing Privileges

- `man privileges` – Lists privilege definitions and their names as they are used by developers
- `ppriv -vl` – Lists privilege definitions and their names as they are used by administrators
- `ppriv -vl basic` – Lists names and definitions of privileges in the basic set of privileges
- `ppriv $$` – Lists the privileges in the current shell (`$$`)
- `getent exec_attr` – Lists all commands that have security attributes (`setuid` or privileges) by rights profile name

```
$ getent exec_attr | more
All:solaris:cmd::*:
Audit Configuration:solaris:cmd:::/usr/sbin/auditconfig:privs=sys_audit
...
Zone Security:solaris:cmd:::/usr/sbin/txzonemgr:uid=0
Zone Security:solaris:cmd:::/usr/sbin/zonecfg:uid=0 ...
```

**EXAMPLE 65** Listing All Privileges and Their Definitions

The privilege format described in the [privileges\(5\)](#) man page is used by developers.

```
$ man privileges
Standards, Environments, and Macros           privileges(5)

NAME
  privileges - process privilege model
...
  The defined privileges are:

  PRIV_CONTRACT_EVENT

      Allow a process to request reliable delivery of events
      to an event endpoint.

      Allow a process to include events in the critical event
```

```

    set term of a template which could be generated in
    volume by the user.
...

```

**EXAMPLE 66** Listing Privileges That Are Used in Privilege Assignment

The `ppriv` command lists all privileges by name. For a definition, use the `-v` option.

This privilege format is used to assign privileges to users and roles with the `useradd`, `roleadd`, `usermod`, and `rolemod` commands, and to rights profiles with the `profiles` command.

```

$ ppriv -lv | more
contract_event
  Allows a process to request critical events without limitation.
  Allows a process to request reliable delivery of all events on
  any event queue.
...
win_upgrade_sl
  Allows a process to set the sensitivity label of a window
  resource to a sensitivity label that dominates the existing
  sensitivity label.
  This privilege is interpreted only if the system is configured
  with Trusted Extensions.

```

**EXAMPLE 67** Listing the Privileges in Your Current Shell

Every user is assigned the basic privilege set by default. The default limit set is all privileges.

The single letters in the output refer to the following privilege sets:

|   |                           |
|---|---------------------------|
| E | Effective privilege set   |
| I | Inheritable privilege set |
| P | Permitted privilege set   |
| L | Limit privilege set       |

```

$ ppriv $$
1200: -bash
flags = <none>
  E: basic
  I: basic
  P: basic

```

```
      L: all
$ ppriv -v $$
1200:  -bash
flags = <none>
E: file_link_any,file_read,file_write,net_access,proc_exec,proc_fork,
   proc_info,proc_session,sys_ib_info
I: file_link_any,file_read,...,sys_ib_info
P: file_link_any,file_read,...,sys_ib_info
L: contract_event,contract_identity,...,sys_time
```

The double dollar sign (\$\$) passes the process number of the parent shell to the command. This listing does not include privileges that are restricted to commands in an assigned rights profile.

**EXAMPLE 68** Listing the Basic Privileges and Their Definitions

```
$ ppriv -vl basic
file_link_any
  Allows a process to create hardlinks to files owned by a uid
  different from the process' effective uid.
file_read
  Allows a process to read objects in the filesystem.
file_write
  Allows a process to modify objects in the filesystem.
net_access
  Allows a process to open a TCP, UDP, SDP or SCTP network endpoint.
proc_exec
  Allows a process to call execve().
proc_fork
  Allows a process to call fork1()/forkall()/vfork()
proc_info
  Allows a process to examine the status of processes other
  than those it can send signals to. Processes which cannot
  be examined cannot be seen in /proc and appear not to exist.
proc_session
  Allows a process to send signals or trace processes outside its
  session.
sys_ib_info
  Allows a process to perform read InfiniBand MAD (Management Datagram)
  operations.
```

**EXAMPLE 69** Listing the Commands With Security Attributes in Your Rights Profiles

The Basic Solaris User profile includes commands that enable users to read and write to CD-ROMs.

```
$ profiles -l
```



```

Basic Solaris User
...
/usr/bin/cdrecord.bin  privs=file_dac_read,sys_devices,
    proc_lock_memory,proc_priocntl,net_privaddr
/usr/bin/readcd.bin    privs=file_dac_read,sys_devices,net_privaddr
/usr/bin/cdda2wav.bin  privs=file_dac_read,sys_devices,
    proc_priocntl,net_privaddr
All
*
```

## Listing Qualified Attributes

- `man user_attr` – Defines qualifiers of security attributes
- `getent` – Lists qualified security attributes of a user or role on the system where the command is run
- `ldapaddent` – Lists all qualified security attributes of a user or role

**EXAMPLE 70** Listing a User's Qualified Attributes on This System

```

system1$ getent user_attr | grep jdoe:
jdoe:system1:::profiles=System Administrator
```

**EXAMPLE 71** Listing All Qualified Attributes for a User in LDAP

```

system1$ ldapaddent -d user_attr | grep ^jdoe:
jdoe:system1:::profiles=System Administrator
jdoe:sysopgroup:::profiles=System Operator
```



## Troubleshooting Rights in Oracle Solaris

---

This chapter provides troubleshooting suggestions when managing and using administrative rights in Oracle Solaris:

- [“Troubleshooting RBAC and Privileges” on page 123](#)
- [“Troubleshooting Passwords” on page 132](#)

For information about using rights, review the following information:

- [Chapter 3, “Assigning Rights in Oracle Solaris”](#)
- [“Who Can Assign Rights” on page 48](#)
- [“User Rights Management” on page 16](#)
- [“Process Rights Management” on page 25](#)

For information about passwords, see [“Special System Accounts” in \*Securing Systems and Attached Devices in Oracle Solaris 11.3\*](#) and the `passwd(1)` and `user_attr(4)` man pages.

### Troubleshooting RBAC and Privileges

The tasks and examples in this section suggest ways to solve problems with rights assignments. For background information, see [“Rights Verification” on page 37](#).

Use the command-line interfaces to assign rights. The following commands modify the rights databases:

- `passwd`
- `useradd`, `usermod`, and `userdel`
- `roleadd`, `rolemod`, and `roledel`
- `profiles`
- `auths`



---

**Caution** - Do not use an editor to modify a rights database. The editor cannot check for syntax validity or update kernel processes.

---

## ▼ How to Troubleshoot Rights Assignments

Several factors can affect why rights are not being evaluated and correctly applied. This procedure helps you debug why assigned rights might not be available to users, roles, or processes. Several of the steps are based on [“Order of Search for Assigned Rights” on page 38](#).

**Before You Begin** You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

### 1. Verify and restart the naming service.

#### a. Verify that the security assignments for the user or role are in the naming service that is enabled on the system.

```
# svccfg -s name-service/switch

svc:/system/name-service/switch>
listprop config

config                application
config/value_authorization astring solaris.smf.value.name-service.switch
config/default         astring files ldap
config/host            astring "files dns mdns ldap"
config/netgroup        astring ldap
config/printer         astring "user files"
```

In this output, all services that are not explicitly mentioned inherit the value of the default, `files ldap`. Therefore, `passwd` and its related attribute databases, `user_attr`, `auth_attr`, and `prof_attr`, are searched first in files, then in LDAP.

#### b. Restart the name service cache, `svc:/system/name-service/cache`.

The `nscd` daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the naming service with current data.

```
# svcadm restart name-service/cache
```

### 2. Determine where a right is assigned to the user by running the `userattr -v` command.

Run the command once for every attribute. For example, the following commands indicate which rights are assigned and where the assignment was made for the user `jdoe`. The lack of output indicates that `jdoe` is using the defaults.

```
$ userattr -v access_times jdoe
```

```

$ userattr -v access_tz jdoe
$ userattr -v auth_profiles jdoe
$ userattr -v defaultpriv jdoe
$ userattr -v limitpriv jdoe
$ userattr -v idlecmd jdoe
$ userattr -v idletime jdoe
$ userattr -v lock_after_retries jdoe
$ userattr -v pam_policy jdoe

$ userattr -v auths jdoe      Output indicates authorizations from rights profiles
Basic Solaris User :solaris.mail.mailq,solaris.network.autoconf.read,
solaris.admin.wusb.read
Console User :solaris.system.shutdown,solaris.device.cdrw,
solaris.device.mount.removable,solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
$ userattr -v audit_flags jdoe
user_attr: fw:no      Output indicates jdoe is individually assigned audit flags
$ userattr -v profiles jdoe
user_attr: Audit Review,Stop      Output indicates two assigned rights profiles
$ userattr roles jdoe
user_attr : cryptomgt,infosec      Output indicates two assigned roles

```

The output indicates that jdoe is directly assigned audit flags, two rights profiles, and two roles. The assigned authorizations are from default rights profiles in the `/etc/security/policy.conf` file.

- Because jdoe is directly assigned audit flags, no audit flag values in the rights profiles will be used.
- The rights profiles are evaluated in order, first the Audit Review rights profile, then the Stop profile.
- All other rights are assigned to jdoe in the roles cryptomgt and infosec. To view those rights, jdoe must assume each role, then list the rights.

If the right is not directly assigned to the user, continue with the following checks.

### 3. Verify that the assigned authorizations are spelled correctly.

The source of an authorization assignment is not important because authorizations accumulate for users. However, a misspelled authorization fails silently.

### 4. For rights profiles that you have created, verify that you have assigned the appropriate [security attributes](#) to the commands in that profile.

For example, some commands require `uid=0` rather than `euid=0` to succeed. Review the man page for the command to determine whether the command or any of its options require authorizations.

### 5. Check the rights in the user's rights profiles.

**a. In order, check for the rights in the list of authenticated rights profiles.**

The value of the attribute in the earliest rights profile in the list is the value in the kernel. If this value is incorrect, either change the value in that rights profile, or reassign the profiles in the correct order. See [“How to Reorder Assigned Rights” on page 128](#).

For privileged commands, check that the privileges are not removed from the `defaultpriv` or `limitpriv` keyword.

**b. In order, check for the rights in the list of regular rights profiles.**

Follow the same checks as you performed for authenticated rights profiles.

**c. If the rights you are searching for are not listed, check the roles that the user is assigned.**

If the right is assigned to a role, the user must assume the role to obtain the rights.

**6. Check whether a failed command requires authorizations to succeed.**

**a. Check whether an existing rights profile includes the required authorization.**

If the profile exists, use it. Assign it to the user as an [authenticated rights profile](#) or a regular rights profile. Order the profile before any other rights profile that includes the command that requires this authorization to succeed.

**b. Check whether an option to the command requires authorization.**

Assign the privilege to the command that requires it, add the required authorizations, place the command and authorizations in a rights profile, and assign the profile to the user.

**7. If a command continues to fail for a user, verify that the user is executing the command in a [profile shell](#).**

Administrative commands must be executed in a profile shell. [Example 72, “Determining Whether You Are Using a Profile Shell,” on page 127](#) shows how to test for a profile shell.

To reduce the likelihood of user error, you can try the following:

- Assign a profile shell as the user's login shell.
- Instruct users to precede all privileged commands with the `pfexec` command.
- Remind the user to run administrative commands in a profile shell.
- If your site is using roles, remind the user to assume the role before running administrative commands. For an example of successful command execution as a role rather than as a user, see [Example 74, “Running the Privileged Commands in Your Role,” on page 128](#).

**8. If a command fails for a role, assume the role and perform the same steps that you performed when checking for a user's rights.**

**Example 72** Determining Whether You Are Using a Profile Shell

When a privileged command does not work, the user tests for the PRIV\_PFEEXEC flag, then runs the command. The error message might not indicate that the problem is a privilege problem.

```
$ praudit 20120814200247.20120912213421.example-system
praudit: Cannot associate stdin with 20120814200247.20120912213421.example-system:
Permission denied

$ ppriv $$
107219: bash
flags = <none>
...

$ pfbash
$ ppriv $$
1072232: bash
flags = PRIV_PFEEXEC
...

$ praudit 20120814200247.20120912213421.example-system
/** Command succeeds **/
```

**Example 73** Determining the Privileged Commands of a Role

In this example, a user assumes an assigned role and lists the rights that are included in one of the rights profiles. The rights are truncated to emphasize the commands.

```
$ roles
devadmin

$ su - devadmin
Password: xxxxxxxx

$ profiles -l
Device Security
...
profiles=Service Configuration
    /usr/sbin/add_drv          uid=0
    /usr/sbin/devfsadm        uid=0
                                privs=sys_devices,sys_config,
                                sys_resource,file_owner,
                                file_chown,file_chown_self,
```

```

                                file_dac_read
/usr/sbin/eeprom                uid=0
/usr/bin/kbd                    uid=0
/usr/sbin/list_devices          euid=0
/usr/sbin/rem_drv               uid=0
/usr/sbin/strace                euid=0
/usr/sbin/update_drv           uid=0
/usr/sbin/add_allocatable      euid=0
/usr/sbin/remove_allocatable   euid=0
Service Configuration
/usr/sbin/svcadm
/usr/sbin/svccfg

```

**Example 74** Running the Privileged Commands in Your Role

In the following example, the `admin` role can change the permissions on the `useful.script` file.

```

$ 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: xxxxxxxx

$ chgrp admin useful.script
$ chown admin useful.script
$ ls -l useful.script
-rwxr-xr-- 1 admin admin 262 Apr 2 10:53 useful.script

```

## ▼ How to Reorder Assigned Rights

You must reorder a user's rights profiles assignments when an unprivileged command is in effect for the user rather than its privileged version. For more information, see [“Order of Search for Assigned Rights” on page 38](#).

**Before You Begin** You must become an administrator who is assigned the User Security rights profile. For more information, see [“Using Your Assigned Administrative Rights” on page 98](#).

1. **View the list of rights profiles that are currently assigned to the user or role.**



The list displays in order.

```
$ profiles username | rolename
```

## 2. Assign the rights profiles in the correct order.

```
# usermod | rolemod -K profiles="list-of-profiles"
```

### Example 75 Assigning Rights Profiles in a Specific Order

In this example, the administrator determines that a rights profile with privileged commands is listed after the All rights profile for the role `devadmin`.

```
# profiles devadmin
```

```
Basic Solaris User
All
Device Management
```

Therefore, the `devadmin` role cannot run the device management commands with the role's assigned privileges.

The administrator reassigns the rights profiles to `devadmin`. In the new order of assignment, the device management commands run with their assigned privileges.

```
# rolemod -K profiles="Device Management,Basic Solaris User,All"
```

```
# profiles devadmin
```

```
Device Management
Basic Solaris User
All
```

## ▼ How to Determine Which Privileges a Program Requires

Use this debugging procedure when a command or process is failing. After finding the first privilege failure and fixing it, you might need to run the `ppriv -eD command` command again to find additional privilege requirements.

### 1. Type the command that is failing as an argument to the `ppriv` debugging command.

```
$ ppriv -eD touch /etc/acct/yearly

touch[5245]: missing privilege "file_dac_write"
           (eid = 130, syscall = 224) needed at zfs_zaccess+0x258
touch: cannot create /etc/acct/yearly: Permission denied
```

**2. Use the `syscall` number from the debugging output to determine which system call is failing.**

You find the name of the `syscall` number in the `/etc/name_to_sysnum` file.

```
$ grep 224 /etc/name_to_sysnum

creat64                224
```

In this example, the `creat64()` call is failing. To succeed, the process must be assigned the right to create a file in the `/etc/acct/yearly` directory.

**Example 76** 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 `file_dac_write` privilege after the error code in `truss` output.

**Example 77** Using the `ppriv` Command to Examine Privilege Use in a Profile Shell

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:

```
jdoe$ ls -l useful.script

-rw-r--r-- 1 aloo staff 2303 Apr 10 10:10 useful.script
```

```

jdoe$
chown objadmin useful.script

chown: useful.script: Not owner
jdoe$
ppriv -eD chown objadmin useful.script

chown[11444]: missing privilege "file_chown"
             (euid = 130, syscall = 16) needed at zfs_zaccess+0x258
chown: useful.script: Not owner

```

When jdoe assumes the objadmin role, the permissions on the file are changed:

```

jdoe$ su - objadmin
Password: xxxxxxxx

$ ls -l useful.script
-rw-r--r-- 1 aloe staff 2303 Apr 10 10:10 useful.script

$ chown objadmin useful.script
$ ls -l useful.script
-rw-r--r-- 1 objadmin staff 2303 Apr 10 10:10 useful.script
$ chgrp admin useful.script

$ ls -l objadmin.script
-rw-r--r-- 1 objadmin admin 2303 Apr 10 10:11 useful.script

```

#### Example 78 Changing a File Owned by the root User

This example illustrates the protections against [privilege escalation](#). For a discussion, see [“Privilege Escalation and Kernel Privileges” on page 36](#). 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.

```

jdoe$ su - objadmin
Password: xxxxxxxx

$ cd /etc; ls -l system
-rw-r--r-- 1 root sys 1883 Oct 10 10:20 system

$ chown objadmin system
chown: system: Not owner
$ ppriv -eD chown objadmin system
chown[11481]: missing privilege "ALL"
             (euid = 101, syscall = 16) needed at zfs_zaccess+0x258
chown: system: Not owner

```

## Troubleshooting Passwords

The following examples suggest ways to debug problems with passwords. For background information, see the [passwd\(1\)](#) and [user\\_attr\(4\)](#) man pages.

### EXAMPLE 79 Using the openldap System Account to Run a cron Job

In this example, the administrator changes the password string for an Oracle Solaris-delivered system account from \*LK\* to NP. The account runs a cron job that updates the data in the directory periodically with an external data source over a UNIX domain socket that is locked down to the openldap user only.

Initially, the openldap account is locked. The time that the password was locked is 12111.

```
# grep openldap /etc/shadow
openldap:*LK*:12111::::::
# passwd -N openldap
WARNING: changing account in reserved uid range: openldap.
passwd: password information changed for openldap
```

The -N makes the password entry a value that cannot be used for login with UNIX authentication.

```
# passwd -u openldap
WARNING: changing account in reserved uid range: openldap.
passwd: password information changed for openldap
```

The -u option unlocks the account. The account remains a non-UNIX authentication account.

```
# grep openldap /etc/shadow
openldap:NP:13222::::::
```

In the final entry, the openldap non-UNIX authentication account has no password. The account is protected by the locked-down UNIX domain socket.

### EXAMPLE 80 Creating a Role That Requires the User's Password

This example shows how to configure administrative accounts to be authenticated with the user's password, similar to the way the sudo user is configured. In the root role, you create the user and the role, make the role require the user password, and assign the role to the user.

```
# useradd [ ... ] jdoe
# roleadd [ ... ] administrator
```

```
# rolemod -K roleauth=user administrator
# usermod -K type=role administrator jdoe
```

**EXAMPLE 81** Overriding the Password Requirements for an Account

In this example, the root role changes a password several times, overriding the password constraints.

```
# passwd -n 14 -x 25 user1
```

The `/etc/shadow` entry resembles the following:

```
user1:$5$Cuz1WCgx$4CFN...:last-changed-date:14:25:::
```

The administrator in the root role is able to change the password several times the same day:

```
# passwd user1
Enter user1's password: password
# grep user1 /etc/shadow
user1:$5$Cuz1WCgx$4CFN...:11333:14:25:::
# passwd user1
Enter user1's password: password
user1:$5$Cuz1WCgx$4CFN...:11444:14:25:::
```

Because the root role has the `solaris.passwd.assign` authorization, root can override the password constraints of a minimum of 14 days between password changes (`-n 14`) and a maximum of 25 days (`-x 25`). `user1`, who does not have the `solaris.passwd.assign` authorization, cannot change the assigned password for at least two weeks. As the following shows, the user also cannot change another user's password:

```
$ passwd user2
Enter user1's password: password
Permission denied: Missing authorization: solaris.passwd.assign
```



## Reference for Oracle Solaris Rights

---

This chapter provides reference material about the use of administrative rights in Oracle Solaris:

- [“Rights Profiles Reference” on page 135](#)
- [“Authorizations Reference” on page 137](#)
- [“Rights Databases” on page 138](#)
- [“Commands for Administering Rights” on page 142](#)
- [“Privileges Reference” on page 144](#)

For information about using rights, including privileges, see [Chapter 3, “Assigning Rights in Oracle Solaris”](#). For overview information, see [“User Rights Management” on page 16](#) and [“Process Rights Management” on page 25](#).

### Rights Profiles Reference

This section describes some typical rights profiles. Rights profiles are convenient collections of authorizations and other [security attributes](#), commands with security attributes, and supplementary rights profiles. Oracle Solaris provides many rights profiles. If they are not sufficient for your needs, you can modify existing ones and create new ones.

Rights profiles must be assigned in order, from most to least powerful. For more information, see [“Order of Search for Assigned Rights” on page 38](#).

To view the contents of the following rights profiles, see [“Viewing the Contents of Rights Profiles” on page 136](#).

- **System Administrator rights profile** – Provides access to most tasks that are not connected with security. This profile includes several other profiles to create a powerful role. Note that the All rights profile is assigned at the end of the list of supplementary rights profiles.
- **Operator rights profile** – Provides limited rights 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. 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 or assign the Stop rights profile. For the implementation of the Basic Solaris User rights profile, see [Example 69, “Listing the Commands With Security Attributes in Your Rights Profiles,”](#) on page 120.
- **Console User rights profile** – For the workstation owner, provides access to authorizations, commands, and actions for the person who is seated at the computer.
- **All rights profile** – For roles, provides access to commands that do not have security attributes. This profile can be appropriate for users with limited rights.
- **Stop rights profile** – A special rights profile that stops the evaluation of further profiles. This profile prevents the evaluation of the `AUTHS_GRANTED`, `PROFS_GRANTED`, and `CONSOLE_USER` variables in the `policy.conf` file. With this profile, you can provide roles and users with a restricted profile shell.

---

**Note** - The Stop profile affects privilege assignment indirectly. Rights profiles that are listed after the Stop profile are not evaluated. Therefore, the commands with privileges in those profiles are not in effect. See [Example 31, “Restricting an Administrator to Explicitly Assigned Rights,”](#) on page 78.

---

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/profiles/locale/C` directory.

## Viewing the Contents of Rights Profiles

You have three views into the contents of rights profiles:

- The `getent` command enables you to view the contents of all of the rights profiles on the system. For sample output, see [Chapter 6, “Listing Rights in Oracle Solaris”](#).
- The `profiles -p "Profile Name" info` command enables you to view the contents of a specific rights profile.
- The `profiles -l account-name` command enables you to view the contents of the rights profiles that are assigned to a specific user or role.



For more information, see [Chapter 6, “Listing Rights in Oracle Solaris”](#) and the [getent\(1M\)](#) and [profiles\(1\)](#) man pages.

## Authorizations Reference

An *authorization* is a discrete right that can be granted to a role or a user. Authorizations are checked by compliant applications before a user gets access to the application or specific operations within the application.

Authorizations are user-level, and therefore extensible. You can write a program that requires authorization, add the authorizations to your system, create a rights profile for these authorizations, and assign the rights profile to users or roles who are allowed to use the program.

### Authorization Naming Conventions

An authorization has a name that is used internally. For example, `solaris.system.date` is the name of an authorization. An authorization has a short description that appears in the graphical user interfaces (GUIs). For example, `Set Date & Time` is the description of the `solaris.system.date` 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 Oracle, 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.

As an example of how authorizations are used, the Network Link Security rights profile has the `solaris.network.link.security` authorization only, while the Network Security rights profile has the Network Link Security profile as a supplementary profile, plus the `solaris.network.*` and `solaris.smf.manage.ssh` authorizations.

### Delegation Authority in Authorizations

An authorization that ends with the suffix `delegate` enables a user or a role to delegate to other users any assigned authorizations that begin with the same prefix.

The `solaris_auth.delegate` authorization enables a user or a role to delegate to other users any authorizations that the delegating users or roles are assigned. For example, a role with the `solaris_auth.delegate` and `solaris.network.wifi.wep` authorizations can delegate the `solaris.network.wifi.wep` authorization to another user or role.

## Rights Databases

The following databases store the data for rights in Oracle Solaris:

- **Extended user attributes database** (`user_attr`) – Associates users and roles with authorizations, privileges, and rights profiles, among other keywords.
- **Rights profile attributes database** (`prof_attr`) – Defines rights profiles, lists the profiles' assigned authorizations, privileges, and keywords, 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 141](#).

## Rights Databases and the Naming Services

The [name service scope](#) of the rights databases is defined in the SMF service for the naming service switch, `svc:/system/name-service/switch`. The properties in this service for the rights databases are `auth_attr`, `password`, and `prof_attr`. The `password` property sets the naming service precedence for the `passwd` and `user_attr` databases. The `prof_attr` property sets the naming service precedence for the `prof_attr` and `exec_attr` databases.

In the following output, the `auth_attr`, `password`, and `prof_attr` entries are not listed. Therefore, the rights databases are using the files naming service.

```
# svccfg -s name-service/switch listprop config
config                               application
config/value_authorization          astring      solaris.smf.value.name-service.switch
config/default                       astring      files
config/host                          astring      "files ldap dns"
config/printer                       astring      "user files ldap"
```

## user\_attr Database

The `user_attr` database contains user and role information that supplements the `passwd` and `shadow` databases. The `attr` field contains security attributes and the `qualifier` field contains attributes that qualify or limit the effect of security attributes to a system or group of systems.

The security attributes in the `attr` field can be set by using the `roleadd`, `rolemod`, `useradd`, `usermod`, and `profiles` commands. They can be set locally and in the LDAP naming scope.

- For a user, the `roles` keyword assigns one or more defined roles.
- For a role, the `user` value to the `roleauth` keyword enables the role to authenticate with the user password rather than with the role password. By default, the value is `role`.
- For a user or role, the following attributes can be set:
  - `access_times` keyword – Specifies the days and times that specified applications and services can be accessed. For more information, see the [getaccess\\_times\(3C\)](#) man page.
  - `access_tz` keyword – Specifies the time zone to use when interpreting the times in `access_times` entries. For more information, see the [pam\\_unix\\_account\(5\)](#) man page.
  - `audit_flags` keyword – Modifies the audit mask. For more information, see the [audit\\_flags\(5\)](#) man page.
  - `auths` keyword – Assigns authorizations. For more information, see the [auths\(1\)](#) man page.
  - `auth_profiles` keyword – Assigns authenticated rights profiles. For reference, see the [profiles\(1\)](#) man page.
  - `defaultpriv` keyword – Adds privileges or removes them from the default [basic](#) set of privileges.
  - `limitpriv` keyword – Adds privileges or removes them from the default limit set of privileges.  
The `defaultpriv` and `limitpriv` privileges are always in effect because they are assigned to the user's initial process. For more information, see the [privileges\(5\)](#) man page and “[How Privileges Are Implemented](#)” on page 29.
  - `idlecmd` keyword – Logs out the user or locks the screen after `idletime` is reached.
  - `idletime` keyword – Sets the time that the system is available after no keyboard activity. Set `idletime` when you specify a value for `idlecmd`.
  - `lock_after_retries` keyword – If the value is `yes`, the system is locked after the number of retries exceeds the number that is allowed in the `/etc/default/login` file. For more information, see the [login\(1\)](#) man page. To unlock a locked account, see the [passwd\(1\)](#) man page.

- `profiles` keyword – Assigns rights profiles. For more information, see the [profiles\(1\)](#) man page.
- `project` keyword – Adds a default project. For more information, see the [project\(4\)](#) man page.

---

**Note** - Because the `access_times` and `access_tz` attributes are PAM attributes, they are checked during authentication. Therefore, they must be assigned either directly to a user or role, or in an authenticated rights profile. They are ignored in a regular rights profile.

---

The qualified attributes can be set for users and roles in the LDAP naming scope only. These qualifiers limit a user or role's attribute assignment, such as a rights profile, to one or more systems. For examples, see the [useradd\(1M\)](#) and [user\\_attr\(4\)](#) man pages.

The qualifiers are `host` and `netgroup`:

- `host` qualifier – Identifies the system where the user or role can perform specified actions.
- `netgroup` qualifier – Lists systems where the user or role can perform specified actions. `host` assignments have priority over `netgroup` assignments.

For more information, see the [user\\_attr\(4\)](#) man page. To view the contents of this database, use the `getent user_attr` command. For more information, see the [getent\(1M\)](#) man page and [Chapter 6, “Listing Rights in Oracle Solaris”](#).

## auth\_attr Database

The `auth_attr` database stores authorization definitions. Authorizations can be assigned to users, to roles, or to rights profiles. The preferred method is to place authorizations in a rights profile, then to assign the rights profile to a role or user.

To view the contents of this database, use the `getent auth_attr` command. For more information, see the [getent\(1M\)](#) man page and [Chapter 6, “Listing Rights in Oracle Solaris”](#).

## prof\_attr Database

The `prof_attr` database stores the name, description, help file location, privileges, 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 141](#).

For more information, see the [prof\\_attr\(4\)](#) man page. To view the contents of this database, use the `getent exec_attr` command. For more information, see the [getent\(1M\)](#) man page and [Chapter 6, “Listing Rights in Oracle Solaris”](#).

## 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 or users to whom the profile is assigned.

For more information, see the [exec\\_attr\(4\)](#) man page. To view the contents of this database, use the `getent` command. For more information, see the [getent\(1M\)](#) man page and [Chapter 6, “Listing Rights in Oracle Solaris”](#).

## policy.conf File

The `/etc/security/policy.conf` file provides a way of granting specific rights profiles, specific authorizations, and specific privileges to all users of a system. The relevant entries in the file consist of *key=value* pairs:

- `AUTHS_GRANTED=authorizations` – Refers to one or more authorizations.
- `AUTH_PROFS_GRANTED=rights profiles` – Refers to one or more authenticated rights profiles.
- `PROFS_GRANTED=rights profiles` – Refers to one or more rights profiles that are not authenticated.
- `CONSOLE_USER=Console User` – Refers to the Console User rights profile. This profile is delivered with a convenient set of authorizations for the console user. You can customize this profile.
- `PRIV_DEFAULT=privileges` – Refers to one or more privileges.
- `PRIV_LIMIT=privileges` – Refers to all privileges.

The following example shows some rights values from a `policy.conf` database:

```
##
AUTHS_GRANTED=
AUTH_PROFS_GRANTED=
CONSOLE_USER=Console User
PROFS_GRANTED=Basic Solaris User
#PRIV_DEFAULT=basic
```

#PRIV\_LIMIT=all

## Commands for Administering Rights

This section lists commands that are used to administer rights. It also includes a table of commands whose access can be controlled by authorizations.

### Commands That Manage Authorizations, Rights Profiles, and Roles

The commands listed in the following table retrieve and set rights on user processes.

**TABLE 3** Rights Administration Commands

| Command                             | Description  |
|-------------------------------------|--|
| <a href="#">auths(1)</a>            | Displays authorizations for a user. Creates new authorizations.  |
| <a href="#">getent(1M)</a>          | Lists the contents of the rights databases.  |
| <a href="#">nscd(1M)</a>            | Name service cache daemon, useful for caching the rights databases. Use the <code>svcadm</code> command to restart the daemon.   |
| <a href="#">pam_roles(5)</a>        | Role account management module for PAM. Checks for the authorization to assume a role.   |
| <a href="#">pam_unix_account(5)</a> | UNIX account management module for PAM. Checks for account restrictions, such as time restrictions and inactivity.               |
| <a href="#">pfbash(1)</a>           | Used to create a profile shell process that can evaluate rights.   |
| <a href="#">pfedit(1M)</a>          | Used to edit administrative files.   |
| <a href="#">pfexec(1)</a>           | Used to execute a command with security attributes.  |
| <a href="#">policy.conf(4)</a>      | Configuration file for system security policy. Lists granted authorizations, granted privileges, and other security information. |
| <a href="#">profiles(1)</a>         | Displays rights profiles for a specified user. Creates or modifies a rights profile.   |
| <a href="#">roles(1)</a>            | Displays roles that a specified user can assume.   |
| <a href="#">roleadd(1M)</a>         | Adds a role to a local system or to an LDAP network.   |
| <a href="#">roleadd(1M)</a>         | Adds a role to a local system or to an LDAP network.   |
| <a href="#">rolemod(1M)</a>         | Modifies a role's properties on a local system or on an LDAP network.  |
| <a href="#">userattr(1)</a>         | Displays the value of a specific right that is assigned to a user or role account.   |
| <a href="#">useradd(1M)</a>         | Adds a user account to the system or to an LDAP network. The <code>-R</code> option assigns a role to a user's account.          |
| <a href="#">userdel(1M)</a>         | Deletes a user's login from the system or from an LDAP network.  |

| Command                  | Description   |
|--------------------------|---|
| <code>usermod(1M)</code> | Modifies a user's account properties on the system. |

## Selected Commands That Require Authorizations

The following table provides examples of how authorizations are used to limit command options on an Oracle Solaris system. For more discussion of authorizations, see [“Authorizations Reference” on page 137](#).

**TABLE 4** Commands and Associated Authorizations

| Command                      | Authorization Requirements   |
|------------------------------|--|
| <code>at(1)</code>           | <code>solaris.jobs.user</code> required for all options (when neither <code>at.allow</code> nor <code>at.deny</code> files exist)  |
| <code>atq(1)</code>          | <code>solaris.jobs.admin</code> required for all options   |
| <code>cdrw(1)</code>         | <code>solaris.device.cdrw</code> required for all options, which is granted by default in the <code>policy.conf</code> file  |
| <code>crontab(1)</code>      | <code>solaris.jobs.user</code> required for the option to submit a job (when neither <code>crontab.allow</code> nor <code>crontab.deny</code> files exist)   |
|                              | <code>solaris.jobs.admin</code> required for the options to list or modify other users' <code>crontab</code> files   |
| <code>allocate(1)</code>     | <code>solaris.device.allocate</code> (or other authorization as specified in <code>device_allocate</code> file) required to allocate a device  |
|                              | <code>solaris.device.revoke</code> (or other authorization as specified in <code>device_allocate</code> file) required to allocate a device to another user (-F option)  |
| <code>deallocate(1)</code>   | <code>solaris.device.allocate</code> (or other authorization as specified in <code>device_allocate</code> file) required to deallocate another user's device   |
|                              | <code>solaris.device.revoke</code> (or other authorization as specified in <code>device_allocate</code> ) required to force deallocation of the specified device (-F option) or all devices (-I option)  |
| <code>list_devices(1)</code> | <code>solaris.device.revoke</code> required to list another user's devices (-U option)   |
| <code>roleadd(1M)</code>     | <code>solaris.user.manage</code> required to create a role. <code>solaris.account.activate</code> required to set the initial password. <code>solaris.account.setpolicy</code> required to set <a href="#">password policy</a> , such as account locking and password aging. |
| <code>roledel(1M)</code>     | <code>solaris.passwd.assign</code> authorization required to delete the password.  |
| <code>rolemod(1M)</code>     | <code>solaris.passwd.assign</code> authorization required to change the password. <code>solaris.account.setpolicy</code> required to change password policy, such as account locking and password aging.   |
| <code>sendmail(1M)</code>    | <code>solaris.mail</code> required to access mail subsystem functions; <code>solaris.mail.mailq</code> required to view mail queue   |
| <code>useradd(1M)</code>     | <code>solaris.user.manage</code> required to create a user. <code>solaris.account.activate</code> required to set the initial password. <code>solaris.account.setpolicy</code> required to set password policy, such as account locking and password aging.                  |

| Command                     | Authorization Requirements   |
|-----------------------------|--|
| <a href="#">userdel(1M)</a> | solaris.passwd.assign authorization required to delete the password.   |
| <a href="#">usermod(1M)</a> | solaris.passwd.assign authorization required to change the password. solaris.account.setpolicy required to change password policy, such as account locking and password aging. |

## Privileges Reference

Privileges restrict processes are implemented in the kernel, and can restrict processes at the command, user, role, or system level.

## Commands for Handling Privileges

The following table lists the commands that are available to handle privileges.

**TABLE 5** Commands for Handling Privileges

| Purpose   | Command                                  | Man Page                         |
|---|--|----------------------------------|
| Debug privilege failure                                   | <code>ppriv -eD failed-operation</code>  | <a href="#">ppriv(1)</a>         |
| List the privileges on the system                         | <code>ppriv -l</code>                    | <a href="#">ppriv(1)</a>         |
| List a privilege and its description                      | <code>ppriv -lv priv</code>              | <a href="#">ppriv(1)</a>         |
| List extended privilege policy on a UID, process, or port | <code>ppriv -lv extended-policy</code>   | <a href="#">ppriv(1)</a>         |
| Examine process privileges                                | <code>ppriv -v pid</code>                | <a href="#">ppriv(1)</a>         |
| Add extended privilege policy to a UID, process, or port  | <code>ppriv -r rule</code>               | <a href="#">privileges(5)</a>    |
| Set process privileges                                    | <code>ppriv -s spec</code>               | <a href="#">ppriv(1)</a>         |
| Remove an extended privilege policy rule                  | <code>ppriv -X rule</code>               | <a href="#">privileges(5)</a>    |
| Assign privileges to a rights profile                     | <code>profiles -p profile-name</code>    | <a href="#">profiles(1)</a>      |
| Assign privileges to a new role                           | <code>roleadd -K defaultpriv=</code>     | <a href="#">roleadd(1M)</a>      |
| Add privileges to an existing role                        | <code>rolemod -K defaultpriv+=</code>    | <a href="#">rolemod(1M)</a>      |
| Assign privileges to a new user                           | <code>useradd -K defaultpriv=</code>     | <a href="#">useradd(1M)</a>      |
| Add privileges to an existing user                        | <code>usermod -K defaultpriv+=</code>    | <a href="#">usermod(1M)</a>      |
| Add device policy to a device                             | <code>add_drv -p policy driver</code>    | <a href="#">add_drv(1M)</a>      |
| Set device policy   | <code>devfsadm</code>                    | <a href="#">devfsadm(1M)</a>     |
| View device policy  | <code>getdevpolicy</code>                | <a href="#">getdevpolicy(1M)</a> |
| Update device policy on open devices                      | <code>update_drv -p policy driver</code> | <a href="#">update_drv(1M)</a>   |



## Files That Contain Privilege Information

The `policy.conf` and `syslog.conf` files contain information about privileges.

- `/etc/security/policy.conf` contains the following privilege information:
  - `PRIV_DEFAULT` – Inheritable set of privileges for the system
  - `PRIV_LIMIT` – Limit set of privileges for the system

For more information, see the [policy.conf\(4\)](#) man page.

- `/etc/syslog.conf` is the system log file for debug messages that are related to privilege debugging. The path for debug messages is set in the `priv.debug` entry.

For more information, see the [syslog.conf\(4\)](#) man page.

## Privileged Actions in the Audit Record

Privilege use can be audited. Any time that a process uses a privilege, the use of privilege is recorded in the audit trail in the `upriv` audit token. When privilege names are part of the record, their textual representation is used. The following audit events record use of privilege:

- **AUE\_SETPPRIV audit 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** – Generates an audit record when a privilege is added from outside the kernel. The `AUE_MODALLOCPRIV` audit event is in the `ad` class.
- **AUE\_MODDEVPLCY audit event** – Generates an audit record when the device policy is changed. The `AUE_MODDEVPLCY` audit event is in the `ad` class.
- **AUE\_PFEEXEC audit event** – Generates an audit record when a call is made to `execve()` with `pfexec()` enabled. The `AUE_PFEEXEC` audit event is in the `as`, `ex`, `ps`, and `ua` audit classes. The names of the privileges are included in the audit record.

The successful use of privileges that are in the [basic set](#) is not audited. An attempt to use a basic privilege that has been removed from a user's basic set is audited.



## Security Glossary

---

|                                     |  |
|-------------------------------------|--|
| <b>authenticated rights profile</b> | A <a href="#">rights profile</a> that requires the assigned user or role to type a password before executing an operation from the profile. This behavior is similar to sudo behavior. The length of time that the password is valid is configurable.  |
| <b>authentication</b>               | The process of verifying the claimed identity of a login or process.   |
| <b>authorization</b>                | A right that can be assigned to a role or user (or embedded in a rights profile) for performing a class of operations that are otherwise prohibited by security policy. Authorizations are enforced at the user application level, not in the kernel.  |
| <b>basic set</b>                    | 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.  |
| <b>effective set</b>                | The set of privileges that are currently in effect on a process.   |
| <b>inheritable set</b>              | The set of privileges that a process can inherit across a call to exec.  |
| <b>least privilege</b>              | A security model which gives a specified process only a subset of superuser powers. The least privilege model assigns enough privilege to regular users that they can perform personal administrative tasks, such as mount file systems and change the ownership of files. On the other hand, processes run with just those privileges that they need to complete the task, rather than with the full power of superuser, that is, all privileges. Damage due to programming errors like buffer overflows can be contained to a non-root user, which has no access to critical abilities like reading or writing protected system files or halting the system. |
| <b>limit set</b>                    | The outside limit of what privileges are available to a process and its children.  |
| <b>name service scope</b>           | The scope in which a role is permitted to operate, that is, an individual host or all hosts that are served by a specified naming service such as files, NIS, or LDAP.   |
| <b>password policy</b>              | 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 password attempts are permitted, and other security considerations. Security policy requires passwords. Password policy might require passwords to be encrypted with the AES algorithm, and might make further requirements related to password strength.  |

|                                     |  |
|-------------------------------------|--|
| <b>permitted set</b>                | The set of privileges that are available for use by a process.   |
| <b>policy</b>                       | 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 passwords be changed every six weeks. See also <a href="#">password policy</a> and <a href="#">rights policy</a> .   |
| <b>principle of least privilege</b> | See <a href="#">least privilege</a> .  |
| <b>privilege</b>                    | <ol style="list-style-type: none"><li>1. In general, a power or capability to perform an operation on a computer system that is beyond the powers of a regular user. Superuser privileges are all the <a href="#">rights</a> that superuser is granted. A privileged user or privileged application is a user or application that has been granted additional rights.</li><li>2. A discrete right on a process in an Oracle Solaris system. Privileges offer a finer-grained control of processes than does root. Privileges are defined and enforced in the kernel. Privileges are also called <i>process privileges</i> or <i>kernel privileges</i>. For a full description of privileges, see the <a href="#">privileges(5)</a> man page.</li></ol> |
| <b>privilege escalation</b>         | Gaining access to resources that are outside the range of resources that your assigned rights, including rights that override the defaults, permit. The result is that a process can perform unauthorized operations.  |
| <b>privilege model</b>              | See <a href="#">rights model</a> .   |
| <b>privilege set</b>                | <p>A collection of privileges. Every process has four sets of privileges that determine whether a process can use a particular privilege. See <a href="#">limit set</a>, <a href="#">effective set</a> set, <a href="#">permitted set</a> set, and <a href="#">inheritable set</a> set.</p> <p>Also, the <a href="#">basic set</a> set of privileges is the collection of privileges that are assigned to a user's process at login.</p>   |
| <b>privilege-aware</b>              | Programs, scripts, and commands that turn on and off the use of privilege in their code. In a production environment, the privileges that are turned on must be supplied to the process, for example, by requiring users of the program to use a rights profile that adds the privileges to the program. For a full description of privileges, see the <a href="#">privileges(5)</a> man page.   |
| <b>privileged application</b>       | An application that can override system controls. The application checks for security attributes, such as specific UIDs, GIDs, authorizations, or privileges.  |
| <b>privileged user</b>              | A user who is assigned <a href="#">rights</a> beyond the rights of regular user on a computer system. See also <a href="#">trusted users</a> .   |

|                            |   |
|----------------------------|---|
| <b>profile</b>             | See <a href="#">rights profile</a> .  |
| <b>profile shell</b>       | In rights management, 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 shell versions correspond to the available shells on the system, such as the pfbash version of bash.  |
| <b>RBAC</b>                | Role-based access control, the user rights management feature of Oracle Solaris. See <a href="#">rights</a> .   |
| <b>RBAC policy</b>         | See <a href="#">rights policy</a> .   |
| <b>reauthentication</b>    | The requirement to provide a password to perform a computer operation. Typically, sudo operations require reauthentication. Authenticated rights profiles can contain commands that require reauthentication. See <a href="#">authenticated rights profile</a> .  |
| <b>rights</b>              | An alternative to the all-or-nothing superuser model. User rights management and process rights management enable an organization to divide up superuser's privileges and assign them to users or roles. Rights in Oracle Solaris are implemented as kernel privileges, authorizations, and the ability to run a process as a specific UID or GID. Rights can be collected in a <a href="#">rights profile</a> and a role.  |
| <b>rights model</b>        | A stricter model of security on a computer system than the superuser model. In the rights 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.  |
| <b>rights policy</b>       | The security policy that is associated with a command. Currently, solaris is the valid policy for Oracle Solaris. The solaris policy recognizes privileges and extended privilege policy, authorizations, and setuid security attributes.   |
| <b>rights profile</b>      | Also referred to as a <a href="#">profile</a> . A collection of security overrides that can be assigned to a role or user. A rights profile can include authorizations, privileges, commands with security attributes, and other rights profiles that are called supplementary profiles.  |
| <b>roles</b>               | Accounts with <a href="#">rights</a> that you create and assign to <a href="#">trusted users</a> to perform administrative tasks. The armor package contains seven predefined roles.  |
| <b>security attributes</b> | 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 root 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 <a href="#">privilege model</a> , kernel privileges and other <a href="#">rights</a> replace setuid root programs as security attributes. 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 policy** See [policy](#).

**separation of duty** Part of the notion of [least privilege](#). Separation of duty prevents one user from performing or approving all operations that complete a transaction. For example, in [RBAC](#), you can separate the creation of a login user from the assignment of security overrides. One role creates the user. A separate role can assign security attributes, such as rights profiles, roles, and privileges to existing users.

**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 system. Typically, to administer the system, a user becomes superuser (root) and can do all administrative activities.

**trusted users** Users whom you have decided can perform administrative tasks at some level of trust. Typically, administrators create logins for trusted users first and assign administrative rights that match the users' level of trust and ability. These users then help configure and maintain the system. Also called *privileged users*.

# Index

---

## Numbers and Symbols

\$\$ (double dollar sign)

parent shell process number, 119

removing basic privilege from your process, 74

\* (asterisk)

checking for in authorizations, 84

wildcard character

in authorizations, 137

+ (plus sign)

keyword modifier, 57

- (minus sign)

keyword modifier, 57

. (dot)

authorization name separator, 137

{ } (curly braces)

extended privileges syntax, 64, 65, 86, 87

## A

access

controlling application access to specified

directories, 92

enabling to restricted files, 64, 100, 105

limiting port privileges, 86

restricting guest access to system, 76

access\_times keyword, 20, 139

access\_tz keyword, 20, 139

account locking, 66

accounts

locking and unlocking, 68

adding

auditing of privileged actions, 102

authorizations

to rights profile, 108

to role, 61

to user, 61

cryptomgt role, 55

extended privileges

by users, 92

to a database, 87

to a port, 86

to a web server, 90

new authorization, 107

new rights profile, 103

new rights profile from existing one, 105

privileges

directly to role, 58

directly to user, 61

to command in rights profile, 104

rights

commands for, 142

to legacy applications, 83

to rights profile, 103

to roles, 52

to users, 59

rights profiles to list of profiles, 57

roles, 47

security-related role, 55

set ID

to legacy applications, 83

trusted users, 60

administering

ARMOR roles, 53

authorizations, 107, 107

extended privilege policy, 85

immutable zones, 95

rights

- authorizations, 107
- commands for, 142
- instructions, 98
- legacy applications, 83, 83
- of a role, 52, 58, 62
- of a user, 59, 66
- rights profiles, 103
- roles, 128
- rights profiles, 63, 103, 129
- role password, 52, 58
- roles to replace superuser, 44
- user password to assume role, 62, 128
- without privileges, 28
- administrative accounts
  - creating roles for, 54
- administrators
  - adding to users' rights, 59
  - installing ARMOR package, 54
  - restricting access to a database, 87
  - restricting access to a port, 86
  - restricting rights, 78
  - restricting users' rights, 66
  - restricting web server privileges, 90
- All rights profile, 136
- allocate command
  - authorizations required for, 143
- Apache HTTP Server
  - assigning extended privileges, 90
  - verifying use of privilege, 91
- applications
  - Apache HTTP Server, 90
  - assigning extended privileges, 94
  - assigning extended privileges to editors, 75
  - checking for authorizations, 84
  - Firefox browser, 93
  - legacy and privileges, 32
  - limiting access to specified directories, 94
  - MySQL database, 87
  - preventing from spawning new processes, 79
  - privilege-aware, 29, 31
- ARMOR
  - assigning roles to trusted users, 53
  - installing package, 54
  - introduction to standard, 17
  - planning use of, 45
- assigning
  - authorizations in a rights profile, 108
  - privileges
    - to commands in a rights profile, 104
    - to commands in a script, 82
    - to role, 58
    - to user, 61
  - profile shell as login shell, 56, 60
- rights
  - securely, 40
  - to specific resources, 85
  - to users, 16
  - usability considerations, 41
- rights profile
  - to a role, 52
  - to a user, 60
- rights profiles, 69, 74
- rights to users
  - to users, 59, 66
  - role to a user locally, 52
- assuming role
  - how to, 59
  - in a terminal window, 101
  - root, 101
  - when assigned, 98
- asterisk (\*)
  - checking for in authorizations, 84
  - wildcard character
    - in authorizations, 137
- at command
  - authorizations required for, 143
- atq command
  - authorizations required for, 143
- audit command
  - s option, 102
- Audit Configuration rights profile
  - use of, 102
- audit\_flags keyword
  - description, 139
- auditing
  - privileges and, 145



roles, 102

auth\_attr database, 138, 140

auth\_profiles keyword  
 description, 139  
 example of, 61

AUTH\_PROFS\_GRANTED keyword  
 policy.conf file, 141

authenticated rights profiles  
 assigning, 61  
 keyword in policy.conf file, 141  
 searched before rights profiles, 38, 125

authorizations, 15  
*See also* rights  
 adding to rights profile, 108  
 checking for wildcards, 84  
 checking in privileged application, 40  
 commands requiring, 143  
 compared to privileges, 20, 23  
 creating new ones, 107  
 database, 138, 140  
 delegating, 137  
 description, 20, 23, 137  
 effect of misspelling, 125  
 granularity, 137  
 listing, 114  
 misspelling, 125  
 naming conventions, 137  
 preventing privilege escalation, 35  
 removing from rights profile, 106  
 troubleshooting, 124

auths command  
 description, 142  
 -t option, 108  
 use, 84, 107, 114

auths keyword  
 description, 108, 139  
 use, 105, 106

AUTHS\_GRANTED keyword  
 policy.conf file, 141

## B

basic privilege set, 30

basic privileges  
 limiting use by service, 87

Basic Solaris User rights profile, 136

browsers  
 protecting user files with extended privileges, 92

## C

capabilities *See* rights

cdwr command  
 authorizations required for, 143

changing  
 password of role, 52, 58  
 rights  
 of a port, 86  
 of a script, 82  
 of a web server, 90  
 of an application, 81  
 of an editor, 75  
 of Firefox, 92  
 of role, 52  
 to MySQL database, 87  
 rights profile contents, 103  
 root role into user, 109  
 umask, 67  
 user file permissions, 67

cloning  
 rights profile contents, 105

commands  
 determining user's privileged commands, 118  
 determining user's qualified attributes, 121  
 for administering privileges, 144  
 rights administration commands, 142  
 that assign privileges, 33  
 that check for privileges, 39

components  
 rights management, of, 20

configuration files  
 policy.conf file, 142  
 syslog.conf file, 145  
 with privilege information, 145

configuring  
 authorizations, 107

- power management, 70
- privileged users, 60
- protected database, 87
- protected port, 86
- protected web server, 90
- protection of user files from applications, 92
- restricted users, 66
- rights, 44, 59, 66
- rights profiles, 103
- roles, 47, 52
  - root role as user, 109
  - trusted users, 52
- Console User rights profile, 70, 136
- CONSOLE\_USER keyword
  - policy.conf file, 141
- creating
  - ARMOR roles, 53
  - authorization, 107
  - privileged users, 60
  - rights profiles, 63, 103
  - roles, 47
  - root user, 109
- crontab files
  - authorizations required for, 143
- Crypto Management rights profile
  - using in a role, 55
- Cryptographic Framework
  - administering with role, 55
- curly braces ({})
  - extended privileges syntax, 64, 65, 86, 87
- D**
- daemons
  - nscd (name service cache daemon), 142
  - running with privileges, 28
- databases
  - auth\_attr, 140
  - exec\_attr, 141
  - MySQL, 87
  - prof\_attr, 140
  - protecting with extended privileges, 87
  - rights, 138
  - user\_attr, 139
- dax\_access privilege, 15
- deallocate command
  - authorizations required for, 143
- defaultpriv keyword, 72
  - description, 139
- defaults
  - privileges settings in policy.conf file, 145
- delegating authorizations, 137
- determining
  - Apache HTTP Server's privileges, 91
  - privileges on a process, 119
  - required privileges, 129
  - rights, available or assigned, 113
  - which rights model to use, 43
- devices
  - rights model and, 31
  - superuser model and, 31
- displaying
  - roles you can assume, 102, 142
- dot (.)
  - authorization name separator, 137
- double dollar sign (\$\$)
  - parent shell process number, 119
  - removing basic privilege from your shell, 74
- E**
- editors
  - preventing from spawning new processes, 75
  - restricting for guest user, 75
- effective privilege set, 29
- escalation of privilege
  - description, 35
  - preventing in devices, 31
- /etc/default/login file, 69
- /etc/security/policy.conf file
  - editing, 68, 71, 73, 74
- exacct files
  - reading with Perl scripts, 64
- exec\_attr database, 138, 141
- expanding users rights, 59

Extended Accounting Net Management rights profile, 64  
 extended policy *See* extended privileges  
 extended privilege policy *See* extended privileges  
 extended privileges  
   administering, 85  
   assigned by regular users, 92  
   assigning  
     in rights profile, 75  
     to a database, 87  
     to a port, 86  
     to trusted users, 64  
     to web server, 90  
   description, 34, 35  
   listing, 89  
   PRIV\_XPOLICY flag, 89  
   protecting files of regular users, 92  
   reading root-owned files, 65

**F**

FILE privileges  
   description, 27  
   file\_chown, 31  
   file\_chown\_self, 36  
 files  
   /etc/default/login, 69  
   containing privilege information, 145  
   privileges relating to, 27  
 Firefox browser  
   assigning extended privileges, 93  
 flags  
   PRIV\_PFEEXEC in profile shells, 127  
   PRIV\_XPOLICY on process, 89

**G**

getent command  
   description, 142  
   listing commands with assigned security attributes, 118  
   listing contents of rights databases, 113

  listing definitions of all authorizations, 114  
   listing definitions of all rights profiles, 115  
   listing qualified security attributes, 121  
   using, 110

**H**

hardware  
   restricting user control of, 70  
 host qualified attribute  
   description, 140

**I**

idlecmd keyword  
   description, 139  
   use, 124  
 idletime keyword  
   description, 139  
   use, 124  
 immutable zones  
   administering, 95  
 inheritable privilege set, 29  
 IPC privileges, 27  
 IPS packages *See* packages

**K**

kernel processes and privileges, 26  
 keywords  
   defaultpriv, 72  
   lock\_after\_retries, 69  
   RETRIES, 69

**L**

ldapaddent command  
   listing all qualified security attributes, 121  
 least privilege

- principle of, 27
- legacy applications and privileges, 32, 83
- limit privilege set, 30
- limitpriv keyword, 139
- Linux behaviors
  - sudo command, 98, 132
  - user password when assuming role, 58, 62, 63
- list\_devices command
  - authorizations required for, 143
- listing
  - all rights, 113
  - authorizations, 114
  - default rights configuration, 113
  - privileges, 118
  - qualifiers to security attributes, 121
  - rights, 113
  - rights of initial user, 113
  - rights profiles, 115
  - roles, 117
  - roles you can assume, 102, 142
  - your rights, 113
- lock\_after\_retries keyword, 69
  - description, 139
- locking
  - accounts, 66
  - user account automatically, 68
- logging in
  - remote root login, 109
  - users' basic privilege set, 30

## M

- man pages
  - commands that require authorizations, 143
  - rights, 142
- managing *See* administering
- Media Backup rights profile
  - assigning to trusted users, 19
- Media Restore rights profile
  - preventing privilege escalation, 35
- minus sign (-)
  - keyword modifier, 57
- modifying *See* changing

- monitoring
  - use of privileged commands, 102
- MySQL database
  - installing IPS package, 87
  - protecting with extended privileges, 87

## N

- naming conventions
  - authorizations, 137
- naming services
  - rights databases and, 138
  - scope of assigned rights, 37
- NET privileges, 27
- netgroup qualified attribute
  - description, 140
- network
  - privileges relating to, 27
- Network IPsec Management rights profile
  - adding solaris.admin.edit authorization, 105
- non-UNIX accounts
  - troubleshooting password assignments, 132
- nscd (name service cache daemon)
  - use, 142

## O

- Object Access Management rights profile, 31
- obtaining
  - privileged commands, 52
  - privileges, 31, 33, 58, 61
  - privileges on a process, 119
- one-time passwords
  - requiring use of, 66
- one-time passwords (OTP), 15
- Operator rights profile
  - assigning to role, 18
  - description, 135
- order of search
  - authenticated rights profiles, 38
  - rights, 38
  - rights profiles example, 57

user security attributes, 38

## P

packages

ARMOR, 54

MySQL, 87

PAM

adding su stack to configuration file, 100

modules, 100

stack to cache authentication, 100

time-sensitive user access, 20, 139

pam\_roles module, 142

pam\_tty\_tickets module, 100

pam\_unix\_account module, 142

passwd command

changing password of role, 52, 58

NP accounts, 132

passwords

changing role password, 52, 58

locking out users, 68

overriding constraints, 133

unlocking user, 70

using user's to assume role, 62, 128

Perl scripts

for extended accounting, 64

permissions

changing user file permissions, 67

permissive security policy

components of, 20

creating, 59

permitted privilege set, 29

pfbash command, 142

pfedit command, 100, 142

pfexec command, 99, 142

planning

ARMOR role use, 45

rights model use, 44

use of rights, 44

plus sign (+)

keyword modifier, 57

policy.conf file

description, 141

keywords

for authenticated rights profiles, 141

for authorizations, 141

for privileges, 141, 145

for rights profiles, 141

for workstation owner, 141

ports

protecting with extended privileges, 86

power management

configuring, 70

powers *See* rights

ppriv command, 118, 119, 144

-eD option, 82, 129

-r option, 92

-s option, 94

predefined roles

ARMOR standard, 17, 53

planning use of, 45

principle of least privilege, 27

Printer Management rights profile, 136

priv.debug entry

syslog.conf file, 145

PRIV\_DEFAULT keyword

policy.conf file, 141

PRIV\_LIMIT keyword

policy.conf file, 141, 145

PRIV\_PFEEXEC flag, 127

PRIV\_PROC\_LOCK\_MEMORY privilege, 32

PRIV\_XPOLICY flag, 89

privilege checking, 39

privilege sets

adding privileges to, 34, 58, 61

basic, 30, 120, 126

effective, 29

inheritable, 29

limit, 30, 126

listing, 30, 119

permitted, 29

removing privileges from, 34, 35, 74, 77, 78, 104

privileged application

authorization checking, 40

checking for security attributes, 39

description, 20

- ID checking, 39
- privilege checking, 39
- privileged users *See* trusted users
- privileges
  - adding to command in rights profile, 104
  - assigning
    - to a command, 33
    - to a script, 35
    - to a user, 33
    - to Apache HTTP Server, 90
    - to MySQL database, 87
    - to role, 58
    - to user, 61
  - auditing and, 145
  - categories, 27
  - checking in applications, 39
  - commands, 144
  - compared to authorizations, 20, 23
  - compared to superuser model, 25
  - dax\_access, 15
  - debugging, 33, 145
  - description, 20, 27, 28
  - devices and, 31
  - differences from superuser model, 28
  - escalation prevention at user level, 35
  - escalation prevention in kernel, 36
  - expanding user or role's, 33
  - extended privilege policy, 34, 35
  - files, 145
  - finding missing, 130
  - implemented in sets, 29
  - inherited by processes, 31
  - legacy applications and, 32, 83
  - limiting users, 72
  - listing on a process, 119
  - PRIV\_PROC\_LOCK\_MEMORY, 32
  - processes with assigned privileges, 31
  - programs aware of privileges, 31
  - protecting kernel processes, 26
  - removing
    - basic privilege, 78
    - basic privilege from your process, 74
    - from a rights profile, 78
    - from a user, 34
    - from a user's limit set, 74
    - from yourself, 74
  - removing basic, 72
  - troubleshooting
    - lack of, 129
    - user assignment, 124
  - using in shell script, 82
- privileges keyword
  - listing, 118
- PROC privileges
  - description, 27
  - proc\_owner, 31
- process privileges, 27
- process rights management *See* privileges, rights
- prof\_attr database, 140
  - summary, 138
- profile shells
  - assigning to users, 56
  - description, 37
  - determining if PRIV\_PFEEXEC flag is set, 127
  - login shells for trusted users, 60
  - opening, 98
  - reading exact network files, 64
  - restricting rights, 78
- profiles *See* rights profiles
- profiles command
  - creating rights profiles, 103
  - description, 142
  - listing user's authenticated rights profiles, 115
  - listing user's rights profiles, 113
  - use, 115
- profiles keyword
  - description, 140
  - listing, 115
- PROFS\_GRANTED keyword
  - policy.conf file, 141
- programs *See* applications
- project.max-locked-memory resource control, 32
- pwhash command, 15

**Q**

qualified user attributes  
 description, 25  
 overview, 21  
 qualifier attribute  
 listing, 121  
 user\_attr database, 140

**R**

-R option

useradd command, 142

removing

basic privilege from application, 87, 92  
 basic privilege from rights profile, 78  
 basic privilege from yourself, 74  
 basic privileges from a rights profile, 78  
 limit privilege from user, 74  
 power management capability from users, 70  
 privileges from a user, 72  
 role assignments, 109  
 users' rights, 66

replacing

keyword values, 57, 61  
 root role with root user, 109  
 root user with root role, 110  
 superuser with roles, 44

resource controls

privileges, and, 32  
 project.max-locked-memory, 32  
 zone.max-locked-memory, 32

restricted files

enabling read access to, 64  
 enabling write access to, 100, 105

restricting

access to computer by time and day, 20  
 database privileges, 87  
 editor of guest user, 75  
 guest access to system, 76  
 login attempts, 66  
 port privileges, 86  
 rights in a rights profile, 77, 78, 104  
 user control of hardware, 70

user file permissions, 67  
 web server privileges, 90

restrictive security policy

components of, 20  
 creating, 66  
 enforcing, 85

RETRIES keyword, 69

rights, 15

*See also* authorizations, privileges, rights profiles, roles

access\_times keyword, 20

access\_tz keyword, 20

account locking, 66

adding privileged users, 60

administration commands, 142

assigning, 59

authenticated rights profiles, 61

to restrict users, 66

to users, 47

auditing use of, 102

authorization database, 140

authorizations, 23

basic concepts, 20

changing role passwords, 52, 58

checking for, 37, 39

checking scripts or programs for authorizations, 84

commands for, 142

commands for managing, 142

compared to superuser model, 16

configuring, 59, 66

considerations when directly assigning, 40

creating authorizations, 107

creating rights profiles, 103

databases, 138

defaults, 113

elements, 20

expanding users, 59

gaining administrative, 98

limiting login attempts, 66

listing all, 113

modifying roles, 52

naming services and, 138

Network Security rights profile, 22

- new features in this release, 15
- order of search, 38
- planning use of, 44
- privileges on commands, 39
- profile shells, 37
- reading exact network files, 64, 64
- recommended roles, 16
- removing from users, 66
- restricting administrator to explicitly assigned, 78
- restricting rights, 78
- restricting users to specific times of access, 20
- restricting users', 66
- rights profile database, 140
- rights profiles, 23
- search order, 38
- securing scripts, 82
- security considerations when assigning, 40
- special ID on commands, 39
- troubleshooting, 124
- usability considerations when assigning, 41
- using user password to assume role, 62, 128
- viewing all, 113
- viewing your, 113
- rights management *See* privileges, rights
- rights profiles
  - adding privileges to command, 104
  - adding `solaris.admin.edit` authorization, 105
  - All, 136
  - assigning
    - to users, 60
  - assigning to trusted users, 19
  - authenticating with user's password, 63, 129
  - Basic Solaris User, 136
  - changing contents of, 103
  - cloning contents of, 105
  - compared to roles, 24
  - Console User, 38, 70, 71, 136
  - contents of typical, 135
  - creating, 103
  - creating and assigning, 69, 73
  - creating for remote users, 77
  - creating for Sun Ray users, 104
  - databases *See* `exec_attr` database, `prof_attr` database
  - description, 20, 23
  - Extended Accounting Net Management, 64
  - first in list, 57
  - major rights profiles descriptions, 135
  - modifying, 103
  - Network IPsec Management, 105
  - Object Access Management, 31
  - Operator, 135
  - order of search, 38
  - preventing privilege escalation, 19, 35
  - Printer Management, 136
  - removing authorizations, 106
  - restricting basic privileges, 78
  - restricting rights of all users of a system, 77
  - Stop, 38, 136
  - System Administrator, 135
  - third-party applications, 63
  - troubleshooting, 124
  - viewing contents, 136
  - VSCAN Management, 106
- role-based access control (RBAC) *See* rights
- `roleadd` command
  - authorizations required for, 143
  - description, 142, 142
  - example of using, 55
  - P option, 100
  - s option, 55
  - S option, 55
- `roleauth` keyword
  - example of using, 58, 62, 64
  - passwords for roles, 62, 128
  - use, 100
- `roledel` command
  - authorizations required for, 143
  - example of using, 59
- `rolemod` command
  - assigning rights to a role, 57
  - authorizations required for, 143
  - changing rights of role, 57
  - description, 142
  - example of using, 58, 62



- K option, 109
- passwords for roles, 62, 128
- roles
  - ARMOR, 17
  - assigning
    - privileges to, 58
    - rights, 47
    - with usermod command, 52
  - assuming
    - after login, 24
    - ARMOR, 101
    - in a terminal window, 37, 101
    - root role, 101
    - to use assigned rights, 98
  - auditing, 102
  - authenticating with user's password, 62, 128
  - changing password of, 52, 58
  - changing properties of, 52
  - compared to rights profiles, 24
  - configured like sudo, 132
  - creating, 47
  - creating ARMOR, 53
  - creating for administrative accounts, 54
  - deleting, 59
  - description, 24
  - determining directly assigned privileges, 62
  - determining role's privileged commands, 127
  - listing local roles, 102, 142
  - making root role into user, 109
  - modifying, 52
  - planning predefined, 45
  - predefined, 17, 53
  - removing assignment from users, 109
  - separation of duty, 55, 102
  - summary, 21
  - use in user rights assignment, 16
  - using an assigned role, 101
  - using user password, 22, 63
  - with user passwords, 132
- roles command
  - description, 142
  - using, 102
- roles keyword

- listing, 117
- root role
  - assuming role, 101
  - changing from root user, 110
  - changing to root user, 109
  - created at installation, 18
  - description, 18
  - overriding password constraints, 133
  - secure remote login, 109
  - troubleshooting, 111
- root user
  - changing into root role, 110
  - replacing in rights model, 24

**S**

- applications
  - protecting administrative accounts, 54
- S option
  - profiles command, 104
- scope of assigned rights, 37
- scripts
  - checking for authorizations, 84
  - for extended accounting, 64
  - Perl scripts, 64
  - running with privileges, 35
  - securing, 82
  - use of privileges in, 82
- security attributes, 15
  - See also* rights
  - description, 20
  - qualified, 21, 25
- security policy
  - default rights, 138
  - restrictive and permissive, 20
- security properties *See* rights
- sendmail command
  - authorizations required for, 143
- separation of duty
  - security and non-security roles, 55
  - two roles to handle auditing, 102
- shell commands
  - passing parent shell process number, 119

- shells
    - determining if privileged, 127
    - listing privileges on process, 119
    - privileged versions, 37
    - troubleshooting if profile, 126
    - usability considerations, 41
    - writing privileged scripts, 82
  - smart cards, 15
  - solaris.\*.assign authorizations
    - preventing privilege escalation, 35
  - solaris.admin.edit authorization
    - adding to rights profile, 105
  - solaris.smf.value authorization
    - removing from rights profile, 106
  - Stop rights profile, 136
  - su command
    - becoming root, 109
    - changing to a role, 55
    - in role assumption, 101
  - subshells
    - restricting editing rights, 75
  - sudo
    - roles configured like, 132
  - sudo command
    - using in Oracle Solaris, 44, 98
  - superuser
    - compared to rights model, 16, 25
    - differences from rights model, 28
    - eliminating by delegating rights, 24
    - troubleshooting becoming root as a role, 111
  - svc:/application/database/mysql:version\_55, 87
  - svc:/network/http:Apache2, 90
  - svc:/system/name-service/switch, 37, 124
  - svccfg command
    - s option, 90, 124
  - svcprop command
    - s option, 88
  - SYS privileges, 27
  - syslog.conf file, 145
  - System Administrator rights profile
    - assigning to role, 18
    - description, 135
  - system properties
    - privileges relating to, 27
  - system security
    - privileges, 25
    - using rights, 16
  - System V IPC privileges, 27
- T**
- third-party applications
    - creating rights profiles for, 63
  - troubleshooting
    - assigning passwords for cron jobs, 132
    - failed use of privilege, 129
    - lack of privilege, 129
    - non-UNIX passwords, 132
    - privilege requirements, 129
    - rights, 124
    - rights assignments, 124
    - root as a role, 111
    - user running privileged commands, 124
    - user running privileged shell, 127
  - truss -t command
    - for privilege debugging, 130
  - trusted users
    - assigning extended privileges to, 64
    - assigning roles to, 53, 57
    - creating, 52, 59
    - profile shell as login shell, 60
- U**
- U option
    - list\_devices command, 143
  - umask value, making more restrictive, 67
  - unlocking user account, 68
  - user procedures
    - assuming a role, 101
    - protecting own files from application access, 92
    - using an assigned role, 101
    - using extended privileges, 92
  - user\_attr database, 138, 139
  - useradd command

- authorizations required for, 143
    - description, 142
    - example of using, 56
  - userattr command
    - description, 142
    - use, 74, 111, 124
  - userdel command
    - authorizations required for, 144
    - description, 142
  - usermod command
    - authorizations required for, 144
    - description, 143
    - R option, 100, 110
    - using to assign role, 52
  - users
    - assigning
      - authenticated rights profiles, 61
      - privileges to, 61
      - rights, 47
      - rights defaults, 141
      - rights profiles, 60
    - authenticating to rights profile, 63, 129
    - authenticating to role, 62, 128
    - basic privilege set, 30
    - creating root user, 109
    - creating with useradd command, 52
    - determining hosts where attributes are valid, 121
    - determining if running a profile shell, 127
    - determining own privileged commands, 118
    - expanding rights, 59
    - file permissions
      - restricting, 67
    - guest restrictions, 75
    - initial inheritable privileges, 30
    - locking account, 68
    - managing third-party accounts, 63
    - protecting their files from access by applications, 92
    - protecting their files from web application access, 92
    - removing basic privileges, 72
    - removing rights, 66
    - requiring use of one-time password, 66
    - restricting control of hardware, 70
    - restricting file permissions, 67
    - troubleshooting running privileged commands, 124
    - umask value, 67
    - unlocking accounts of, 70
    - using rights profile, 63, 129
  - using
    - auths command, 107
    - getent command, 110, 114, 115, 118
    - ipadm set-prop command, 88
    - ppriv command, 119, 119
    - profiles command, 55, 63
    - rights defaults, 113
    - rolemod command, 58
    - roles command, 118
    - sudo command, 44
    - svccfg command, 86, 124
    - svcprop command, 88
    - truss command, 130
    - usermod command, 61
    - your assigned administrative rights, 98
- V**
- viewing
    - contents of rights profiles, 136
    - directly assigned privileges, 61
    - privileges in a shell, 62, 119
    - privileges on a process, 119
    - rights of initial user, 113
    - your rights, 113
  - VSCAN Management rights profile
    - cloning to modify, 106
- W**
- web browsers
    - assigning limited privileges, 93
  - web servers
    - Apache HTTP Server, 90
    - checking protections, 91
    - protecting with extended privileges, 90

wildcard characters  
in authorizations, 137

## **Z**

zone.max-locked-memory resource control, 32