Managing Kerberos and Other Authentication Services in Oracle® Solaris 11.3

Part No: E54787
May 2019

ORACLE

Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3

Part No: E54787

Référence: E54787

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

Tables

Examples

Using This Documentation

- **Overview** – Describes how to administer secure authentication on one or more Oracle Solaris systems. The guide covers Pluggable Authentication Modules (PAM), Kerberos, the Simple Authentication and Security Layer (SASL), two-factor authentication (2FA) with smart cards and one-time passwords (OTP), and Secure RPC for NFS and NIS.
- **Audience** – System, security, and network security administrators.
- **Required knowledge** – Access requirements and network 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.

♦ ♦ ♦ CHAPTER 1

Using Pluggable Authentication Modules

This chapter covers the pluggable authentication module (PAM). PAM provides a framework to plug in checks for users of applications on the Oracle Solaris OS. PAM provides a central framework for managing the use of applications, including authenticating the user, managing password changes, closing and opening the user's session, and tracking account limitations, such as time of day. PAM is extensible to third-party applications, and therefore can provide seamless management of access to services on a system.

This chapter covers the following topics:

What's New in Authentication in Oracle Solaris 11.3

This section highlights information for existing customers about important new features in authentication technologies in this release.

- Oracle Solaris provides two multifactor authentication technologies – smart cards and one-time passwords (OTP).
  - For administering and using smart cards, see Chapter 7, "Using Smart Cards for Multifactor Authentication in Oracle Solaris".
  - For administering OTP and using mobile authenticators, see Chapter 8, "Using One-Time Passwords for Multifactor Authentication in Oracle Solaris".
- Kerberos in the Oracle Solaris 11.3 SRU21 release is based on a recent version of MIT Kerberos. See Chapter 2, "Kerberos on Oracle Solaris".
- PAM provides modules that can restrict console login to specified users. See "How to Restrict Who Can Log In to the Console" on page 22. Secure administration

of immutable zones requires that administrators authenticate to the trusted path through a restricted console. See “How to Restrict Access to the Trusted Path Domain” on page 23.

About PAM

PAM provides a framework for applications to perform various authentication functions. It provides central authentication, session management, password management, and account limitations for users of applications, including system programs. PAM enables these programs, such as `login`, `su`, and `ssh`, to remain unchanged when user management details change. Site applications can use PAM to manage their own account, credential, session, and password requirements. PAM is "plugged in" to these applications.

Introduction to the PAM Framework

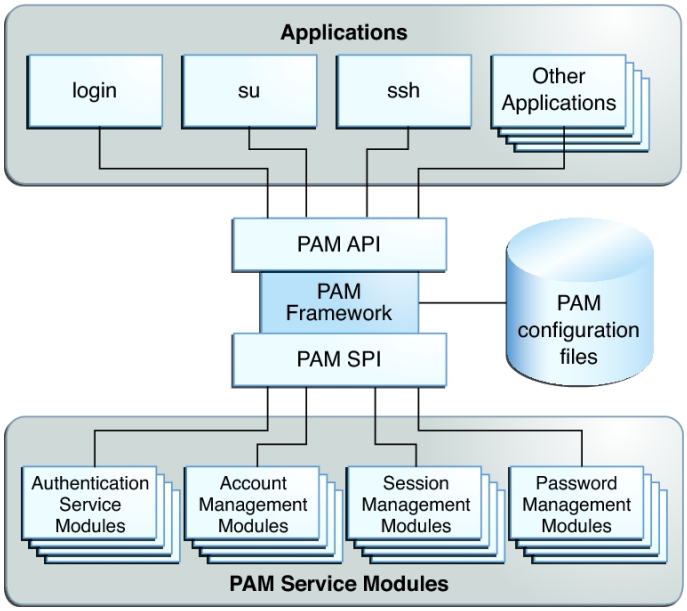
The PAM framework consists of four parts:

- Applications that use PAM
- PAM framework
- PAM service modules
- PAM configuration, including choice of modules and user assignment

The framework provides a uniform way for authentication-related activities to take place. This approach enables application developers to use PAM services without having to know the semantics of the authentication policy. With PAM, administrators can tailor the authentication process to the needs of a particular system without having to change any applications. Rather, administrators adjust the PAM configuration.

The following figure illustrates the PAM architecture.

FIGURE 1 PAM Architecture



The architecture works as follows:

- Applications communicate with the PAM framework through the PAM application programming interface (API).

  For information about using the API, see the `pam`(3PAM) man page and Chapter 3, "Writing PAM Applications and Services" in *Developer's Guide to Oracle Solaris 11.3 Security*.
- PAM service modules communicate with the PAM framework through the PAM service provider interface (SPI). For more information, see the `pam_sm`(3PAM) man page.

  For a brief description of selected service modules, see "PAM Service Modules" on page 41 and the `pam.conf`(4) and `pam_user_policy`(5) man pages.

Administrators can configure one or more series of modules to manage site requirements. This series of modules is called a PAM *stack*. The stack is evaluated in order. If an application requires more than one PAM stack, the application developer must create more than one *service name*. For example, the `sshd` daemon provides and requires several service names for PAM. For the list of PAM service names for the `sshd` daemon, search for the word `PAM` in the `sshd`(1M) man page. For details of the PAM stack, see "PAM Stacking" on page 36. "PAM Stacking Example" on page 40 steps through a PAM authentication stack.

Benefits of Using PAM

The PAM framework enables you to configure the requirements that users must satisfy to use an application. The following are some of the benefits that PAM provides:

- Flexible PAM configuration policy
  - Per-service name authentication policy
  - Site-wide PAM policy and per-user PAM policy
  - Administrative choice of a default authentication policy
  - Enforcement of multiple user requirements on high-security systems
- Ease of use for the end user
  - No retyping of identical passwords for different authentication services
  - User prompting by multiple authentication services rather than requiring a user to type multiple commands
- Ease of configuration for the administrator
  - The ability to pass options to the PAM service module
  - The ability to implement a site-specific security policy without having to change the applications

Planning a Site-Specific PAM Configuration

As delivered, the PAM configuration implements a standard security policy that covers system services that require authentication, such as `login` and `ssh`. If you need to implement a different security policy for some system services or create a policy for third-party applications, consider the following issues:

- Determine that the provided configuration files do not satisfy your requirements.

  Test the default configuration. Test the per-user files in the `/etc/security/pam_policy` directory. Test whether the default service name `other` handles your requirements. "PAM Stacking Example" on page 40 steps you through the `other` stack.
- Identify any service names whose stack needs modification. For an example of modifying a service name's PAM stack, see "How to Create a Site-Specific PAM Configuration File" on page 24.
- For any third-party application that is coded to use the PAM framework, determine the PAM service names that the application uses.
- For each service name, determine which PAM modules to use.

  Review the section 5 man pages for the PAM modules. These man pages describe how each module functions, what options are available, and the interactions between stacked modules.

For a brief summary of selected modules, see "PAM Service Modules" on page 41. PAM modules are also available from outside sources.

- Per service name, decide the order in which to run the modules.
- Select the control flag for each module. For more information about control flags, see "PAM Stacking" on page 36. Note that the control flags can have security implications.

  For a visual representation, see Figure 2, "PAM Stacking: Effect of Control Flags," on page 39 and Figure 3, "PAM Stacking: How Integrated Value Is Determined," on page 40.
- Choose the options that are necessary for each module. The man page for each module lists the options that are available for that module.
- Test the use of the application with the PAM configuration. Test as the `root` role, other roles, privileged users, and regular users. If some users are not permitted to use the application, test those users.

Assigning a Per-User PAM Policy

The `pam_user_policy` PAM module enables system administrators to assign specific PAM configurations on a per-user basis. When this module is the first module in a PAM stack, and the `pam_policy` security attribute for the user specifies a PAM configuration file, that file specifies the PAM policy for the user.

For more information, see the following:

- `pam_user_policy(5)` man page
- "PAM Stacking" on page 36
- "How to Create a Site-Specific PAM Configuration File" on page 24
- Example 4, "Setting Per-User PAM Policy by Using a Rights Profile," on page 29

Configuring PAM

You can use PAM as is. This section gives examples of PAM configurations that are not in effect by default.

TABLE 1 PAM Task Map

Task	Description	For Instructions
Plan for your PAM installation.	Covers how to plan customizing PAM for your site.	"Planning a Site-Specific PAM Configuration" on page 20

Task	Description	For Instructions
Ensure that console login is restricted.	Limits console logins to specified users and netgroups.	"How to Restrict Who Can Log In to the Console" on page 22
Limit administrative access to an immutable zone.	Limits who can log in from the console to immutable zones.	"How to Restrict Access to the Trusted Path Domain" on page 23
Assign a new PAM policy to a user.	Customizes per-user authentication requirements for multiple services.	"How to Create a Site-Specific PAM Configuration File" on page 24
Create users with encrypted home directories.	Modifies a PAM stack to enable the creation of encrypted home directories.	Example 1, "Using a Modified PAM Stack to Create an Encrypted Home Directory," on page 25
Add new PAM modules.	Explains how to install and test customized PAM modules.	"How to Add a PAM Module" on page 27
Assign a non-default PAM policy to users.	Shows how to add a PAM policy to a rights profile for assignment to a range of users at sites that use Kerberos, LDAP, or a combination of logins.	"How to Assign a Modified PAM Policy" on page 29
Assign a non-default PAM policy to users.	Distributes customized PAM stacks to all systems.	"How to Assign a Modified PAM Policy" on page 29
Initiate error logging.	Logs PAM error messages through `syslog`.	"How to Log PAM Error Reports" on page 32
Troubleshoot PAM errors.	Provides steps to locate, solve, and test PAM misconfigurations.	"How to Troubleshoot PAM Configuration Errors" on page 33

▼ How to Restrict Who Can Log In to the Console

In this task, you limit access to the console to particular users. The `/etc/pam.d/login` configuration file controls console login.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. Modify the `/etc/pam.d/login` file.

a. Save a copy of the `login` file, then open the original file.

```
# cd /etc/pam.d
# cp login login.orig
# pfedit login
```

b. Add the following entries:

```
## Account management for login(1) incorporates pam_list(5)
## Restricts who can log in on the console to the users and netgroups
## that are listed in the /etc/users.allow file
```

```
account requisite pam_roles.so.1
account definitive pam_user_policy.so.1
account required pam_unix_account.so.1
account required pam_list.so.1 allow=/etc/users.allow
account required pam_tsol_account.so.1
```

2. **Create and protect the `/etc/users.allow` file.**

```
# cd /etc
# touch users.allow ; chmod 644 users.allow
```

3. **Add users to the `/etc/users.allow` file.**

   - **For example, add the `jdoe` account.**

     ```
     ## permitted console logins
     jdoe
     ```

   - **For example, add netgroups.**
     Netgroups are groups that are centrally defined in LDAP or NIS and have user members. Members of a listed netgroup will be able to log in to this particular system on the console.

     ```
     ## permitted console logins
     jdoe
     @alladmins
     ```

     For more information, see the `netgroup`(4) and `pam_list`(5) man pages.

How to Restrict Access to the Trusted Path Domain

The Trusted Path Domain (TPD) provides secure access to immutable zones and can be used to administer them. In this task, you restrict console access to the TPD to administrators and netgroups that you specify.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Modify the `/etc/pam.d/tpdlogin` file.**
   Uncomment the `pam_list` entry:

```
# cd /etc/pam.d
# pfedit tpdlogin
```

```
## To restrict which users and netgroups are allowed to log in to the
## trusted path, uncomment the line below and add those users and
## netgroups to the /etc/security/tpdusers configuration file.
##
account required       pam_list.so.1 allow=/etc/security/tpdusers
```

2. **Create and protect the `/etc/security/tpdusers` file.**

```
# cd /etc/security
# touch tpdusers ; chmod 644 tpdusers
```

3. **Add the login ID of immutable zone administrators to the `/etc/security/tpdusers` file.**

   - **For example, add the `jdoe` account.**

```
# pfedit tpdusers
## permitted console logins
jdoe
```

   - **For example, add netgroups.**

     Netgroups are groups that are centrally defined, such as in LDAP, and have user members. Members of a listed netgroup will be able to log in to this particular system on the console.

```
## permitted console logins
jdoe
@zoneadmins
```

     For more information, see the `netgroup(4)` and `pam_list(5)` man pages.

     The named administrators can now use the **STOP-A** non-maskable interrupt (NMI) from a Sun keyboard on a SPARC system, or the **F1-A** NMI for x86 access to the console of an immutable zone.

▼ How to Create a Site-Specific PAM Configuration File

In the default configuration, the `ssh` and `telnet` entry services are covered by the `other` service name. The PAM configuration file in this procedure changes the requirements for `ssh` and `telnet`.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Create a new PAM policy configuration file.**

   Use the `pfedit` command to create the file. Place the file in a site configuration directory such as `/opt`. You can also place it in the `/etc/security/pam_policy` directory.

   Note - Do not modify existing files in the `/etc/security/pam_policy` directory.

   Include explanatory comments in the file.

```
# pfedit /opt/local_pam/ssh-telnet-conf
#
# PAM configuration which uses UNIX authentication for console logins,
# (see pam.d/login), and LDAP for SSH keyboard-interactive logins
# This stack explicitly denies telnet logins.
#
sshd-kbdint  auth requisite          pam_authtok_get.so.1
sshd-kbdint  auth binding            pam_unix_auth.so.1 server_policy
sshd-kbdint  auth required           pam_unix_cred.so.1
sshd-kbdint  auth required           pam_ldap.so.1
#
telnet auth     requisite    pam_deny.so.1
telnet account  requisite    pam_deny.so.1
telnet session  requisite    pam_deny.so.1
telnet password requisite    pam_deny.so.1
```

2. **Protect the file.**

   Protect the file with `root` ownership and `444` permissions.

```
# ls -l /opt/local_pam

total 5
-r--r--r--   1 root         4570 Jun 21 12:08 ssh-telnet-conf
```

3. **Assign the policy.**

   See "How to Assign a Modified PAM Policy" on page 29.

Example 1 Using a Modified PAM Stack to Create an Encrypted Home Directory

By default, the `zfs_pam_key` module is not in the `/etc/security/pam_policy/unix` file. In this example, the administrator creates a version of the `unix` PAM per-user policy, then uses the new version to create users whose home directories are encrypted.

```
# cp /etc/security/pam_policy/unix /opt/local_pam/unix-encrypt
# pfedit /opt/local_pam/unix-encrypt.conf
...
```

```
other   auth required           pam_unix_auth.so.1
other   auth required           pam_unix_cred.so.1
## pam_zfs_key auto-creates an encrypted home directory
##
other auth required           pam_zfs_key.so.1 create
```

The administrator uses this policy file when adding users. Note that encryption cannot be added to a filesystem. The filesystem must be created with encryption turned on. For more information, see the zfs_encrypt(1M).

The administrator creates a user and assigns a password.

```
# useradd -K pam_policy=/opt/local_pam/unix-encrypt.conf jill
# passwd jill
New Password: xxxxxxxx
Re-enter new Password: xxxxxxxx
passwd: password successfully changed for jill
```

Then, the administrator creates the encrypted home directory by logging in as the user.

```
# su - jill
Password: xxxxxxxx
Creating home directory with encryption=on.
Your login password will be used as the wrapping key.
Oracle Corporation      SunOS 5.11      11.3    October 2014

# logout
```

For the options to the ZFS service module, see the pam_zfs_key(5) man page.

Finally, the administrator verifies that the new home directory is an encrypted filesystem.

```
# mount -p | grep ~jill
rpool/export/home/jill - /export/home/jill zfs - no
rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
# zfs get encryption,keysource rpool/export/home/jill
NAME                   PROPERTY    VALUE              SOURCE
rpool/export/home/jill  encryption  on                 local
rpool/export/home/jill  keysource   passphrase,prompt  local
```

Example 2 Preventing Users From Seeing Error Messages at Login

In this example, the administrator prevents the display of a login error message by using the nowarn option.

```
# pfedit /opt/local_pam/unix_ldap.conf
...
##
```

```
## turn off login error message`
sshd-kbdint  auth required          pam_unix_cred.so.1    nowarn
sshd-kbdint  auth required          pam_ldap.so.1    nowarn
```

How to Add a PAM Module

This procedure shows how to add, protect, and test a new PAM module. New modules might be required for site-specific security policies or to support third-party applications. To create a PAM module, see Chapter 3, "Writing PAM Applications and Services" in *Developer's Guide to Oracle Solaris 11.3 Security*.

Note - You must install a 32-bit version and a 64-bit version of the PAM service module.

Before You Begin Complete "Planning a Site-Specific PAM Configuration" on page 20.

You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Install then protect both versions of the PAM service module on disk.**

   Ensure that the ownership and permissions protect the module files with `root` ownership and `444` permissions.

```
# cd /opt/pam_modules
# ls -lR
.:
total 4
-r--r--r--   1 root     root        4570 Nov 27 12:34 pam_app1.so.1
drwxrwxrwx   2 root     root           3 Nov 27 12:38 sparcv9

./64:
total 1
-r--r--r--   1 root     root        4862 Nov 27 12:38 pam_app1.so.1
```

   The 32-bit module is in the `/opt/pam_modules` directory and the 64-bit module is in the `64` subdirectory.

2. **Add the module to the appropriate PAM configuration file.**

   In the following example, the module is for a new application, `app1`. Its service name is the same as the application name. Create an `app1` service-name file in the `/etc/pam.d` directory. The first entry in the file enables the `app1` service to be assigned to individual users.

```
# cd /etc/pam.d
```

```
# pfedit app1
...
# PAM configuration
#
# app1 service
#
auth definitive         pam_user_policy.so.1
auth required           /opt/pam_modules/$ISA/pam.app1.so.1  debug
```

The `$ISA` token in the module path directs the PAM framework to the appropriate 32-bit or 64-bit architecture version of the service module for the calling application. For 32-bit applications, */a/b/*$ISA/*module*.so becomes */a/b/module*.so. and for 64-bit applications it becomes */a/b/*64/*module*.so. In this example, you installed the 32-bit `pam.app1.so.1` service module in the `/opt/pam_modules` directory and the 64-bit module in the `/opt/pam_modules/64` directory.

For more information, see the `pfedit`(1M) and `pam.conf`(4) man pages.

To limit the `app1` PAM policy to selected users, see Example 3, “Adding a New Module to a Per-User PAM Policy File,” on page 28.

3. **Test your new service.**

   Log in directly by using `login` or `ssh`. Then, run the commands that are affected by the new module. Test users who are allowed and who are denied use of the affected commands. For troubleshooting assistance, see “How to Troubleshoot PAM Configuration Errors” on page 33.

4. **Assign the policy.**

   See “How to Assign a Modified PAM Policy” on page 29.

Example 3 Adding a New Module to a Per-User PAM Policy File

In this example, the `app1` service is not used by all users, so the administrator adds the service as a per-user policy.

```
# cd /etc/pam.d
# cp app1 /opt/local_pam/app1-conf
# pfedit /opt/local_pam/app1-conf

## app1 service
##
app1 auth definitive         pam_user_policy.so.1
app1 auth required           /opt/pam_modules/$ISA/pam_app1.so.1  debug
```

The administrator deletes the `app1` file from the `pam.d` directory.

```
# rm /etc/pam.d/app1
```

Then, the administrator adds the `app1-conf` policy to the system administrator's PAM policy.

```
# rolemod -K pam_policy=/opt/local_pam/app1-conf sysadmin
```

Example 4 Setting Per-User PAM Policy by Using a Rights Profile

This example uses the `pam_policy` security attribute to enable users from different naming services to be authenticated. The `any` PAM policy file is provided in the `/etc/security/pam_policy` directory. The comments in the file describe this policy.

Do not modify files in this directory.

```
# profiles -p "PAM Per-User Policy of Any" \
'set desc="Profile which sets pam_policy=any";
set pam_policy=any; exit;'
```

To assign this rights profile, see "How to Assign a Modified PAM Policy" on page 29.

▼ How to Assign a Modified PAM Policy

In this procedure, you configure a non-default PAM policy on all system images. After all files are copied, you can assign the new or modified PAM policy to individual users or to all users.

Before You Begin You have modified and tested the PAM configuration files that implement the new policy.

You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Add the non-default PAM files to all systems.**
   You must add all new PAM modules and new and modified PAM configuration files to all systems.

   a. **First, add any new PAM modules to every system.**

      i. **Add the 32-bit PAM module to the architecture-appropriate directory.**

      ii. **Add the 64-bit PAM module to the architecture-appropriate directory.**

      For an example of directory setup, see Step 1 in "How to Add a PAM Module" on page 27.

   b. **Next, add any new PAM configuration files to every system.**

For example, add the `/opt/local_pam/ssh-telnet-conf` file to every system.

c. **Then, copy any modified PAM configuration files to every system.**

For example, copy a modified `/etc/pam.conf` file and any modified `/etc/pam.d/`*service-name-files* to every system.

2. **Assign a non-default PAM policy to all users.**

a. **Modify the `policy.conf` file in one of the following ways:**

- **Add a PAM configuration file to the `PAM_POLICY` keyword in the `policy.conf` file.**

```
# pfedit /etc/security/policy.conf
...
# PAM_POLICY=
PAM_POLICY=/opt/local_pam/ssh-telnet-conf
...
```

- **Add a rights profile to the `PROFS_GRANTED` keyword in the `policy.conf` file.**

For example, assign the PAM Per-User Policy of Any rights profile from Example 4, “Setting Per-User PAM Policy by Using a Rights Profile,” on page 29.

```
# pfedit /etc/security/policy.conf
...
AUTHS_GRANTED=
# PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=PAM Per-User Policy of Any,Basic Solaris User
...
```

b. **Copy the modified `policy.conf` file to every system.**

3. **To assign a non-default PAM policy to individual users, you can assign the policy directly to a user or add the policy to a rights profile that is assigned to users.**

- **Assign the PAM policy directly to individual users.**

```
# usermod -K pam_policy="/opt/local_pam/ssh-telnet-conf" jill
```

- **Include the PAM policy in a rights profile and assign the profile to individual users.**

This example uses the `ldap` PAM policy.

```
# profiles -p "PAM Per-User Policy of LDAP" \
'set desc="Profile which sets pam_policy=ldap";
set pam_policy=ldap; exit;'
```

Then assign the rights profile to a user.

```
# usermod -P +"PAM Per-User Policy of LDAP" jill
```

Example 5 Limiting the `ktelnet` PAM Stack to Selected Users

The administrator wants to allow a limited number of users the ability to use `telnet` in a Kerberos realm. So, before the `telnet` service is enabled, the administrator changes the default `ktelnet` configuration file, and places the default `ktelnet` file in the `pam_policy` directory.

First, the administrator configures a per-user `ktelnet` file.

```
# cp /etc/pam.d/ktelnet /etc/security/pam_policy/ktelnet-conf
# pfedit /etc/security/pam_policy/ktelnet-conf
...
# Kerberized telnet service
#
ktelnet  auth required          pam_unix_cred.so.1
ktelnet  auth required          pam_krb5.so.1
```

The administrator protects the file with `444` permissions.

```
# chmod 444 /etc/security/pam_policy/ktelnet-conf
# ls -l /etc/security/pam_policy/ktelnet-conf
-r--r--r--   1 root     root         228 Nov 27 15:04 ktelnet-conf
```

Then, the administrator modifies the `ktelnet` file in the `pam.d` directory.

- The first entry enables per-user assignment.
- The second entry denies the use of `ktelnet` unless you are assigned `pam_policy=ktelnet` by the administrator.

```
# cp /etc/pam.d/ktelnet /etc/pam.d/ktelnet.orig
               # pfedit /etc/pam.d/ktelnet
...
# Denied Kerberized telnet service
#
auth definitive         pam_user_policy.so.1
auth required           pam_deny.so.1
```

The administrator tests the configuration with a privileged user, a regular user, and the root role. When the configuration passes, the administrator enables the `telnet` service and assigns the per-user policy to the Kerberos administrators.

```
# svcadm enable telnet
# rolemod -S ldap -K pam_policy=ktelnet-conf kerbadmin
```

The administrator copies the modified files to all Kerberos servers, and enables `telnet` on those servers.

How to Log PAM Error Reports

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Determine which `system-log` service instance is online.**

```
# svcs system-log
STATE          STIME    FMRI
disabled       13:11:55 svc:/system/system-log:rsyslog
online         13:13:27 svc:/system/system-log:default
```

2. **Configure the `syslog.conf` file for the level of logging that you need.**

   See the `DESCRIPTION` section of the `syslog.conf(4)` man page for information about the logging levels. Most PAM error reporting is done through the `LOG_AUTH` facility.

   For example, create a file for debug output.

```
# touch /var/adm/pam_debuglog
```

   Then, add the `syslog.conf` entry to send debug output to that file.

Note - If the `rsyslog` service instance is online, modify the `rsyslog.conf` file.

```
# pfedit /etc/syslog.conf
...
*.debug        /var/adm/pam_debuglog
...
```

3. **Refresh the configuration information for the `system-log` service.**

```
# svcadm refresh system-log:default
```

Note - Refresh the `system-log:rsyslog` service instance if the `rsyslog` service is online.

▼ How to Troubleshoot PAM Configuration Errors

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **For each PAM entry that you are troubleshooting, add the `debug` option.**

   For example, the following entries in the `/etc/pam.d/cron` file create debug output for the service.

   ```
   account definitive      pam_user_policy.so.1     debug
   account required        pam_unix_account.so.1    debug
   ```

2. **Log PAM errors at the appropriate level and refresh the `syslog` daemon.**

   For details, see "How to Log PAM Error Reports" on page 32.

3. **If the problem is a corrupt PAM configuration, do the following:**

   a. **Run the application from one terminal window and modify the PAM configuration file in another window.**

   b. **Verify that the errors are corrected by testing the changes in the application window.**

4. **If the problem is a corrupt PAM configuration that prevents login, boot into single-user mode, then correct the file, reboot, and test.**

   - **To boot a SPARC system, type the following command at the PROM prompt:**

     ```
     ok > boot -s
     ```

   - **To boot an x86 system, add the `-s` option to the kernel options line in the GRUB menu.**

   For more information, see the `boot`(1M) and `grub`(5) man pages.

5. **Verify that the errors are corrected.**

   Log in directly by using `login` or `ssh`. Test that regular users, privileged users, and roles can use the affected commands.

PAM Configuration Reference

This section provides additional details about PAM, including PAM stacking.

PAM Configuration Files

System applications, such as `login` and `ssh`, that use the PAM framework are configured in the PAM configuration files in the `/etc/pam.d` directory. The `/etc/pam.conf` file can also be used. Changes to these files affect all users on the system.

Additionally, the `/etc/security/pam_policy` directory holds PAM configuration files. These files cover multiple services and are designed for per-user assignment. Files in this directory must not be modified.

- `/etc/pam.d` **directory –** Contains service-specific PAM configuration files, including the wildcard file, `other`. To add a service for an application, add a single *service-name* file that is the service name used by the application. If appropriate, your application can use the PAM stack in the `other` file.

  The service files in the `/etc/pam.d` directory provide the default configuration in most PAM implementations. They are self-assembled by using the IPS mechanism as described in the `pkg(5)` man page. This default simplifies interoperability with other cross-platform PAM applications. For more information, see the `pam.conf(4)` man page.
- `/etc/pam.conf` **file –** The legacy PAM configuration and policy file. This file is delivered empty. The preferred mechanism for configuring PAM is to use the files in the `/etc/pam.d` directory. For more information, see the `pam.conf(4)` man page.
- `/etc/security/pam_policy` **directory –** Contains PAM policy files that contain policies for multiple services. These files can be assigned to an individual, to a group of individuals, or to all users, as needed. Such an assignment overrides the system PAM configuration files in `pam.conf` or the `/etc/pam.d` directory. Do not modify these files. To add a per-user file, see “How to Create a Site-Specific PAM Configuration File” on page 24. For information about per-user files, see the `pam_user_policy(5)` man page.

The security administrator manages all PAM configuration files. An incorrect order of entries, that is, an incorrect PAM stack, can cause unforeseen side effects. For example, a badly configured file might lock out users so that single-user mode becomes necessary for repair. For assistance, see “PAM Stacking” on page 36 and “How to Troubleshoot PAM Configuration Errors” on page 33.

PAM Configuration Search Order

Application calls to the PAM framework search for the configured PAM services in the following order:

1. The service name is looked up in the `/etc/pam.conf` file.
2. Specific services are used in the `/etc/pam.d/`*service-name* file.
3. The service name `other` is checked in the `/etc/pam.conf` file.
4. The `/etc/pam.d/other` file is used.

This order enables an existing `/etc/pam.conf` file to work with the per-service PAM configuration files in the `/etc/pam.d` directory.

Note - If an unknown option is passed in, for example, a misspelled module name, an error is logged in `syslog` and the option is ignored.

PAM Configuration File Syntax

The `pam.conf` file and the PAM per-user files use a syntax that is different from the service-specific files in the `pam.d` directory.

- The entries in the `/etc/pam.conf` file and the `/etc/security/pam_policy` files are in one of two formats:

  service-name module-type control-flag module-path module-options

  service-name module-type `include` *path-to-included-PAM-configuration*

- The entries in the *service-name* files in the `/etc/pam.d` directory omit the service name. The name of the file provides the service name.

  module-type control-flag module-path module-options

  module-type `include` *path-to-included-PAM-configuration*

The PAM configuration file syntax items are as follows:

service-name

The case-insensitive name of the service, for example, `login` or `ssh`. An application can use different service names for the services that the application provides. For example, search for the word `PAM` in the `sshd`(1M) man page for the service names for the different services that the `sshd` daemon provides.

The predefined service name "`other`" is the default service name if no specific service configuration is provided.

module-type

Indicates the type of service, that is, `auth`, `account`, `session`, or `password`.

control-flag

Indicates the role of the module in determining the success or failure value for the service. Valid control flags are described in "PAM Stacking" on page 36.

module-path

The path to the module that implements the module type. If the pathname is not absolute, it is assumed to be relative to the path `/usr/lib/security/$ISA/`. The `$ISA` macro or token directs the PAM framework to look in the module path's architecture-specific directory.

module-options

Options such as `nowarn` and `debug` that can be passed to the service modules. A module's man page describes the options for that module.

path-to-included-PAM-configuration

Specifies the full path to a PAM configuration file or a file name that is relative to the `/usr/lib/security` directory.

PAM Stacking

When an application calls one of the following functions, the PAM framework reads the PAM configuration files to determine which modules implement this application's PAM service name:

- `pam_authenticate`(3PAM)
- `pam_acct_mgmt`(3PAM)
- `pam_setcred`(3PAM)
- `pam_open_session`(3PAM)
- `pam_close_session`(3PAM)
- `pam_chauthtok`(3PAM)

If the configuration files contain only one module, the result of that module determines the outcome of the operation. For example, the default authentication operation for the `passwd` application contains one module, `pam_passwd_auth.so.1`, in the `/etc/pam.d/passwd` file.

```
auth required           pam_passwd_auth.so.1
```

If, on the other hand, multiple modules implement a service, those modules are said to be *stacked*, that is, a PAM stack exists for that service name. For example, consider the entries in a sample `/etc/pam.d/login` service:

```
auth definitive         pam_user_policy.so.1
auth requisite          pam_authtok_get.so.1
auth required           pam_unix_auth.so.1
auth required           pam_dhkeys.so.1
auth required           pam_unix_cred.so.1
auth required           pam_dial_auth.so.1
```

These entries create an `auth` stack for the `login` service name. To determine the outcome of this stack, the result codes of the individual modules require an *integration process*.

In the integration process, the modules are executed in their order in the file. Each success or failure code is integrated in to the overall result according to the module's control flag. The control flag can cause early termination of the stack. For example, the failure of a `requisite` or `definitive` module terminates the stack. If there are no previous failures, the success of a `sufficient`, `definitive`, or `binding` module also terminates the stack. After the stack is processed, the individual results are combined into a single, overall result that is delivered to the application. For a graphic view of the flow, see Figure 2, “PAM Stacking: Effect of Control Flags,” on page 39 and Figure 3, “PAM Stacking: How Integrated Value Is Determined,” on page 40.

The control flag indicates the role that a PAM module plays in determining success or failure. The control flags and their effects are:

- **Binding** – Success in meeting a binding module's requirements returns success immediately to the application if no previous failures have been recorded. If these conditions are met, then no further execution of modules occurs.

  Failure causes a required failure to be recorded and the processing of modules to be continued.
- **Definitive** – Success in meeting a definitive module's requirements returns success immediately to the application if no previous failures have been recorded.

  If a previous failure has been recorded, that failure is immediately returned to the application with no further execution of modules. Failure results in an immediate error return with no further execution of modules.
- **Include** – Adds lines from a separate PAM configuration file to be used at this point in the PAM stack. This flag does not control success or failure behaviors. When a new file is read, the PAM `include` stack is incremented. When the stack check in the new file finishes, the `include` stack value is decremented. When the end of a file is reached and the PAM `include` stack is `0`, then the stack processing ends. The maximum number for the PAM `include` stack is `32`.

- **Optional** – Success in meeting an optional module's requirements is not necessary for using the service.

  Failure causes an optional failure to be recorded.
- **Required** – Success in meeting a required module's requirements is necessary for the stack to succeed. Final success for the stack is returned only if no binding or required modules have reported failures.

  Failure results in an error return after the remaining modules for this service have been executed.
- **Requisite** – Success in meeting a requisite module's requirements is necessary for the stack to succeed. All requisite modules in the stack must return success for the stack to be able to return success to the application.

  Failure results in an immediate error return with no further execution of modules.
- **Sufficient** – If no previous required failures have been recorded, success in a sufficient module returns success immediately with no further execution of modules.

  Failure causes an optional failure to be recorded.

The following two connected diagrams show how a result is determined in the integration process.

- The first diagram shows how success or failure is recorded for each type of control flag. The results are shown in the second diagram.
- The second diagram shows how the integrated value is determined. Optional failure and required failure return failure, and success returns success. The application determines how to handle these return codes.

FIGURE 2 PAM Stacking: Effect of Control Flags

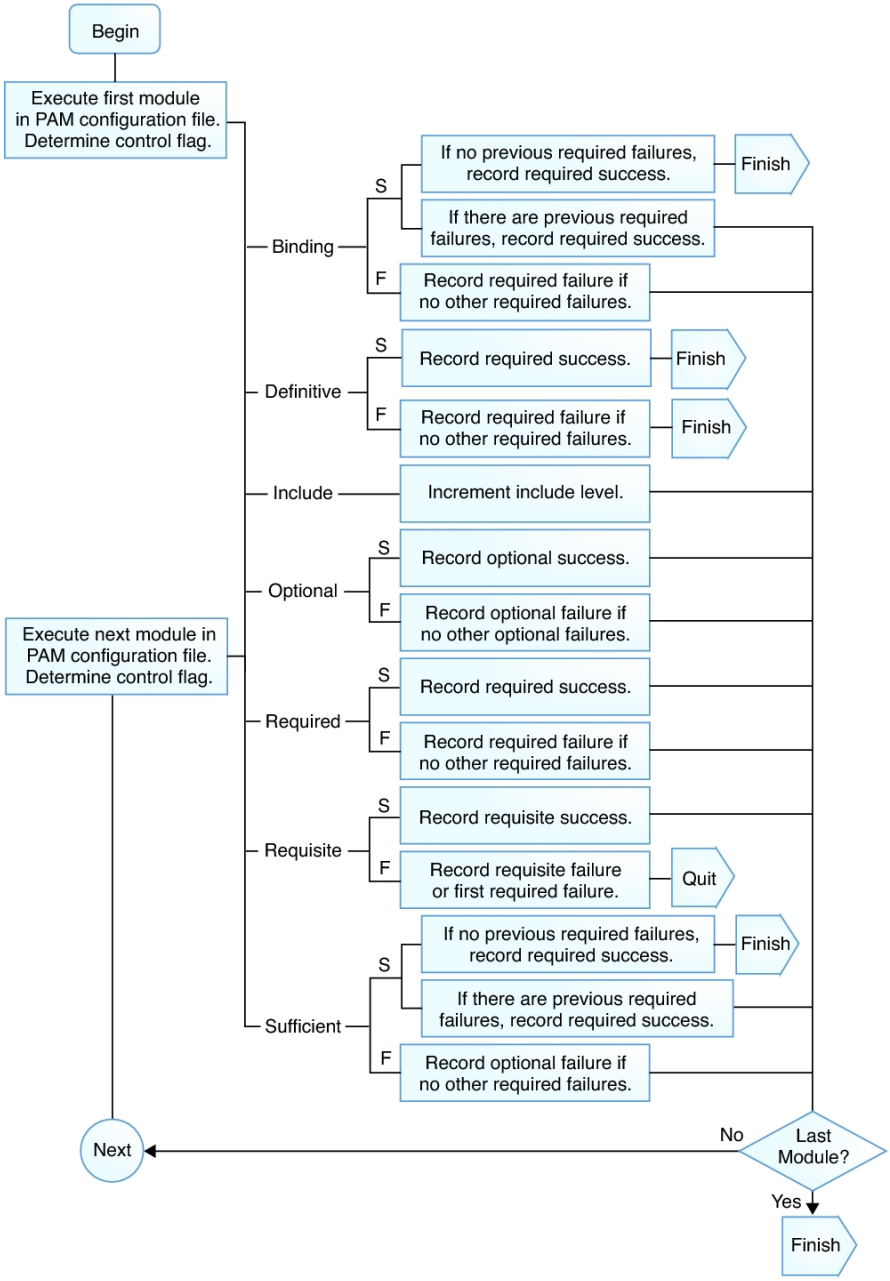
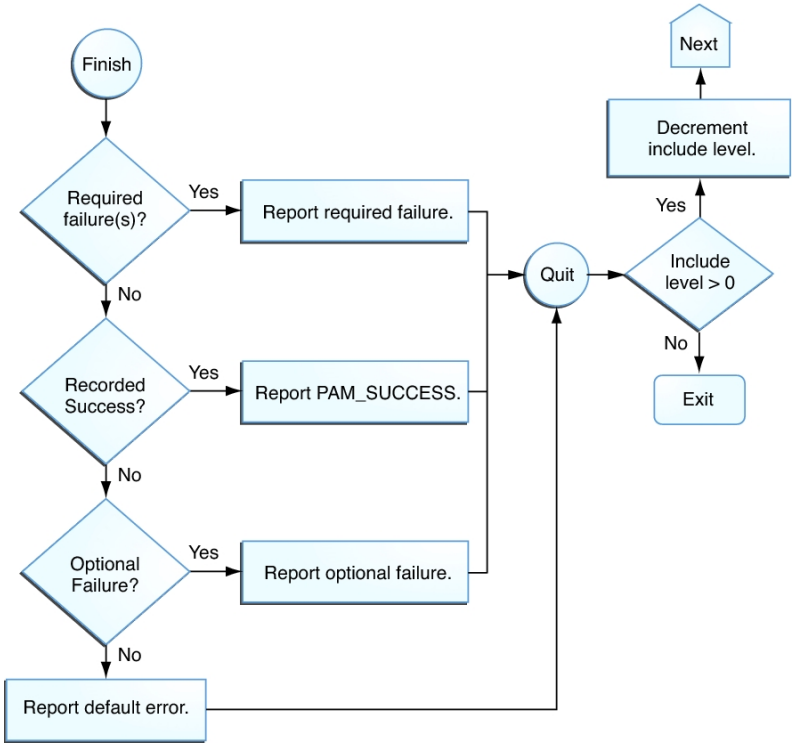


FIGURE 3 PAM Stacking: How Integrated Value Is Determined



PAM Stacking Example

The following example shows the default definitions for authentication management in a sample `/etc/pam.d/other` file. These definitions are used for authentication if no service-specific authentication definitions have been configured.

```
##
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
auth definitive         pam_user_policy.so.1
auth requisite          pam_authtok_get.so.1
```

```
auth required           pam_dhkeys.so.1
auth required           pam_unix_auth.so.1
auth required           pam_unix_cred.so.1
```

First, the PAM policy for the user is checked by using the `pam_user_policy.so` module. The `definitive` control flag dictates that if the evaluation of the configured PAM stack succeeds, success is returned to the application, because no other modules have been checked at this point. If the evaluation of the configured PAM stack fails, then a failure code is returned to the application and no further checking is done. If no per-user PAM policy is assigned to this user, then the next module is executed.

If a per-user PAM policy is not assigned to this user, then the `pam_authtok_get` module is executed. The control flag for this module is set to `requisite`. If `pam_authtok_get` fails, then the authentication process ends and the failure is returned to the application.

If `pam_authtok_get` does not fail, then the next three modules are executed. These modules are configured with the `required` control flag so that the integration process continues regardless of whether an individual failure is returned. After `pam_unix_cred` is executed, no modules remain. At this point, if all the modules succeeded, success is returned to the application. If any of `pam_dhkeys`, `pam_unix_auth`, or `pam_unix_cred` has returned a failure, failure is returned to the application.

PAM Service Modules

This section lists selected PAM service modules. The modules are listed by their man page followed by a brief description of where and when they are used. For more information, read the man page.

For a list of all PAM service modules that Oracle Solaris provides, see section 5 of the man pages. New modules are added on a regular basis. For example, in this release, a number of modules are added for authentication with Windows systems. Your site might also add PAM modules from third parties.

`pam_allow(5)`	Returns `PAM_SUCCESS` for all calls. See also the `pam_deny(5)` man page.
`pam_authtok_check(5)`	Validates the password token for password change.
`pam_authtok_get(5)`	Provides password prompting functionality to the PAM stack.
`pam_authtok_store(5)`	Updates the password token for `PAM_USER`.
`pam_deny(5)`	Returns the module type default failure return code for all calls. See also the `pam_allow(5)` man page.

pam_dhkeys(5)	Provides functionality to two PAM services: Secure RPC authentication and Secure RPC authentication token management.
pam_krb5(5)	Provides functions to verify the identity of a Kerberos user and to manage the Kerberos credentials cache.
pam_krb5_migrate(5)	Helps to migrate PAM_USER to the client's local Kerberos realm.
pam_ldap(5)	Provides functionality for the PAM authentication and account management stacks by the configured LDAP directory server.
pam_list(5)	Provides functions to validate the user's account on this host. The validation is based on a list of users and netgroups on the host.
pam_passwd_auth(5)	Provides authentication functionality to the password stack.
pam_pkcs11(5)	Enables a user to log in to a system by using an X.509 certificate and its dedicated private key that is stored in a PKCS#11 token.
pam_roles(5)	Verifies that a user is authorized to assume a role and prevents direct login by a role.
pam_smb_passwd(5)	Supports the changing or adding of SMB passwords for local Oracle Solaris users. See also the smb(4) man page.
pam_smbfs_login(5)	Synchronizes passwords between Oracle Solaris clients and their CIFS/SMB servers.
pam_tsol_account(5)	Verifies Trusted Extensions account limitations that are related to labels.
pam_tty_tickets(5)	Provides a mechanism for checking a ticket that was created by a prior successful authentication.
pam_unix_account(5)	Provides functions to validate that the user's account is not locked or expired and that the user's password does not need to be changed. Includes checks of access_times and access_tz.
pam_unix_auth(5)	Provides functions to verify that the password is the correct password for PAM_USER.
pam_unix_cred(5)	Provides functions that establish user credential information. It enables the authentication functionality to be replaced independently from the credential functionality.

pam_unix_session(5)	Opens and closes a session, and also updates the /var/adm/lastlog file.
pam_user_policy(5)	Calls a user-specific PAM configuration.
pam_zfs_key(5)	Provides functions to load and change the ZFS encryption passphrase for a user's encrypted home directory.

♦ ♦ ♦ CHAPTER 2

Kerberos on Oracle Solaris

This chapter introduces how Kerberos runs on Oracle Solaris. This chapter contains the following information:

What's New in Kerberos in Oracle Solaris 11.3

Kerberos in the Oracle Solaris 11.3 SRU21 release is based on a recent version of MIT Kerberos. To see which version is installed, run the `klist -V` command.

Introduction to MIT Kerberos on Oracle Solaris

MIT Kerberos on Oracle Solaris takes advantage of Oracle Solaris features, such as the Image Packaging Service (IPS), SMF services, and Automated Installation (AI). See also "Native Oracle Solaris Features Integrated With Kerberos" on page 55.

Comparison of MIT Kerberos and Oracle Solaris Kerberos

The following table describes the differences between MIT Kerberos and the Oracle Solaris version.

TABLE 2 Differences Between MIT Kerberos and Oracle Solaris Kerberos

MIT Kerberos Behavior	Oracle Solaris Kerberos Behavior	Difference in Oracle Solaris
Users download MIT Kerberos from the web.	Administrators install Kerberos as IPS packages.	IPS repositories provide security for data at rest and data in transit.
`k*` commands run Kerberos.	`svc*` commands run Kerberos, which is an SMF service.	Some Kerberos commands are replaced by SMF commands. See "Differences in Defaults Between MIT Kerberos and Oracle Solaris Kerberos" on page 46.
Users create scripts to configure Kerberos clients identically.	Kerberos is integrated with the Automated Install (AI) feature.	Kerberos clients can be installed automatically and identically through AI.
Users create scripts to configure Kerberos clients identically.	Oracle Solaris provides a `kclient` configuration script.	The `kclient` configuration script can configure clients similarly.
Users configure KDCs manually.	Oracle Solaris provides a `kdcmgr` configuration script.	The `kdcmgr` configuration script can configure the KDC with minimal input.
Tickets cannot be automatically renewed.	Oracle Solaris provides the `ktkt_warnd` daemon.	The `ktkt_warnd` daemon can enable automatic ticket renewal.
Relation default values can be different.	Oracle Solaris changes the default for some relations and adds relations.	Oracle Solaris changes the defaults of some Kerberos relations. See "Differences in Defaults Between MIT Kerberos and Oracle Solaris Kerberos" on page 46.

For additional information, see "Documentation About Kerberos" on page 47 and Chapter 4, "Configuring the Kerberos Service".

Differences in Defaults Between MIT Kerberos and Oracle Solaris Kerberos

SMF services for Kerberos and some relations are unique to Oracle Solaris Kerberos. Also, some relations in Oracle Solaris have different default values than the relations in MIT Kerberos.

`kadmin` service

The `svc:/network/security/kadmin:default` SMF service manages the Kerberos database administration daemon in Oracle Solaris. SMF administrative commands include `svcs` for determining the status of the service and `svcadm` for administering the service.

`krb5kdc` service

The `svc:/network/security/krb5kdc:default` SMF service manages the KDC in Oracle Solaris.

`krb5_prop` service

The `svc:/network/security/krb5_prop:default` SMF service manages the Kerberos database propagation daemon in Oracle Solaris.

`-u` permission

In the `kadm5.acl` file, allows or disallows the creation of one-component user principals whose password can be validated with PAM.

`kdc_max_tcp_connections` relation

In the `kdc.conf` file, controls the maximum number of TCP connections that the KDC allows. The minimum value is 10. If this relation is not specified, the Kerberos server allows a maximum of 30 TCP connections.

`admin_server_rotate` and `kdc_rotate` relations

In the `kdc.conf` file, enables log files to be rotated to multiple files on a schedule. The `admin_server_rotate` relation controls the `kadmin` log file and the `kdc_rotate` relation controls the `kdc` log file.

Rotation can be used to avoid logging to a file which might grow too large and halt the KDC. See the `kdc.conf(4)` man page for how to set file versions and the time interval.

`auth_to_local_realm` relation

In the `krb5.conf` file, enables non-default realms to equate with the default realm for authenticated name-to-local name mapping. Unique to Oracle Solaris.

`verify_ap_req_nofail` relation

In the `krb5.conf` file, causes credential verification to fail if the client system does not have a keytab. The default value in Oracle Solaris is `true`.

Documentation About Kerberos

Kerberos documentation for features that Oracle Solaris does not change is on the MIT Kerberos Documentation web site (`http://web.mit.edu/kerberos/krb5-1.14/doc/index.html`). This guide documents Oracle Solaris changes to default Kerberos behavior or Kerberos behaviors that are integrated with Oracle Solaris features.

Kerberos Documentation

Kerberos documentation from MIT covers the following topics:

- What is Kerberos? – Describes the Kerberos environment.

- Administrator Documentation – Includes planning; administering the Key Distribution Center (KDC), also called the database; configuring Kerberos in an LDAP environment; and so on. Includes man pages and troubleshooting. See the Table of Contents.
- User Documentation – Includes ticket and password management, configuration files, and user commands.

Other topics on the MIT Kerberos Documentation web site include developer and build information, plugins, and advanced configuration.

Oracle Solaris Documentation for Kerberos

Supplementary information or information specific to Oracle Solaris is covered in this guide in the following sections:

- "How the Kerberos Service Works" on page 48 – Discusses details about ticket handling by Kerberos.
- "Kerberos and FIPS 140-2 Mode" on page 53 – Describes configuring Kerberos in FIPS 140-2 mode in Oracle Solaris.
- Chapter 3, "Planning for the Kerberos Service" – Describes planning issues that are specify to Oracle Solaris.
- Chapter 4, "Configuring the Kerberos Service" – Describes procedures that use Oracle Solaris features to install and configure Kerberos.
- Chapter 5, "Users Using Kerberos" – Describes Kerberos password, ticketing, and remote login considerations in an Oracle Solaris environment.
- Modified MIT Kerberos man pages – Delivered in the Kerberos IPS packages to describe Oracle Solaris-specific features of Kerberos.

How the Kerberos Service Works

This section provides an overview of the Kerberos authentication system.

From the user's standpoint, the Kerberos service is mostly invisible after the Kerberos session has been started. Commands such as `ssh` or `ftp` work about the same. Initializing a Kerberos session often involves no more than logging in and providing a Kerberos password.

The Kerberos system revolves around the concept of a *ticket*. A ticket is a set of electronic information that identifies a user or a service such as the NFS service. Just as your driver's license identifies you and indicates what driving privileges you have, so a ticket identifies

you and your network access privileges. When you perform a Kerberos-based transaction (for example, if you request an NFS-mounted file), you transparently send a request for a ticket to a *Key Distribution Center*, or KDC. The KDC accesses a database to authenticate your identity and returns a ticket that grants you permission to access the NFS server. "Transparently" means that you do not need to explicitly request a ticket. The request happens when you attempt to access the server. Because only authenticated clients can get a ticket for a specific service, another client cannot access the NFS server under an assumed identity.

Tickets have certain attributes associated with them. For example, a ticket can be *forwardable*, which means that it can be used on another system without a new authentication process. A ticket can also be *postdated*, which means that it is not valid until a specified time. How tickets can be used is set by *policies*, for example, to specify which users are allowed to obtain which types of ticket. Policies are determined when the Kerberos service is installed or administered.

Note - You will frequently see the terms *credential* and *ticket*. While they are often used interchangeably, technically a credential is a ticket plus the *session key* for that session.

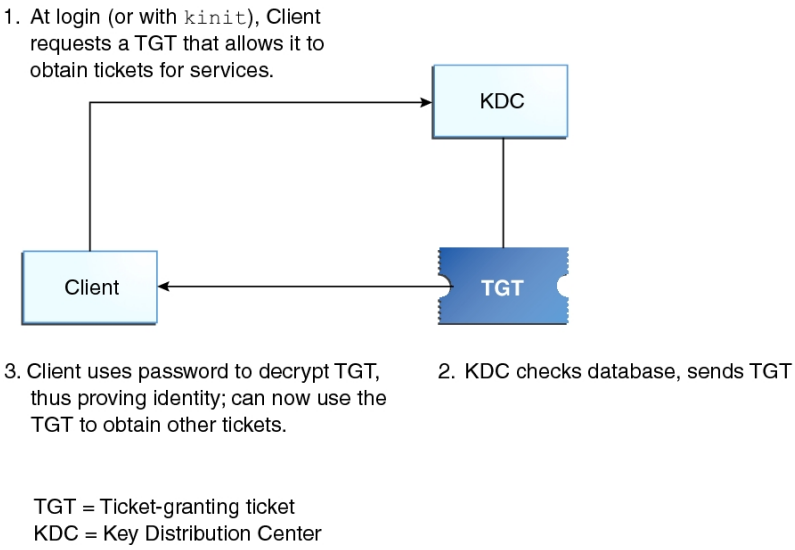
The following sections further explain the Kerberos authentication process.

Initial Authentication: the Ticket-Granting Ticket

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

The following figure shows how the initial authentication takes place.

FIGURE 4 Initial Authentication for a Kerberos Session



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

   A ticket-granting ticket is needed to obtain other tickets for specific services. Think of the ticket-granting ticket as similar to a passport. Like a passport, the ticket-granting ticket identifies you and allows you to obtain numerous "visas" (tickets), which instead of granting access to foreign countries enable you to access remote systems or network services. Like passports and visas, the ticket-granting ticket and the other various tickets have limited lifetimes. The difference is that "Kerberized" commands notice that you have a passport and obtain the visas for you. You don't have to perform the transactions yourself.

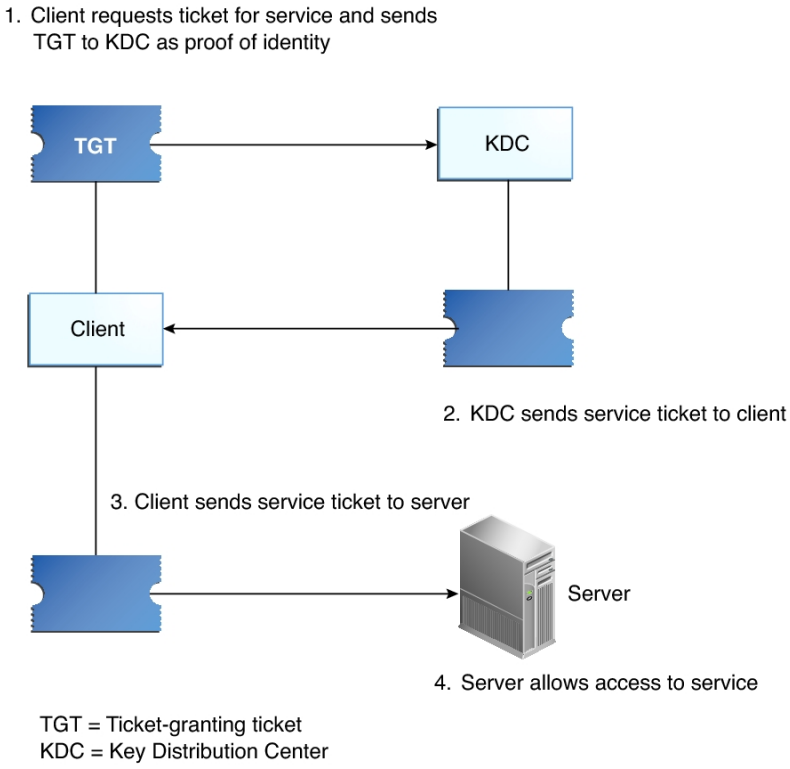
   Another analogy for the ticket-granting ticket is that of a three-day ski pass that is good at four different ski resorts. You show the pass at whichever resort you decide to go to and you receive a lift ticket for that resort as long as the pass has not expired. Once you have the lift ticket, you can ski all you want at that resort. If you go to another resort the next day, you once again show your pass and you get an additional lift ticket for the new resort. The difference is that the Kerberos-based commands notice that you have the weekend ski pass and get the lift ticket for you so you don't have to perform the transactions yourself.
2. The KDC creates a ticket-granting ticket and sends it back, in encrypted form, to the client. The client decrypts the ticket-granting ticket by using the client's password.

3. Now in possession of a valid ticket-granting ticket, the client can request tickets for all sorts of network operations, such as `nfs` or `ssh`, for as long as the ticket-granting ticket lasts. This ticket usually lasts for a few hours. Each time the client performs a unique network operation, it requests a ticket for that operation from the KDC.

Subsequent Kerberos Authentications

After the client has received the initial authentication, each subsequent authentication follows the pattern that is shown in the following figure.

FIGURE 5 Obtaining Access to a Service Using Kerberos Authentication



1. The client requests a ticket for a particular service, for example, to log in remotely to another system, from the KDC by sending the KDC its ticket-granting ticket as proof of identity.
2. The KDC sends the ticket for the specific service to the client.

   Suppose user `jdoe` wants to access an NFS file system that has been shared with `krb5` authentication required. Because `jdoe` is already authenticated (that is, `jdoe` already has a ticket-granting ticket), as `jdoe` attempts to access the files, the NFS client system automatically and transparently obtains a ticket from the KDC for the NFS service. To use a different Kerberized service, `jdoe` obtains another ticket, as in Step 1.
3. The client sends the ticket to the server.

   When using the NFS service, the NFS client automatically and transparently sends the ticket for the NFS service to the NFS server.
4. The server allows the client access.

Although these steps imply that the server never communicates with the KDC, the server does register itself with the KDC, just as the first client does. For simplicity's sake, that section has been omitted.

Kerberos Authentication of Batch Jobs

Batch jobs, such as cron, at, and batch, are delayed execution processes. In a Kerberos environment, all processes including delayed execution processes require credentials. However, users' credentials are relatively short-lived. By default, user credentials are valid for 8 hours and renewable for as long as a week. These times are designed to limit the exposure of sensitive keys to malicious users, but can prevent the execution of jobs at arbitrary times.

In Oracle Solaris, batch jobs that access Kerberos services can run without exposing the user's longterm key. The solution involves storing credentials that include the Kerberos service, the user name, and the client host name in a per-session user credential cache. A PAM module is used to authenticate the batch job. Which services a host can obtain tickets for can be centrally stored in the LDAP directory server.

For more information, see the `pam_krb5_keytab`(5) and `pam_gss_s4u`(5) man pages and “Configuring Delayed Execution for Access to Kerberos Services” on page 99.

Kerberos, DNS, and the Naming Service

The Kerberos service is compiled to use DNS to resolve host names. The `nsswitch` service is not checked at all to resolve host names.

Kerberos and Strong Encryption

Kerberos tickets, described in "How the Kerberos Service Works" on page 48, are encrypted. Kerberos provides several algorithms, or *encryption types*, for encrypting tickets. By default, weak types, such as `des` and `arcfour-hmac` are disallowed. These types should only be allowed for backward compatibility or interoperability. For instructions about limiting encryption to the strongest encryption types, see "How to Require Strong Encryption in Kerberos" on page 64 and the `krb5.conf`(4) man page.

Kerberos and FIPS 140-2 Mode

You can configure Kerberos to run in FIPS 140-2 mode in Oracle Solaris. If your realm contains legacy applications or systems that are not FIPS 140-2-compliant, then the realm cannot run in FIPS 140-2 mode.

When running in FIPS 140-2 mode, Kerberos is said to be a *consumer* of the FIPS 140-2 *provider*. The provider in Oracle Solaris is the OpenSSL FIPS 140-2 provider. For instructions, see "How to Configure Kerberos to Run in FIPS 140-2 Mode" on page 64 and *Using a FIPS 140-2 Enabled System in Oracle Solaris 11.3*.

♦ ♦ ♦ CHAPTER 3

Planning for the Kerberos Service

This chapter covers several installation and configuration options that are specific to Oracle Solaris that you must resolve before you configure or deploy the Kerberos service:

Native Oracle Solaris Features Integrated With Kerberos

Kerberos uses many features that are native to Oracle Solaris, including the Image Packaging Service (IPS), Automated Installation (AI), the Service Management Facility (SMF), and privileges. Oracle Solaris Kerberos may require the use of features that are available but not required on other operating systems, such as PAM. Also, Oracle Solaris can have different defaults than MIT Kerberos. You should plan accordingly.

- Image Packaging Service (IPS) – In Oracle Solaris, MIT Kerberos software is stored in packages in your IPS repository. You install the packages from the repository rather than download the software from the MIT web site.
- Automated Installation (AI) – In Oracle Solaris, AI enables you to install your Kerberos clients identically.
- Pluggable Authentication Modules (PAM) – All authentication on Oracle Solaris systems calls a PAM stack.

  Oracle Solaris provides several PAM stacks that are specific to Kerberos. These stacks are likely different from PAM stacks on other UNIX systems. For more information, read the `/etc/pam.conf` file, then list the modules in the `/etc/pam.d` and `/etc/security/pam_policy` directories and review their corresponding man pages.
- Relation defaults – See "Differences in Defaults Between MIT Kerberos and Oracle Solaris Kerberos" on page 46 for the differences.

Oracle Solaris provides modified Kerberos man pages on your installed system. Use these pages rather than the man pages on the MIT Kerberos Documentation web site.

Planning KDCs

KDCs use specific ports, require additional servers to handle larger ticket loads, and then require propagation techniques to keep the servers synchronized. Additionally, encryption types are centrally managed. You have several options for initially configuring your KDCs.

You can configure a KDC manually as described on the MIT Kerberos Documentation web site, while using Oracle Solaris features such as PAM. Or, you can use the Oracle Solaris `kdcmgr` utility.

The `kdcmgr` utility provides a simple way to configure the KDC automatically or interactively. In the automatic version, you define the configuration parameters as options on the command line. This version is especially useful for scripts. The interactive version prompts you for all information that is needed. For pointers to the instructions for using this command, see "Configuring KDC Servers" on page 62.

You can also use LDAP to manage the database files for Kerberos. For instructions, see "Configuring KDC Servers on LDAP Directory Servers" on page 69. LDAP simplifies administration at sites that require coordination between the Kerberos databases and their existing directory server setup.

Planning for Kerberos Clients

You can use choose one of three ways to install Kerberos clients:

- Automatically – See "Using Automatic Installation to Install Kerberos Clients" on page 57.
- By script – See "Using the `kclient` Profile to Install Kerberos Clients" on page 57.
- Manually – See the MIT Kerberos Documentation web site while taking into account Oracle Solaris features such as PAM.

Client configuration planning includes which PAM module to use and whether to allow delegation of services.

- In Oracle Solaris, the PAM framework provides protected network logins with the `pam_krb5` module and other PAM modules. See "Kerberos Client Login Security" on page 58 and the `pam_krb5(5)` man page.

- When a client requests a service, that service can be granted by a server other than the master KDC. For more information, see "Trusted Delegated Services in Kerberos" on page 58.

For the procedures, see "Configuring Kerberos Clients" on page 80.

Using Automatic Installation to Install Kerberos Clients

Kerberos clients can be configured quickly and easily by using the Oracle Solaris Automated Installer (AI) feature. AI server administrators create and assign Kerberos configuration profiles to AI clients. Additionally, the AI server delivers the client keys. Therefore, at installation, the Kerberos client is a fully provisioned Kerberos system, capable of hosting secure services. Using the Automated Installer can lower system administration and maintenance costs.

You run the `kclient` command to create Kerberos configuration profiles for AI. For more information, see the `kclient`(1M) man page. For instructions to configure Kerberos clients by using AI, see "How to Configure Kerberos Clients Using AI" in *Installing Oracle Solaris 11.3 Systems*.

You can use AI for clients that are not clients of an Oracle Solaris KDC. For the list of KDC vendors, see the `kclient`(1M) man page.

For all KDC types, pre-generated keytab transfer is supported. Oracle Solaris KDC and MS AD also support auto-registering.

Using the `kclient` Profile to Install Kerberos Clients

In addition to AI configuration, Oracle Solaris provides the `kclient` configuration utility. This utility runs in interactive mode and noninteractive mode. Interactive mode prompts you for Kerberos-specific parameter values, so you can make changes when configuring each client. In noninteractive mode, you supply a file with parameter values and you can supply command-line options. The `kclient` utility requires fewer steps than manual configuration and are quicker and less prone to error.

If the following setup is in effect, then no explicit configuration of your Kerberos client is necessary:

- DNS is configured to return SRV records for KDCs.

- The realm name matches the DNS domain name, or the KDC supports referrals.
- The Kerberos client does not require keys that are different from the KDC server's keys.

You still might want to explicitly configure the Kerberos client for the following reasons:

- The zero-configuration process performs more DNS lookups than a directly configured client, and therefore is less efficient than direct configuration.
- If referrals are not used, the zero-configuration logic depends on the DNS domain name of the host to determine the realm. This configuration introduces a small security risk, but the risk is much smaller than enabling `dns_lookup_realm`.
- The `pam_krb5` module relies on a host key entry in the keytab file. Although this requirement can be disabled in the `krb5.conf` file, doing so is not recommended for security reasons. For more information, see "Kerberos Client Login Security" on page 58 and the `krb5.conf(4)` man page.

Kerberos Client Login Security

At login, a client uses a Kerberos PAM module to verify that the KDC that issued the latest TGT is the same KDC that issued the client host principal that is stored in the `/etc/krb5/krb5.keytab` file. The PAM module must be configured in the authentication stack to verify the KDC.

For some configurations, such as DHCP clients that do not store a client host principal, this check needs to be disabled. To disable the check, see "Verifying Kerberos Clients Without a Host Principal" on page 87.

Trusted Delegated Services in Kerberos

For some applications, a client might need to delegate authority to a server to act on its behalf in contacting other services. The client must forward credentials to an intermediate server. The client's ability to obtain a service ticket to a server conveys no information to the client about whether the server can be trusted to accept delegated credentials. The `ok_to_auth_as_delegate` option to the `kadmin` command provides a way for a KDC to communicate the local realm policy to a client regarding whether an intermediate server is trusted to accept such credentials.

The encrypted part of the KDC reply to the client can include a copy of the credential ticket flags with the `ok_to_auth_as_delegate` option set. A client can use this setting to determine whether to delegate credentials (by granting either a proxy or a forwarded TGT) to this server. When setting this option, consider the security and placement of the server on which the service runs, as well as whether the service requires the use of delegated credentials.

Planning Kerberos Use of UNIX Names and Credentials

The Kerberos service provides a default mapping of GSS credential names to UNIX user IDs (UIDs) for GSS applications that require this mapping, such as NFS. GSS credential names are equivalent to Kerberos principal names when using the Kerberos service. Also, UNIX users who do not have valid user accounts in the default Kerberos realm can be automatically migrated by using the PAM framework.

Automatic User Migration to a Kerberos Realm

UNIX users who do not have valid user accounts in the default Kerberos realm can be automatically migrated by using the PAM framework. Specifically, you add the `pam_krb5_migrate.so` module to the authentication stack of the PAM service. Services are then configured so that whenever a user who does not have a Kerberos principal performs a successful password login to a system, a Kerberos principal would be automatically created for that user. The new principal password is then the same as the UNIX password. For instructions about using the `pam_krb5_migrate.so` module, see "How to Configure Automatic Migration of Users in a Kerberos Realm" on page 89.

Synchronizing Clocks Between KDCs and Kerberos Clients

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

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

The default value for the maximum clock skew is 300 seconds (five minutes). You can lower this default in the `libdefaults` section of the `krb5.conf` file.

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

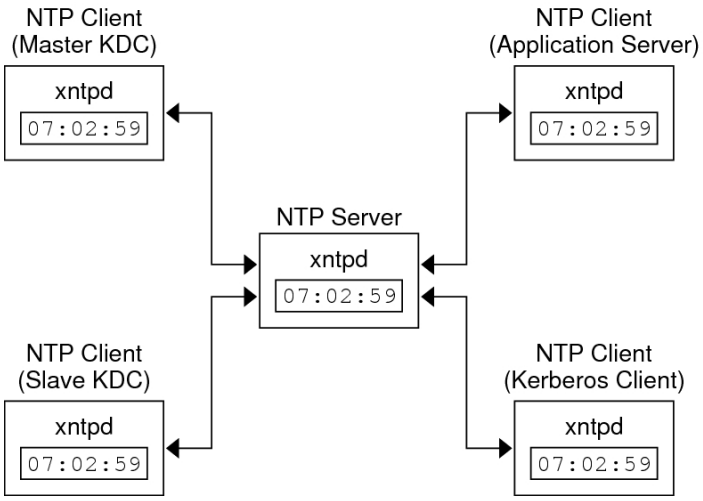
Because maintaining synchronized clocks between the KDCs and Kerberos clients is important, use the Precision Time Protocol (PTP) or Network Time Protocol (NTP) software to synchronize the clocks. For how to configure clock synchronization in Oracle Solaris, see

Enhancing System Performance Using Clock Synchronization and Web Caching in Oracle Solaris 11.3.

The NTP software is installed by default on most Oracle Solaris systems. You can install the PTP software by using the `pkg install ptp` command.

The following figure shows an example of NTP clock synchronization.

FIGURE 6 Synchronizing Clocks by Using NTP



Ensuring that the KDCs and Kerberos clients maintain synchronized clocks involves implementing the following steps:

1. Setting up a PTP or an NTP server on your Kerberos network. This server can be any system except the master KDC.
2. As you configure the KDCs and Kerberos clients on the network, make them clients of the clock synchronization server. Return to the master KDC to configure the KDC as a client of the clock synchronization server.
3. Enabling the clock synchronization service on all systems.

   For the procedures, see *Enhancing System Performance Using Clock Synchronization and Web Caching in Oracle Solaris 11.3*.

♦ ♦ ♦ CHAPTER 4

Configuring the Kerberos Service

This chapter provides configuration procedures for KDC servers, network application servers, NFS servers, and Kerberos clients. Many of these procedures require `root` access, so they should be performed by system administrators or advanced users. Cross-realm configuration procedures and other topics related to KDC servers are also covered.

This chapter covers the following topics:

Configuring the Kerberos Service

Because some procedures in the configuration process depend on other procedures, they must be done in a specific order. These procedures often establish services that are required to use the Kerberos service. Other procedures are not ordered, and so can be performed when appropriate. The following task map shows a suggested order for a Kerberos installation.

Note - The examples in these sections use default encryption types, which are not FIPS 140-2-validated for Oracle Solaris. To run in FIPS 140-2 mode, you must limit the encryption types to only FIPS 140-2-validated encryption types for the database, servers, and client communications. For more information, see "How to Configure Kerberos to Run in FIPS 140-2 Mode" on page 64.

TABLE 3 Task Map: Configuring the Kerberos Service

Task	Description	For Instructions
1. Plan your Kerberos installation.	Resolves configuration issues before you start the software configuration process. Planning ahead saves you time and other resources later.	Chapter 3, "Planning for the Kerberos Service"
2. Configure the KDC servers.	Configures and builds the master KDC and the slave KDC servers and KDC database for a realm.	"Configuring KDC Servers" on page 62
2a. (Optional) Configure Kerberos to run in FIPS 140-2 mode.	Enables the use of FIPS 140-2-validated algorithms only.	"How to Configure Kerberos to Run in FIPS 140-2 Mode" on page 64
2b. (Optional) Configure Kerberos to run on LDAP.	Configures the KDC to use an LDAP Directory Server.	"Configuring KDC Servers on LDAP Directory Servers" on page 69
3. Install clock synchronization software.	Creates a central clock that provides the time for all hosts on the network.	"Synchronizing Clocks Between KDCs and Kerberos Clients" on page 59
4. (Optional) Increase security on the KDC servers.	Prevents security breaches on the KDC servers.	"Restricting Access to KDC Servers" on page 102

After you have completed the required steps, perform the following procedures when appropriate:

- Configure Kerberos application servers, such as an `ftp` server – "Configuring Kerberos Network Application Servers" on page 92
- Enable a `cron` host to execute tasks at arbitrary times – "Configuring Delayed Execution for Access to Kerberos Services" on page 99
- Enable a server to share a file system that requires Kerberos authentication – "Configuring Kerberos NFS Servers" on page 95
- Enable a client to use Kerberos services – "Configuring Kerberos Clients" on page 80
- Maintain the Kerberos database – "Administering the Kerberos Database" on page 101

Configuring KDC Servers

After you install the Kerberos software, you must connect clients to existing Key Distribution Center (KDC) servers, or configure a master KDC and at least one slave KDC. KDCs issue credentials. These credentials are the basis for the Kerberos service, so the KDCs must be configured before you attempt other tasks.

Note - If you have an existing KDC, go to "Configuring Kerberos Clients" on page 80.

You can choose to configure and build the master KDC server, the database, and additional servers on Oracle Solaris with the `kdcmgr` utility or manually.

Note - For all configuration methods, you must install the KDC package from your repository as described in "How to Install the KDC Package" on page 63.

Perform the following procedures to configure KDC servers:

- Install the KDC package from your IPS repository – "How to Install the KDC Package" on page 63
- Enable the use of FIPS 140-2-validated algorithms only – "How to Configure Kerberos to Run in FIPS 140-2 Mode" on page 64
- Use a script to configure the KDCs –
  - "How to Use `kdcmgr` to Configure the Master KDC" on page 65
  - "How to Use `kdcmgr` to Configure a Slave KDC" on page 67
  - Example 6, "Running the `kdcmgr` Command Without Arguments," on page 67
- If you are configuring your KDC server manually, follow the instructions on the MIT Kerberos Documentation web site (`http://web.mit.edu/kerberos/krb5-1.14/doc/index.html`), while keeping in mind "Native Oracle Solaris Features Integrated With Kerberos" on page 55.

How to Install the KDC Package

By default, Kerberos client software is installed on your system, but the Key Distribution Center (KDC) software is not. To install a KDC, you must add the KDC package.

Before You Begin You must be assigned the Software Installation rights profile to add packages to the system. The initial user has this right as does the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Install the KDC package.**

```
$ pkg install security/kerberos-5/kdc
```

For more information about package installation, see the `pkg(1)` man page.

2. **(Optional) List the Kerberos services.**

With the addition of the server package, your system has three Kerberos services, two for the KDC and one for the Kerberos client. These services are disabled until you configure Kerberos and then explicitly enable the services.

```
$ svcs -a krb5
STATE          STIME    FMRI
```

```
disabled       Sep_10   svc:/security/kerberos-5/krb5kdc:default
disabled       Sep_10   svc:/security/kerberos-5/krb5_prop:default
$ svcs -a | grep kerb
STATE          STIME    FMRI
disabled       Sep_07   svc:/security/kerberos-5/install:default
```

How to Require Strong Encryption in Kerberos

This procedure completely disables the use of the `arcfour-hmac` and `des3-cbc-sha1` encryption types.

Caution - This procedure breaks interoperability for deployments that join Oracle Solaris systems to domains and forests that are using weaker encryption.

1. **On the KDC, require strong encryption types for all tickets.**

   Modify the permitted encryption types in the `[libdefaults]` section of the `krb5.conf` file.

   ```
   kdc # cd /etc/krb5
   kdc # pfedit krb5.conf
        [libdefaults]
        ...
        permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
   ```

2. **On Kerberos clients, require strong encryption types for all tickets.**

   Modify the default encryption types in the `[libdefaults]` section of the `krb5.conf` file.

   ```
   Kerberos-client # cd /etc/krb5
   Kerberos-client # pfedit krb5.conf
       [libdefaults]
        ...
       default_tgs_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
       default_tkt_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
   ```

How to Configure Kerberos to Run in FIPS 140-2 Mode

Before You Begin For Kerberos to run in FIPS 140-2 mode, you must enable FIPS 140-2 mode on your system. See "How to Create a Boot Environment With FIPS 140-2 Enabled" in *Managing Encryption and Certificates in Oracle Solaris 11.3*.

1. **On the master KDC, edit the encryption types for the KDC.**

   In the [realms] section of the kdc.conf file, set the master key type for the KDC database:

   ```
   # pfedit  /etc/krb5/kdc.conf
   ...
   master_key_type = des3-cbc-sha1-kd
   ```

2. **In the same file, explicitly forbid other encryption types.**

   Because you can also set encryption by running a command, the configuration files should prevent the use of a non-FIPS 140-2 algorithm argument to a command.

   ```
   supported_enctypes = des3-cbc-sha1-kd:normal
   ```

3. **Edit the encryption types for transactions in the [libdefaults] section of the krb5.conf file.**

   These parameters limit the encryption types for the Kerberos servers, services, and clients.

   ```
   # pfedit /etc/krb5/krb5.conf
           default_tgs_enctypes = des3-cbc-sha1-kd
           default_tkt_enctypes = des3-cbc-sha1-kd
           permitted_enctypes = des3-cbc-sha1-kd
   ```

4. **In the same file, explicitly forbid weak encryption types.**

   ```
           allow_weak_enctypes = false
   ```

Troubleshooting For the encryption types that Kerberos recognizes, see Kerberos Encryption Types on the MIT Kerberos Documentation web site. For the encryption types that OpenSSL provides in FIPS 140-2 mode, see https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp1747.pdf. An encryption type that is both in Kerberos and in the FIPS 140-2 mode of OpenSSL can be used to run Kerberos in FIPS 140-2 mode.

▼ How to Use kdcmgr to Configure the Master KDC

The kdcmgr script provides a command-line interface to install the master and slave KDCs. For the master, you must create a password for the Kerberos database and a password for the administrator. On the slave KDCs, you must supply these passwords to complete the installation. For information about these passwords, see the kdcmgr(1M) man page.

Before You Begin You must assume the root role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Create the master KDC.**

   On the command line, run the `kdcmgr` command and name the administrator and the realm.

   You are prompted for the Kerberos database password, called the *master key* and the password for the administrative principal. The script prompts for the passwords.

```
kdc1# kdcmgr -a admin-name/admin -r DOMAIN.SUFFIX create master

Starting server setup
--------------------------------------

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@DOMAIN.SUFFIX'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     /** Type strong password **/
Re-enter KDC database master key to verify: xxxxxxxx

Authenticating as principal root/admin@DOMAIN.SUFFIX with password.
WARNING: no policy specified for admin-name/admin@DOMAIN.SUFFIX; defaulting to no
 policy
Enter password for principal "admin-name/admin@DOMAIN.SUFFIX":     /** Type strong
 password **/
Re-enter password for principal "admin-name/admin@DOMAIN.SUFFIX": xxxxxxxx
Principal "admin-name/admin@DOMAIN.SUFFIX" created.

Setting up /etc/krb5/kadm5.acl.

--------------------------------------------------
Setup COMPLETE.

kdc1#
```

   Note - Save and store these passwords in a safe location.

2. **(Optional) Display the status of the master KDC.**

```
# kdcmgr status
```

3. **Synchronize this system's clock with other clocks in the realm.**

   Note - A master KDC cannot be the clock synchronization server.

For more information and pointers to procedures, see “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 59. See also the krb5.conf(4) man page.

Example 6 Running the kdcmgr Command Without Arguments

In this example, the administrator supplies the realm name and admin principal when prompted by the script.

```
kdc1# kdcmgr create master

Starting server setup
---------------------------------------

Enter the Kerberos realm: EXAMPLE.COM

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     /** Type strong password **/
Re-enter KDC database master key to verify: xxxxxxxx

Enter the krb5 administrative principal to be created: kws/admin

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM":     /** Type strong password **/
Re-enter password for principal "kws/admin@EXAMPLE.COM": xxxxxxxx
Principal "kws/admin@EXAMPLE.COM" created.

Setting up /etc/krb5/kadm5.acl.

-------------------------------------------------
Setup COMPLETE.

kdc1#
```

▼ How to Use kdcmgr to Configure a Slave KDC

Before You Begin The master KDC server is configured.

You must assume the root role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Create a slave KDC.**

   On the command line, run the kdcmgr command and name the administrator, the realm, and the master KDC.

   The script prompts for the two passwords that you created when you created the master KDC, one for the administrative principal and one for the KDC database. For the EXAMPLE.COM example, you created the passwords in Example 6, "Running the kdcmgr Command Without Arguments," on page 67.

```
kdc2# kdcmgr -a kws/admin -r EXAMPLE.COM create -m kdc1 slave

Starting server setup
--------------------------------------

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf
Obtaining TGT for kws/admin ...
Password for kws/admin@EXAMPLE.COM: xxxxxxxx

Setting up /etc/krb5/kadm5.acl.

Setting up /etc/krb5/kpropd.acl.

Waiting for database from master...
Waiting for database from master...
Waiting for database from master...
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key
Enter KDC database master key: xxxxxxxx

--------------------------------------------------
Setup COMPLETE.

kdc2#
```

2. **(Optional) Display the status of the KDC.**

```
# kdcmgr status
```

3. **Synchronize this system's clock with other clocks in the realm.**

   For more information and pointers to procedures, see "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 59. See also the krb5.conf(4) man page.

4. **Return to the master KDC to make it a client of the clock synchronization server.**

Configuring KDC Servers on LDAP Directory Servers

In order to configure and build a master KDC server and secondary servers on LDAP, you have to create the LDAP back end and the Kerberos KDC, and then configure Kerberos and LDAP to recognize each other. This section shows how to configure an OpenLDAP back end and an Oracle Unified Directory (OUD) back end and connect them to the KDC. It also describes how to add Kerberos attributes to the LDAP people object class and how to destroy a Kerberos realm on an LDAP directory server.

The following tasks are in this section:

- "Configuring a Master KDC on an OpenLDAP Directory Server" on page 69
- "Configuring a Master KDC on an Oracle Unified Directory Server" on page 73
- "How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type on an OpenLDAP Server" on page 78
- "How to Destroy a Kerberos Realm on an LDAP Directory Server" on page 79

Configuring a Master KDC on an OpenLDAP Directory Server

By installing the KDC and OpenLDAP on the same server you get better performance.

The main steps involved in configuring the KDC and OpenLDAP on the same server are:

1. Installing the OpenLDAP package
2. Enabling the LDAP service
3. Configuring access to the OpenLDAP server
4. Ensuring that the OpenLDAP daemon is listening on `ldapi://`
5. Adding organizational entries to the OpenLDAP server
6. Adding the OpenLDAP server to the KDC configuration file
7. Creating LDAP entries in the Kerberos database
8. Adding the KDC and `kadmin` roles to the OpenLDAP server
9. Creating the Kerberos database keys
10. Synchronizing the master KDC's clock with the clock synchronization server

11. Enabling the KDC and `kadmin` services

▼ How to Configure a Master KDC on an OpenLDAP Directory Server

This procedure configures a KDC master and an OpenLDAP server on the same system. The KDC uses the OpenLDAP client library, as will the Kerberos clients that you configure later.

Before You Begin Make sure the system is configured to use DNS. For more information about OpenLDAP, see the OpenLDAP Home Page.

You are in the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. Install the `openldap` server package.

```
# pkg install service/network/ldap/openldap
```

2. Enable the OpenLDAP service.

This step enables the directory server to read the configuration file and be populated.

```
# svcadm enable ldap/server
```

3. Configure access to the OpenLDAP server.

Modify access information for the OpenLDAP configuration by creating and loading the `access.ldif` file.

```
# cat <<- EOF >access.ldif
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to dn.subtree="cn=example.com,cn=krbcontainer,dc=example,dc=com"
   by dn.base="cn=kdc service,ou=profile,dc=example,dc=com" write
   by dn.base="cn=kadmin service,ou=profile,dc=example,dc=com" write
   by * none
-
add: olcAccess
olcAccess: {1}to dn.subtree="ou=users,dc=example,dc=com"
   by dn.base="cn=kdc service,ou=profile,dc=example,dc=com" write
   by dn.base="cn=kadmin service,ou=profile,dc=example,dc=com" write
   by * none
EOF
# ldapmodify -D "cn=config" -W -f access.ldif
```

Note - In the "Providing access to" sections, `kdc service` needs `write` access to any accounts that account lockout should apply to. Write access enables the service to lock out an account after its account password has expired.

4. **Ensure that the `slapd` daemon is listening on the `ldapi://` UNIX domain socket.**

```
# ldapsearch -H ldapi:/// -x -b "" -s base '(objectclass=*)' namingContexts
```

5. **Add organizational entries to the OpenLDAP server.**

```
# cat <<- EOF >entries.ldif
dn: ou=groups,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: groups

dn: ou=users,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: users
EOF
# ldapadd -D "cn=Manager,dc=example,dc=com" -W -f entries.ldif
```

6. **Add the OpenLDAP server to the Kerberos configuration file.**

```
# pfedit /etc/krb5/krb5.conf
[realms]
        EXAMPLE.COM = {
                kdc = krb1.example.com
                admin_server = krb1.example.com
                database_module = LDAP
        }

[dbmodules]
        LDAP = {
                db_library = kldap
                ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
                ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
                ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
                ldap_servers = ldapi:///
        }
...
```

7. **Create the LDAP entries in the Kerberos database.**

```
# kdb5_ldap_util -D "cn=Manager,dc=example,dc=com" create \
```

```
   -subtrees ou=users,dc=example,dc=com -r EXAMPLE.COM -s
```

For more information, see the kdb5_ldap_util(1M) man page.

8. **Create and add KDC and kadmin roles.**

```
# cat <<- EOF >kdc_roles.ldif
dn: cn=kdc service,ou=profile,dc=example,dc=com
cn: kdc service
sn: kdc service
objectclass: top
objectclass: person
userpassword: nnnnnnnn

dn: cn=kadmin service,ou=profile,dc=example,dc=com
cn: kadmin service
sn: kadmin service
objectclass: top
objectclass: person
userpassword: nnnnnnnn
EOF
# ldapadd -D "cn=Manager,dc=example,dc=com" -W -f kdc_roles.ldif
```

The passwords for the `kdc` service and the `kadmin` service should be different and difficult to guess. Remember these passwords. You use them in the following step.

9. **Create stash files for LDAP binding to the KDC and `kadmin` services.**

```
# kdb5_ldap_util -D "cn=Manager,dc=example,dc=com" stashsrvpw \
   cn="kdc service,ou=profile,dc=example,dc=com"
Password for "cn=Manager,dc=example,dc=com": nnnnnnnn
Password for "cn=kdc service,ou=profile,dc=example,dc=com": nnnnnnnn
Re-enter password for "cn=kdc service,ou=profile,dc=example,dc=com": nnnnnnnn
# kdb5_ldap_util -D "cn=Manager,dc=example,dc=com" stashsrvpw \
   cn="kadmin service,ou=profile,dc=example,dc=com"
Password for "cn=Manager,dc=example,dc=com": nnnnnnnn
Password for "cn=kadmin service,ou=profile,dc=example,dc=com": nnnnnnnn
Re-enter password for "cn=kadmin service,ou=profile,dc=example,dc=com": nnnnnnnn
```

10. **Synchronize this system's clock with other clocks in the realm.**

Note - A master KDC cannot be the clock synchronization server.

For more information and pointers to procedures, see “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 59. See also the krb5.conf(4) man page.

11. **Enable the KDC and `kadmin` services.**

```
# svcadm enable krb5kdc
# svcadm enable kadmin
```

Configuring a Master KDC on an Oracle Unified Directory Server

By installing the KDC and LDAP on the same server you get better performance.

The main steps are:

1. Installing the OUD package
2. Configuring the OUD server
3. Adding the OUD server to the Kerberos configuration file
4. Creating keys for the KDC and specifying a privileged port for the OUD servers
5. Configuring KDC roles and services on the OUD server
6. Creating and installing a certificate and keys for the OUD server
7. Testing
8. Synchronizing the master KDC's clock with the clock synchronization server

▼ How to Configure a Master KDC on an Oracle Unified Directory LDAP Directory Server

This procedure configures a KDC master and an Oracle Unified Directory (OUD) server on the same system. The KDC uses the OpenLDAP client library, as will the Kerberos clients that you configure later.

Before You Begin Make sure the system is configured to use DNS. This procedure uses Oracle Unified Directory (OUD) for LDAP. For more information, see the Oracle Fusion Middleware Installation Guide for Oracle Unified Directory 11g Release 2 (11.1.2) .

You are in the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. Download the OUD package.

Follow the directions on the Oracle Identity Management Downloads web site.

2. Configure the OUD LDAP server.

Follow the Oracle Unified Directory (OUD) links on the Oracle Identity Management Documentation page to the documentation for your OUD.

This sample configuration uses the following parameters:

- Listener port: `1389`
- TLS port: `1636` (privileged port)
- Administrator connector port: `4444`
- Password: *nnnnnnn*
- Certificates: StartTLS and TLS
- Process: `java -server -Dorg.opends.server.scriptName=sta...`

```
# cd Oracle/Middleware/Oracle_OUD1
# export JAVA_HOME=/usr/jdk/instances/jdkversion
# ./oud-setup
```

3. **Verify that the LDAP server is listening.**

```
# ldapsearch -x -p 1389 -D "cn=directory manager" -h $HOSTNAME -b "" -s base
 objectclass='*'
```

4. **Add the initial profile entries to the OUD configuration.**

```
# pfedit profile.ldif
  dn: ou=profile,dc=example,dc=com
  ou: profile
  objectclass: top
  objectclass: organizationalUnit
# ldapmodify -a -h $HOSTNAME -D "cn=directory manager" -f profile.ldif
```

5. **Remove the newlines from all the attribute types in the `kerberos.ldif` file, then add the file to the OUD configuration.**

```
# pfedit /usr/share/lib/ldif/kerberos.ldif
# ldapmodify -p 1389 -a -h $HOSTNAME -D "cn=directory manager" \
   -f /usr/share/lib/ldif/kerberos.ldif
```

6. **Add the OUD server to the Kerberos configuration file.**

```
# pfedit /etc/krb5/krb5.conf
 [realms]
        EXAMPLE.COM = {
                kdc = krb1.example.com
                admin_server = krb1.example.com
                database_module = LDAP
        }

 [dbmodules]
        LDAP = {
```

```
                db_library = kldap
                ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
                ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
                ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
                ldap_cert_path = /var/ldap
                ldap_servers = ldap://krb1:1389
        }
...
```

7. Create the keys and stash files for LDAP binding to the KDC and `kadmin` services.

```
# kdb5_ldap_util -D "cn=directory manager" create -P nnnnnnnn -r EXAMPLE.COM -s
# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

8. Modify the `ldap_servers` entry in the Kerberos configuration file to use a privileged port.

```
# pfedit /etc/krb5/krb5.conf
       ldap_servers = ldaps://krb1:1636
```

9. Add Kerberos entries to the OUD server.

a. Create and add KDC roles.

```
# pfedit kdc_roles.ldif
 dn: cn=kdc service,ou=profile,dc=example,dc=com
 cn: kdc service
 sn: kdc service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

 dn: cn=kadmin service,ou=profile,dc=example,dc=com
 cn: kadmin service
 sn: kadmin service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

# ldapmodify -p 1389 -a -h $HOSTNAME -D "cn=directory manager" -f kdc_roles.ldif
```

b. Create and add administrative users.

```
# pfedit example.ldif
dn: dc=example,dc=com
changetype: modify
```

```
replace: aci
aci: (target ="ldap:///dc=example,dc=com")(targetattr !=
 "userPassword")(version 3.0;acl "Anonymous read-search access";
 allow (read, search, compare)(userdn = "ldap:///anyone");)
aci: (target="ldap:///dc=example,dc=com") (targetattr =
  "*")(version 3.0; acl "allow all Admin group"; allow(all) groupdn =
  "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)

dn: ou=Groups, dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: Groups

dn: cn=Directory Administrators, ou=Groups, dc=example,dc=com
cn: Directory Administrators
objectclass: top
objectclass: groupofuniquenames
ou: Groups
uniquemember: uid=kvaughan, ou=People, dc=example,dc=com
uniquemember: uid=rdaugherty, ou=People, dc=example,dc=com
uniquemember: uid=hmiller, ou=People, dc=example,dc=com

dn: ou=People, dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: People
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr =
 "userpassword || telephonenumber || facsimiletelephonenumber")(version 3.0;
 acl "Allow self entry modification";allow (write)(userdn = "ldap:///self");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Accounting)")(version 3.0;
 acl "Accounting Managers Group Permissions";allow (write) (groupdn =
  "ldap:///cn=Accounting Managers,ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Human Resources)")(version 3.0;
 acl "HR Group Permissions";allow (write)(groupdn = "ldap:///cn=HR Managers,
 ou=groups,dc=example,dc=com
 ");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn ||sn || uid")(targetfilter ="(ou=Product Testing)")(version 3.0;
 acl "QA Group Permissions";allow (write)(groupdn = "ldap:///cn=QA Managers,
 ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Product Development)")(version 3.0;
 acl "Engineering Group Permissions";allow (write)(groupdn = "ldap:///
 cn=PD Managers,ou=groups,dc=example,dc=com");)

dn: ou=Special Users,dc=example,dc=com
```

```
objectclass: top
objectclass: organizationalUnit
ou: Special Users
description: Special Administrative Accounts

# ldapmodify -p 1389 -a -h $HOSTNAME -D "cn=directory manager" -f example.ldif
```

c. Create and add ACLs for LDAP entries.

```
# pfedit kadmin.aci
## Set kadmin ACL for everything under krbcontainer.
dn: cn=krbcontainer,dc=example,dc=com
changetype: modify
replace: aci
aci: (target="ldap:///cn=krbcontainer,dc=example,dc=com") (targetattr="*")
 (version 3.0; acl "kadmin_ACL"; allow (all)
 userdn="ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

## Set kadmin ACL for everything under the people subtree if there are
## mix-in entries for krb princs:
dn: ou=people,dc=example,dc=com
changetype: modify
replace: aci
aci: (target="ldap:///ou=people,dc=example,dc=com") (targetattr="*")
 (version 3.0; acl "kadmin_ACL"; allow (all)
 userdn="ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

# ldapmodify -h $HOSTNAME -D "cn=directory manager" -f kadmin.aci
```

10. Generate and store the TLS certificate for the OUD server.

This set of commands also creates the key manager provider, trust manager provider, and connection handler.

```
# export LDAPHOME=~OUD-base-location/ORACLE_HOME
# export LDAPHOME=$PWD
# export LDAP_SERVER_DN=krb1.example.com
# export STORE_PASSWD=xxxxxxxx
# export LDAP_BINDPWF=$LDAPHOME/config/keystore.pin
# export LDAP_ADMIN_PORT=4444
# export LDAP_BINDDN="cn=directory manager"
# export LDAP_SERVER=krb1.example.com
# rm $LDAPHOME/config/keystore
# rm $LDAPHOME/config/truststore
# echo $STORE_PASSWD > LDAP_BINDPWF
# keytool -genkeypair -alias server-cert -keyalg rsa \
   -dname "CN=$LDAP_SERVER_DN" -keystore $LDAPHOME/config/keystore \
   -storepass $STORE_PASSWD -keypass $STORE_PASSWD
```

```
# keytool -selfcert -alias server-cert -validity 1825 \
   -keystore $LDAPHOME/config/keystore -storetype JKS -storepass $STORE_PASSWD
# keytool -list -alias server-cert -keystore $LDAPHOME/config/keystore \
   -storepass $STORE_PASSWD
# keytool -exportcert -alias server-cert -file $LDAPHOME/config/server-cert.txt \
   -rfc -keystore $LDAPHOME/config/keystore -storepass $STORE_PASSWD
# cp $LDAPHOME/config/server-cert.txt /var/ldap/certdb.pem
```

11. **Enable the key manager provider, trust manager provider, and connection handler.**

```
# ldapservercfg -X -n -h $LDAP_SERVER -p $LDAP_ADMIN_PORT -D "$LDAP_BINDDN" \
  -j $LDAP_BINDPWF set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set key-manager-provider:JKS --set trust-manager-provider:JKS \
  --set listen-port:1636 --set enabled:true
# bin/stop-ds
```

12. **(Optional) Verify the configuration with an SSL LDAP query.**

```
# /usr/lib/openldap/bin/ldapsearch -x -v -x -D "$LDAP_BINDDN" -w $LDAP_BINDPW \
   -H ldapi://$LDAP_SERVER_DN:1636 -b "" -s base objectclass='*'
```

13. **Synchronize this system's clock with other clocks in the realm.**

Note - A master KDC cannot be the clock synchronization server.

For more information and pointers to procedures, see “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 59. See also the `krb5.conf`(4) man page.

▼ How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type on an OpenLDAP Server

In this procedure, the `krbprincipalaux`, and `krbTicketPolicyAux` and `krbPrincipalName` attributes are associated with the people object class.

This procedure uses the following configuration parameters:

- OpenLDAP Server = `krb1.example.com`
- User principal = `mre@EXAMPLE.COM`

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Prepare each entry in the people object class.**

   On the OpenLDAP server, repeat this step for each entry.

   ```
   cat << EOF | ldapmodify -h openldap.example.com \
   -D "cn=directory manager,dc=example,dc=com"
   dn: uid=mre,ou=people,dc=example,dc=com
   changetype: modify
   objectClass: krbprincipalaux
   objectClass: krbTicketPolicyAux
   krbPrincipalName: mre@EXAMPLE.COM
   EOF
   ```

2. **Add a subtree attribute to the realm container.**

   This example enables searching principal entries in the `ou=people,dc=example,dc=com` container, as well as in the default `EXAMPLE.COM` container.

   ```
   # kdb5_ldap_util -D "cn=directory manager" modify \
      -subtrees 'ou=people,dc=example,dc=com' -r EXAMPLE.COM
   ```

3. **(Optional) If the KDC records are stored in DB2, migrate the DB2 entries.**

   a. **Dump the DB2 entries.**

      ```
      # kdb5_util dump > dumpfile
      ```

   b. **Load the database into the LDAP server.**

      ```
      # kdb5_ldap_util load -update dumpfile
      ```

4. **(Optional) Add the principal attributes to the KDC.**

   ```
   # kadmin.local -q 'addprinc mre'
   ```

How to Destroy a Kerberos Realm on an LDAP Directory Server

Use this procedure if a different LDAP Directory Server is handling a realm.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

- **Destroy the realm.**

```
# kdb5_ldap_util -D "cn=directory manager" destroy
```

Configuring Kerberos Clients

Kerberos clients include any host on the network that is not a KDC server and that provides or uses Kerberos services. This section provides procedures for installing a Kerberos client, as well as information about using `root` authentication to mount NFS file systems. For an overview of client configuration options, see "Planning for Kerberos Clients" on page 56.

The following task map describes the tasks that are covered in this section.

TABLE 4 Task Map: Configuring Kerberos Clients

Task	Description	For Instructions
Install clients by using the Automated Installer (AI).	Appropriate when you want the Kerberos client to be configured during system installation.	"How to Configure Kerberos Clients Using AI" in *Installing Oracle Solaris 11.3 Systems*
Create an installation profile for similar Kerberos clients.	Creates a kclient installation profile.	"How to Create a Kerberos Client Installation Profile" on page 81
Install clients with a script.	Appropriate when the installation parameters for each client are the same.	"How to Use a Kerberos Client Profile" on page 81
Install clients by answering prompts.	Appropriate when only a few of the installation parameters need to change.	"How to Use the kclient Utility Without an Installation Profile" on page 83
Install clients manually.	Appropriate when each client installation requires unique installation parameters.	How to Manually Configure a KDC
Join a Kerberos client to an Active Directory Server.	Automatically installs a Kerberos client of an Active Directory server.	"How to Join a Kerberos Client to an Active Directory Server" on page 86
Enable a client to access an NFS file system as the `root` user	Enables the client to mount an NFS file system with `root` access. Also, enables the client to access the NFS file system so that cron jobs can run.	"How to Access a Kerberos Protected NFS File System as the `root` User" on page 88
Enable a client without a fixed IP address to use Kerberos services.	Changes the default security requirement to verify the KDC of the client's ticket.	"Verifying Kerberos Clients Without a Host Principal" on page 87

How to Create a Kerberos Client Installation Profile

This procedure creates a kclient profile that can be used when you install a Kerberos client. By using the profile, you reduce the likelihood of typing errors. Also, using the profile reduces user intervention as compared to the interactive process.

Note - To create systems that initially boot as fully configured Kerberos clients, see "Configuring Security" in *Installing Oracle Solaris 11.2 Systems* .

Before You Begin A KDC server is installed and configured.

The `security/kerberos-5` package must be on your system. If you installed the `security/kerberos-5/kdc` package, the `kerberos-5` package with the client features is installed.

You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Create a `kclient` installation profile.**

   The following is a sample `kclient` profile:

```
client# pfedit kcprofile
REALM EXAMPLE.COM
KDC kdc1.example.com
ADMIN clntconfig
FILEPATH /net/denver.example.com/export/install/krb5.conf
NFS 1
DNSLOOKUP none
```

2. **Protect the file and store it for use by other clients.**

```
client# cp kcprofile /net/denver.example.com/export/install
denver# chown root kcprofile; chmod 644 kcprofile
```

How to Use a Kerberos Client Profile

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

You have access to a `kclient` profile, such as the profile in "How to Create a Kerberos Client Installation Profile" on page 81.

- **Run the `kclient` command with a profile argument.**

  You must provide the password for the `clntconfig` principal to complete the process. You created this password when you configured your master KDC. For more information, see the `kclient`(1M) man page.

```
client# /usr/sbin/kclient -p /net/denver.example.com/export/install/kcprofile

Starting client setup
-------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM: xxxxxxxx

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

-------------------------------------------------
Setup COMPLETE.

client#
```

Example 7 Sample Use of `kclient` Utility

The following example uses the `kcprofile` client profile and two command-line overrides to configure the client.

```
# /usr/sbin/kclient -p /net/denver.example.com/export/install/kcprofile \
-d dns_fallback -k kdc2.example.com

Starting client setup
-------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:  xxxxxxxx
```

```
nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

-------------------------------------------------
Setup COMPLETE.

client#
```

How to Use the kclient Utility Without an Installation Profile

This procedure uses the `kclient` installation utility without an installation profile. If the client is to join an Active Directory server, go to "How to Join a Kerberos Client to an Active Directory Server" on page 86.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Run the `kclient` command with no arguments.**

   ```
   client# /usr/sbin/kclient
   ```

   The script prompts you for the following information:

   - Kerberos realm name
   - KDC master host name
   - KDC slave host names
   - Domains to map to the local realm
   - PAM service names and options to use for Kerberos authentication

   For more information, see the `kclient`(1M) man page.

2. **If the KDC server is not running an Oracle Solaris release, answer `y` and define the type of server that is running the KDC.**

   For the list of available servers, see the `-T` option in the `kclient`(1M) man page.

3. **If DNS should be used for Kerberos lookups, answer `y` and indicate the DNS lookup option to use.**

   Valid options are `dns_lookup_kdc`, `dns_lookup_realm`, and `dns_fallback`. Use the `man` command to view a description of these values in the `krb5.conf`(4) man page.

4. **Define the name of the Kerberos realm and the master KDC host name.**

   This information is added to the `/etc/krb5/krb5.conf` configuration file.

5. **If slave KDCs are in the realm, answer `y` and provide the slave KDC host names.**

   This information is used to create additional KDC entries in the client's configuration file.

6. **If service or host keys are required, answer `y`.**

   Tip - For security, all clients should have a host key. See "Verifying Kerberos Clients Without a Host Principal" on page 87.

   Service or host keys are *required* only when the client system is hosting Kerberized services.

7. **If the client is a member of a cluster, answer `y` and provide the logical name of the cluster.**

   The logical host name is used when creating service keys, which is required when hosting Kerberos services from clusters.

8. **Identify any domains or hosts to map to the current realm.**

   This mapping enables the client to recognize other domains as belonging to the client's default domain.

9. **Specify whether the client will use Kerberized NFS.**

   NFS service keys need to be created if the client will host NFS services using Kerberos.

10. **Indicate whether a new PAM policy needs to be created.**

    To set which PAM services use Kerberos for authentication, you provide the service name and a flag that indicates how Kerberos authentication is to be used. The valid flag options are:

    - `first` – Use Kerberos authentication first, and only use UNIX if Kerberos authentication fails
    - `only` – Use Kerberos authentication only
    - `optional` – Use Kerberos authentication optionally

    For information about provided PAM services for Kerberos, review the files in the `/etc/security/pam_policy` directory.

11. Specify whether the master `/etc/krb5/krb5.conf` file should be copied.

This option enables specific configuration information to be used when the arguments to `kclient` are not sufficient.

Example 8 Sample Run of the `kclient` Script

```
...
Starting client setup
--------------------------------------------------

Is this a client of a non-Solaris KDC ? [y/n]: n
No action performed.
Do you want to use DNS for kerberos lookups ? [y/n]: y
...
Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC host name for the above realm: kdc1.example.com

Note, this host and the KDC's time must be within 5 minutes of each other for
Kerberos to function. Both hosts should run some form of time synchronization
system like Network Time Protocol (NTP).
Do you have any slave KDC(s) ? [y/n]: y
Enter a comma-separated list of slave KDC host names: kdc2.example.com

Will this client need service keys ? [y/n]: n
No action performed.
Is this client a member of a cluster that uses a logical host name ? [y/n]: n
No action performed.
Do you have multiple domains/hosts to map to realm ? [y/n]: y
Enter a comma-separated list of domain/hosts to map to the default realm: corphdqtrs.
example.com, \
example.com

Setting up /etc/krb5/krb5.conf.

Do you plan on doing Kerberized nfs ? [y/n]: y
Do you want to update /etc/pam.conf ? [y/n]: y
Enter a comma-separated list of PAM service names in the following format:
service:{first|only|optional}: gdm:first
Configuring /etc/pam.conf.

Do you want to copy over the master krb5.conf file ? [y/n]: n
No action performed.

--------------------------------------------------
Setup COMPLETE.
```

How to Join a Kerberos Client to an Active Directory Server

This procedure uses the `kclient` command without an installation profile.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. (Optional) Enable DNS resource record creation for the client.

```
client# sharectl set -p ddns_enable=true smb
```

2. Run the `kclient` command.

The following output shows sample output from running the `kclient` command to join the client to the AD domain, `EXAMPLE.COM`.

The `-T` option selects a KDC server type, in this case, a Microsoft Active Directory (AD) server type. By default, you must provide the password for the Administrator principal of the AD server.

```
client# /usr/sbin/kclient -T ms_ad
Starting client setup
-------------------------------------------------

Attempting to join 'CLIENT' to the 'EXAMPLE.COM' domain.
Password for Administrator@EXAMPLE.COM: xxxxxxxx
Forest name found: example.com
Looking for local KDCs, DCs and global catalog servers (SVR RRs).

Setting up /etc/krb5/krb5.conf

Creating the machine account in AD via LDAP.
-------------------------------------------------
Setup COMPLETE.
#
```

For more information, see the `kclient`(1M) man page.

3. Add the Kerberos credentials.

The `kclient` does not store the Kerberos credentials for later use.

```
# smbadm join
```

Example 9 Sample Kerberos Client of a Non-Oracle Solaris KDC

A Kerberos client can be set up to work with a non-Oracle Solaris KDC by adding a line to the `/etc/krb5/krb5.conf` file in the `realms` section. This line changes the protocol that is used when the client is communicating with the Kerberos password-changing server. The following excerpt shows the format of this line.

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

Verifying Kerberos Clients Without a Host Principal

By default, Kerberos checks that the KDC of the host principal that is stored in the local `/etc/krb5/krb5.keytab` file is the same KDC that issued the ticket-granting ticket (TGT). This check, `verify_ap_req_nofail`, prevents DNS spoofing attacks.

However, this check must be disabled for client configurations where the host principal is unavailable. The following configurations require this check to be disabled:

- The client IP address is dynamically assigned, for example, a DHCP client.
- The client is not configured to host any services, so no host principal was created.
- The host key is not stored on the client.

To disable TGT verification, set the `verify_ap_req_nofail` option to `false` in the `krb5.conf` file. The `verify_ap_req_nofail` option can be entered in either the `[libdefaults]` or the `[realms]` section of the `krb5.conf` file. In the `[libdefaults]` section, the setting is used for all realms:

```
client # pfedit /etc/krb5/krb5.conf
[libdefaults]
default_realm = EXAMPLE.COM
verify_ap_req_nofail = false
...
```

If the option is in the `[realms]` section, the setting applies only to the defined realm. For more information about Use the `man` command to view a description of this option in the `krb5.conf`(4) man page.

How to Access a Kerberos Protected NFS File System as the root User

This procedure enables a client to access an NFS file system that requires Kerberos authentication with the root principal and in particular, when the NFS file system is shared with options like: -o sec=krb5p,root=client1.example.com.

1. **Run the kadmin command.**

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

2. **Create a root principal for the NFS client.**

   This principal is used to provide root equivalent access to NFS-mounted file systems that require Kerberos authentication. The root principal should be a two-component principal. The second component should be the host name of the Kerberos client system to avoid the creation of a realm-wide root principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters regardless of the case of the domain name in the naming service.

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin:
```

3. **Add the root principal to the server's keytab file and quit kadmin.**

   This step is required for the client to have root access to NFS-mounted file systems. This step is also required for non-interactive root access, such as running cron jobs as root.

```
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

How to Configure Automatic Migration of Users in a Kerberos Realm

Users who do not have a Kerberos principal can be automatically migrated to an existing Kerberos realm by using PAM. You customize per-system PAM configuration files on the migration server and the master server to handle the recognition of UNIX credentials and the re-authentication in the Kerberos realm.

For information about PAM, see Chapter 1, "Using Pluggable Authentication Modules" in *Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3* and the `pam.conf`(4) man page.

In this procedure, the login service names are configured to use automatic migration. This example uses the following configuration parameters:

- Realm name = `EXAMPLE.COM`
- Master KDC = `kdc1.example.com`
- Machine hosting the migration service = `server1.example.com`
- Migration service principal = `host/server1.example.com`

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Ensure that a host service principal for `server1` exists.**

   The host service principal in the `keytab` file of `server1` is used to authenticate the server to the master KDC.

   ```
   server1 # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   KVNO Principal
   ---- -------------------------------------------------
   3 host/server1.example.com@EXAMPLE.COM
   ...
   ```

   For information about the options to the `klist` command, see the `klist`(1) man page.

2. **If `server1` is not listed, configure it as a Kerberos client of the realm `EXAMPLE.COM`.**

   For the steps, see the examples in "Configuring Kerberos Clients" on page 80.

3. **Modify the PAM policy for `server1`.**

   For more information, see "Assigning a Per-User PAM Policy" in *Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3*.

a. Determine which Kerberos policy is in use on `server1`.

```
% grep PAM_POLICY /etc/security/policy.conf
# PAM_POLICY specifies the system-wide PAM policy (see pam_user_policy(5))
...
PAM_POLICY=krb5_first
```

b. Copy that PAM policy file, then modify the new policy file to append the `pam_krb5_migrate.so.1` module to each authentication stack.

```
server1 # cd /etc/security/pam_policy/; cp krb5_first krb5_firstmigrate
server1 # pfedit /etc/security/pam_policy/krb5_firstmigrate
# login service (explicit because of pam_dial_auth)
...
login auth required  pam_unix_auth.so.1
login    auth optional  pam_krb5_migrate.so.1
#
# PPP service (explicit because of pam_dial_auth)
...
ppp auth required  pam_unix_auth.so.1
ppp auth optional  pam_krb5_migrate.so.1
#
# GDM Autologin (explicit because of pam_allow). ...
#
gdm-autologin auth  required    pam_unix_cred.so.1
gdm-autologin auth  sufficient  pam_allow.so.1
gdm-autologin auth optional    pam_krb5_migrate.so.1
#
# Default definitions for Authentication management
...
OTHER   auth required          pam_unix_auth.so.1
OTHER   auth optional    pam_krb5_migrate.so.1
#
# passwd command (explicit because of a different authentication module)
#
passwd  auth required          pam_passwd_auth.so.1
passwd  auth optional    pam_krb5_migrate.so.1
#
...
```

c. (Optional) Edit the `krb5_firstmigrate` file to force an immediate password change.

For the newly created Kerberos accounts, set the password expiration time to the current time by adding the `expire_pw` option to the `pam_krb5_migrate` entries. For more information, see the `pam_krb5_migrate(5)` man page.

```
service-name auth optional     pam_krb5_migrate.so.1 expire_pw
```

d. In this policy file, modify the `OTHER` account stack to block access if the Kerberos password has expired.

```
# Definition for Account management
# Used when service name is not explicitly mentioned for account management
# Re-ordered pam_krb5 causes a Kerberos password expiration to block access
#
OTHER   account requisite   pam_roles.so.1
OTHER   account required pam_krb5.so.1
OTHER   account required pam_unix_account.so.1
OTHER   account required pam_tsol_account.so.1
## OTHER account required pam_krb5.so.1
#
...
```

e. Change the `PAM_POLICY` entry in the `policy.conf` file to use the modified configuration file.

```
server1 # pfedit /etc/security/policy.conf
...
# PAM_POLICY=krb5_first
PAM_POLICY=krb5_firstmigrate
```

For more information, review the comments in the `policy.conf` file.

4. On the master KDC, update the `kadm5.acl` access control file.

The following entries grant migrate and inquire privileges to the `host/server1.example.com` service principal for all users except the `root` user. Use the `U` privilege to list users who must not be migrated. These exceptions must precede the permit all or `ui` entry. Use the `man` command to view a description of this file in the `kadm5.acl`(4) man page.

```
kdc1# pfedit /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```

5. On the master KDC, enable the `kadmind` daemon to use the `k5migrate` PAM service.

If a `k5migrate` service file is not in the `/etc/pam.d` directory, add the service file to the directory. The contents are as follows:

```
kdc1# cat /etc/pam.d/k5migrate
...
## Permits validation of migrated UNIX accounts
```

```
auth    required        pam_unix_auth.so.1
account required        pam_unix_account.so.1
```

This modification enables the validation of UNIX user passwords for accounts that require migration. For more information, see the `pam.d(4)` man page.

Note - `k5migrate` is the name of a PAM service. The file must be named `k5migrate`.

6. **Test your configuration before putting it in production.**

   - **As a regular user, test each modified PAM service.**
   - **As `root`, test each modified PAM service.**
   - **Force a password change, then test the modified PAM services.**

Configuring Kerberos Network Application Servers

Network application servers are hosts that provide access using secure network applications, such as `ftp`. Only a few steps are required to enable the Kerberos version of these applications on a server.

▼ How to Configure a Kerberos Network Application Server

This procedure uses the following configuration parameters:

- Application server = `boston`
- `admin` principal = `kws/admin`
- DNS domain name = `example.com`
- Realm name = `EXAMPLE.COM`

Before You Begin Make sure the master KDC is configured and the clocks are synchronized as described in "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 59. To fully test the process, you need several clients.

You must assume the `root` role on the application server. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Determine if a host principal exists for the new server.**

   The following command reports the existence of the host principal:

   ```
   boston # klist -k | grep host
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   ```

   If the command does returns a principal, you are done. If it does not return a principal, then create new principals by using the following steps.

2. **Log in to the server with one of the `admin` principal names that you created when configuring the master KDC.**

   ```
   boston # /usr/sbin/kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin:
   ```

3. **Create the server's `host` principal.**

   ```
   kadmin: addprinc -randkey host/boston.example.com
   Principal "host/boston.example.com" created.
   kadmin:
   ```

   The `host` principal is used in the following ways:

   - To authenticate traffic when using remote commands such as `ftp`.
   - By `pam_krb5` to prevent KDC spoofing attacks by using the `host` principal to verify that a user's Kerberos credential was obtained from a trusted KDC.
   - To enable the `root` user to automatically acquire a Kerberos credential without requiring that a `root` principal exist. This capability can be useful when doing a manual NFS mount where the share requires a Kerberos credential.

   This principal is required if traffic using the remote application is to be authenticated using the Kerberos service. If the server has multiple host names associated with it, then create a principal for each host name using the FQDN form of the host name.

4. **Add the server's `host` principal to the server's keytab file and quit `kadmin`.**

   If the `kadmin` command is not running, restart it with a command similar to the following: `/usr/sbin/kadmin -p` *kws*`/admin`

If the server has multiple host names associated with it, then add a principal to the keytab for each host name.

```
kadmin: ktadd host/boston.example.com
Entry for principal host/boston.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

How to Use the Generic Security Service With Kerberos When Running FTP

The generic security service (GSS) can be used by Kerberos network applications for authentication, integrity, and privacy. The following steps show how to enable the GSS service for ProFTPD.

Before You Begin You must assume the `root` role on the FTP server. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Add principals for the FTP server and create the FTP server's keytab file.**

   These steps might not be needed if the changes were made earlier.

   a. **Start the `kadmin` command.**

   ```
   ftpserver1 # /usr/sbin/kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin:
   ```

   b. **Add the `ftp` service principal for the FTP server.**

   ```
   kadmin: addprinc -randkey ftp/ftpserver1.example.com
   ```

   c. **Add the `ftp` service principal to a new keytab file.**

   A new keytab file makes this information available to the `ftp` service without exposing all of the information in the server's keytab file.

   ```
   kadmin: ktadd -k /etc/krb5/ftp.keytab ftp/ftpserver1.example.com
   ```

For more information, see the `ktadd` command in the `kadmin`(1M) man page.

2. **Change ownership of the new keytab file.**

```
ftpserver1 # chown ftp:ftp /etc/krb5/ftp.keytab
```

3. **Enable GSS for the FTP server.**

Make the following changes to the `/etc/proftpd.conf` file.

```
# pfedit /etc/proftpd.conf
LoadModule     mod_gss.c

GSSEngine      on
GSSKeytab      /etc/krb5/ftp.keytab
```

4. **Restart the FTP server.**

```
# svcadm restart network/ftp
```

Configuring Kerberos NFS Servers

NFS services use UNIX user IDs (UIDs) to identify a user and cannot directly use GSS credentials. To translate the credential to a UID, you might need to use the `auth_to_local` relation or a custom `auth_to_local` plugin. Kerberos NFS servers can be protected with multiple security modes.

This section contains two procedures:

- "How to Configure Kerberos NFS Servers" on page 95
- "How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 97

▼ How to Configure Kerberos NFS Servers

This procedure uses the following configuration parameters:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- NFS server = `denver.example.com`

- `admin` principal = `kws/admin`

Before You Begin You must assume the `root` role on the NFS server. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

Make sure the master KDC is configured and the clocks are synchronized as described in "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 59. To fully test the process, you need several clients.

1. **Configure the NFS server as a Kerberos client.**

   Follow the instructions in "Configuring Kerberos Clients" on page 80.

2. **Add the NFS service principal.**

   Use the `kadmin` command.

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

   a. **Create the NFS service principal.**

      Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters regardless of the case of the domain name in the naming service.

      Repeat this step for each unique interface on the host that might be used to access NFS data. If a host has multiple interfaces with unique names, each unique name must have its own NFS service principal.

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

   b. **Add the server's NFS service principal to the server's keytab file and quit `kadmin`.**

      Repeat this step for each unique service principal that you created in Step 2a.

      ```
      kadmin: ktadd nfs/denver.example.com
      Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS
       mode
      with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
      Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS
       mode
      with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
      Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES
       cbc
      mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
      ```

```
kadmin: quit
```

3. **Share the NFS file system with Kerberos security modes.**

   For more information, see "How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 97.

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

This procedure enables an NFS server to provide secure NFS access by using several security modes. When a client negotiates a security mode with the NFS server, the client uses the first mode that is offered by the server. This mode is used for all subsequent client requests of the file system shared by that server.

Before You Begin You must assume the `root` role on the NFS server. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Verify that an NFS service principal entry is in the keytab file.**

   The `klist` command reports if a keytab file exists and displays the principals. If the results show that no keytab file exists or that no NFS service principal exists, you need to verify the completion of all the steps in "How to Configure Kerberos NFS Servers" on page 95.

```
# klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab
KVNO Principal
---- -------------------------------------------------------
3 nfs/denver.example.com@EXAMPLE.COM
3 nfs/denver.example.com@EXAMPLE.COM
3 nfs/denver.example.com@EXAMPLE.COM
3 nfs/denver.example.com@EXAMPLE.COM
```

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

2. **Enable Kerberos security modes in the `/etc/nfssec.conf` file.**

   In the `/etc/nfssec.conf` file, remove the "#" that comments out the Kerberos security modes.

```
# pfedit /etc/nfssec.conf
.
.
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
```

```
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

3. **Share the file systems with the appropriate security modes.**

   - Choose `krb5p` to provide `krb5` authentication, integrity and privacy protection for confidential data transmitted over NFS. Use this mode unless it strains the server's processing resources.
   - Choose `krb5i` to provide `krb5` authentication and integrity protection in addition to the minimum protection that TCP/IP provides for NFS data.
   - Choose `krb5` for `krb5` authentication only. This security mode provides the least protection of the security modes but also has the smallest impact on the processor.

   share -F nfs -o sec=*mode file-system*

   mode — Specifies the security modes to be used when sharing the file system. When using multiple security modes, the first mode in the list is used as the default.

   file-system — Defines the path to the file system to be shared.

   All clients that attempt to access files from the named file system require Kerberos authentication. To access files, the user principal on the NFS client should be authenticated.

4. **(Optional) Mount a file system by using a security mode other than the default.**

   Do not perform this procedure if the default security mode is acceptable.

   - **If the automounter is being used, edit the `auto_master` database to enter a security mode other than the default.**

     file-system `auto_home -nosuid,sec=`*mode*

   - **Manually issue the `mount` command to access the file system by using a non-default mode.**

     # mount -F nfs -o sec=*mode file-system*

Example 10 Sharing a File System With One Kerberos Security Mode

In this example, authentication with the `krb5` security mode must succeed before any files can be accessed through the NFS service.

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

Example 11 Sharing a File System With Multiple Kerberos Security Modes

In this example, all three Kerberos security modes have been selected. The mode that is used is negotiated between the client and the NFS server. If the first mode in the command fails, then the next mode is tried. For more information, see the nfssec(5) man page.

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

Configuring Delayed Execution for Access to Kerberos Services

In the default Kerberos environment, credentials expire after a limited amount of time. For processes that can execute at arbitrary times, such as `cron` and `at`, the limited time presents a problem.

This procedure describes how to configure the Kerberos environment to support delayed execution processes that require authenticated services through Kerberos. Oracle Solaris provides PAM modules, uses service keys, and uses `kclient` configuration options to make delayed execution with Kerberos authentication possible and more secure than alternative solutions.

Note - If the `cron` server becomes compromised, an attacker could impersonate users to gain access to target services that are configured for the `cron` server. Therefore, consider that the `cron` host that is configured in this procedure as a more sensitive system, because it provides intermediate services for users.

▼ How to Configure a cron Host for Access to Kerberos Services

This procedure uses the following configuration parameters:

- `cron` host = `host1.example.com`
- NFS server = `host2.example.com`
- LDAP server = `host3.example.com`

Note - Delayed execution works only with an LDAP back end.

1. **Configure the `cron` service to support Kerberos.**

   - **If the `cron` host is not configured for Kerberos, then run the `kclient` command on the system.**

     For more information, see the `kclient(1M)` man page.

     For example, the following command configures the client in the `EXAMPLE.COM` realm. The command includes the `pam_gss_s4u` file in the `/etc/pam.d/cron` service file by using the `include` mechanism.

     ```
     # kclient -s cron:optional -R EXAMPLE.COM
     ```

   - **If the `cron` host is already configured for Kerberos, then you must modify the PAM configuration for the `cron` service on that host manually.**

     Ensure that the PAM configuration for the `cron` service includes the `pam_gss_s4u` file.

     ```
     # cd /etc/pam.d ; cp cron cron.orig
     # pfedit cron
           # PAM include file for optional set credentials
           # through Kerberos keytab and GSS-API S4U support
           auth include          pam_gss_s4u
     ```

2. **Enable the `cron` host to act as a delegate.**

   For example:

   ```
   # kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin: modprinc +ok_as_delegate host/host1.example.com@EXAMPLE.COM
   Principal host/host1.example.com@EXAMPLE.COM modified.
   ```

3. **Enable the `cron` host to request tickets for itself on behalf of the user who created the `cron` job.**

   ```
   kadmin: modprinc +ok_to_auth_as_delegate host/host1.example.com@EXAMPLE.COM
   Principal host/host1.example.com@EXAMPLE.COM modified.
   kadmin: quit
   ```

4. **In LDAP, configure the `cron` host to specify the services that it uses as a delegate.**

   For example, to enable the `cron` host to access the user's home directory on `host2`, a Kerberized NFS server, add the NFS host to the `krbAllowedToDelegateTo` parameter in the `cron` server's LDAP definition.

   a. **Create the delegate assignment.**

```
# pfedit /tmp/delghost.ldif
dn: krbprincipalname=host/host1.example.com@EXAMPLE.COM,cn=EXAMPLE.COM,
cn=krbcontainer,dc=example,dc=com
changetype: modify
krbAllowedToDelegateTo: nfs/host2.example.com@EXAMPLE.COM
```

b. Add the assignment to LDAP.

```
# ldapmodify -h host3 -D "cn=directory manager" -f delghost.ldif
```

Administering the Kerberos Database

The Kerberos database is the backbone of Kerberos and must be maintained properly. This section provides some procedures for administering the Kerberos database, such as backing up and restoring the database, setting up incremental or parallel propagation, and administering the stash file. The steps to initially set up the database are in MIT Kerberos Installation Guide.

How to Convert a Kerberos Database After a Server Upgrade

If your KDC database was created on a server that was running an old release, converting the database enables you to take advantage of the improved database format.

Before You Begin Use this procedure only if the database is using an older format.

On the KDC master, you must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. On the master, stop the KDC daemons.

```
kdc1# svcadm disable network/security/krb5kdc
kdc1# svcadm disable network/security/kadmin
```

2. Create a directory to store a temporary copy of the database.

```
kdc1# mkdir /var/krb5/tmp
kdc1# chmod 700 /var/krb5/tmp
```

3. Dump the KDC database.

```
kdc1# kdb5_util dump /var/krb5/tmp/prdb.txt
```

4. **Save copies of the current database files.**

```
kdc1# cd /var/krb5
kdc1# mv princ* tmp/
```

5. **Load the database.**

```
kdc1# kdb5_util load /var/krb5/tmp/prdb.txt
```

6. **Start the KDC daemons.**

```
kdc1# svcadm enable -r network/security/krb5kdc
kdc1# svcadm enable -r network/security/kadmin
```

Observing Mapping From GSS Credentials to UNIX Credentials

To be able to monitor the credential mappings, first uncomment this line from the `/etc/gss/gsscred.conf` file.

```
SYSLOG_UID_MAPPING=yes
```

Next, make the `gssd` service read the `/etc/gss/gsscred.conf` file.

```
# pkill -HUP gssd
```

Now you can monitor the credential mappings as `gssd` requests them. The mappings are recorded by the `rsyslog` daemon, if the `rsyslog.conf` file is configured for the `auth` system facility with the `debug` severity level.

Increasing Security on Kerberos Servers

This section provides advice about increasing security on Kerberos application servers and on KDC servers.

Restricting Access to KDC Servers

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

- Restrict physical access to the hardware that supports the KDC.

  Make sure that the KDC server and its monitor are located in a secure facility. Regular users should not be able to access this server in any way.
- Store KDC database backups on local disks or on the KDC slaves.

  Make tape backups of your KDC only if the tapes are stored securely. Follow the same practice for copies of keytab files.

  Store these files on a local file system that is not shared with other systems. The storage file system can be on either the master KDC server or any of the slave KDCs.

Using a Dictionary File to Increase Password Security

A dictionary file can be used by the Kerberos service to prevent words in the dictionary from being used as passwords for new credentials. Preventing the use of dictionary words as passwords makes it harder for someone else to guess any password. By default, the `/var/krb5/kadm5.dict` file is used, but it is empty.

Add a line to the KDC configuration file, `kdc.conf` to instruct the service to use a dictionary file. In this example, the administrator uses the dictionary that is included with the `spell` utility, then restarts the Kerberos services. For a full description of the configuration file, use the `man` command to view the `kdc.conf`(4) man page.

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

[realms]
     EXAMPLE.COM = {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        iprop_enable = true
        iprop_master_ulogsize = 1000
        dict_file = /usr/share/lib/dict/words
     }
kdc1#
kdc1# svcadm restart -r network/security/krb5kdc
kdc1# svcadm restart -r network/security/kadmin
```

♦ ♦ ♦ CHAPTER 5

Users Using Kerberos

This chapter is intended for Kerberos users. It briefly explains Kerberos password and ticket management, and describes remote access to Kerberos applications.

Kerberos Password and Ticket Management

Kerberos is a *single sign-on* environment, which means that you type your password only once when using network applications. Kerberos authentication and encryption is built into each of a suite of existing, familiar network applications. The Kerberos V5 applications are versions of existing UNIX network applications with Kerberos features added.

Administrative Responsibilities for Kerberos Password and Ticket Management

The administrator configures Kerberos to handle user passwords and tickets.

- In Oracle Solaris, Kerberos is built into the `login` command.

  If the administrator configures the PAM service for the applicable login services, users can obtain tickets automatically. For more information, see the `pam_krb5`(5) man page.
- If the administrator configures the `ssh` command to forward copies of user tickets to the other hosts, then users do not have to explicitly ask for tickets to get access to those hosts.

  For security reasons, the administrator might prevent ticket forwarding. For more information, see the discussion about agent forwarding in the `ssh`(1) man page.

User Responsibilities for Kerberos Ticket Management

Typically, Kerberos creates a ticket for you when you log in, so you need not do anything special to obtain a ticket.

User responsibilities for Kerberos tickets include the following:

- Create a ticket if your ticket expires.

  The `kinit` command prompts you for a password, then creates the ticket.
- Create a ticket for a different principal.

  When you use a different principal besides your default principal, you might need to create a ticket. For example, you might use the `ssh -l` command to log in to a host as another user.
- Create a ticket for a new host when your tickets are not forwarded.

  If the administrator configures the `ssh` command to forward copies of your tickets to the other hosts, then you do not have to explicitly ask for tickets to get access to those hosts. For security reasons, the administrator might prevent ticket forwarding. For more information, see the discussion about agent forwarding in the `ssh`(1) man page.
- List the properties of your ticket, such as whether it can be forwarded or is invalid.

  Not all tickets are alike. For example, one ticket might be `forwardable`, another ticket might be `postdated`, and a third ticket might be both forwardable and postdated. You can list the properties of your tickets with the `klist -f` command.
- Destroy your tickets at the end of a session.

  The `kdestroy` command destroys your credential cache, which destroys all your credentials and tickets. While this destruction is not usually necessary, running `kdestroy` reduces the chance of the credential cache being compromised during times that you are not logged in.

  If you are going to be away from your system, you should either use the `kdestroy` command or lock the screen with a screen saver.

For more information, see the MIT Kerberos User Commands Documentation (`http://web.mit.edu/kerberos/krb5-1.14/doc/user/user_commands/index.html`).

User Responsibilities for Kerberos Password Management

In a Kerberos environment, you have two passwords: the regular Oracle Solaris UNIX password and a Kerberos password. You can make both passwords the same, or they can be different.

Note - The behavior of the `passwd` command depends on how the PAM module is configured. The administrator might require users to change both passwords. For some sites, the UNIX password must be changed, while other sites require the Kerberos password to change.

If PAM is properly configured, you can change your Kerberos password in two ways.

- Use the `passwd` command. With the Kerberos service configured, the `passwd` command also automatically prompts for a new Kerberos password.

  By using the `passwd` command, you can set both your UNIX and Kerberos passwords at the same time. You can also change only one password and leave the other password untouched.
- Use the `kpasswd` command. `kpasswd` changes only Kerberos passwords. You must use `passwd` if you want to change your UNIX password.

  A primary use for `kpasswd` is to change a password for a Kerberos principal that is not a valid UNIX user. For example, `jdoe/admin` is a Kerberos principal but not an actual UNIX user, so you must use `kpasswd` to change the password.

For more information, see the MIT Kerberos User Commands Documentation.

After you change your password, the password must propagate through the network. The size of the Kerberos network affects the time that is required for the propagation.

Tip - If you need new Kerberos tickets shortly after you change your password, try the new password first. If the new password doesn't work, try again using the old password.

Kerberos policy defines the criteria for passwords. The administrator configures the policy. Password character classes are lowercase, uppercase, numbers, punctuation, and all other characters.

User Remote Logins in Kerberos

When you use a Kerberized application to connect to a remote host, the application, the KDC, and the remote host perform a set of rapid negotiations. When these negotiations are completed and the application has proven your identity on your behalf to the remote host, then the remote host grants you access.

You can log in remotely by using `ssh` or `ftp`.

- After installation, the `ssh` command is the only network service in Oracle Solaris, including Kerberos, that accepts network requests. See the `ssh`(1) man page.
- The `OPTIONS` section in the `ftp`(1) man page describes the Kerberos features in the FTP application. Your administrator must configure access to FTP.

♦ ♦ ♦ CHAPTER 6

Using Simple Authentication and Security Layer

This chapter includes information about the Simple Authentication and Security Layer (SASL).

About SASL

The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. An application calls the SASL library, `/usr/lib/libsasl.so`, which provides a glue layer between the application and the various SASL mechanisms. The mechanisms are used in the authentication process and in providing optional security services. The version of SASL is derived from the Cyrus SASL with a few changes.

SASL provides the following services:

- Loading of any plugins
- Determining the necessary security options from the application to aid in the choice of a security mechanism
- Listing of plugins that are available to the application
- Choosing the best mechanism from a list of available mechanisms for a particular authentication attempt
- Routing the authentication data between the application and the chosen mechanism
- Providing information about the SASL negotiation back to the application

SASL Reference

The following section provides information about the implementation of SASL.

SASL Plugins

SASL plugins provide support for security mechanisms, user-canonicalization, and auxiliary property retrieval. By default, the dynamically loaded 32-bit plugins are installed in `/usr/lib/sasl`, and the 64-bit plugins are installed in `/usr/lib/sasl/`*$ISA*. The following security mechanism plugins are provided:

`crammd5.so.1`	CRAM-MD5, which supports authentication only, no authorization
`digestmd5.so.1`	DIGEST-MD5, which supports authentication, integrity, and privacy, as well as authorization
`gssapi.so.1`	GSSAPI, which supports authentication, integrity, and privacy, as well as authorization. The GSSAPI security mechanism requires a functioning Kerberos infrastructure.
`plain.so.1`	PLAIN, which supports authentication and authorization.

In addition, the EXTERNAL security mechanism plugin and the INTERNAL user canonicalization plugins are built into `libsasl.so.1`. The EXTERNAL mechanism supports authentication and authorization. The mechanism supports integrity and privacy if the external security source provides it. The INTERNAL plugin adds the realm name if necessary to the username.

The Oracle Solaris release is not supplying any `auxprop` plugins at this time. For the CRAM-MD5 and DIGEST-MD5 mechanism plugins to be fully operational on the server side, the user must provide an `auxprop` plugin to retrieve clear text passwords. The PLAIN plugin requires additional support to verify the password. The support for password verification can be one of the following: a callback to the server application, an `auxprop` plugin, `saslauthd`, or `pwcheck`. The `saslauthd` and `pwcheck` daemons are not provided in the Oracle Solaris releases. For better interoperability, restrict server applications to those mechanisms that are fully operational by using the `mech_list` SASL option.

SASL Environment Variable

By default, the client authentication name is set to `getenv("LOGNAME")`. This variable can be reset by the client or by the plugin.

SASL Options

The behavior of `libsasl` and the plugins can be modified on the server side by using options that can be set in the `/etc/sasl/`*app*`.conf` file. The variable *app* is the server-defined name for the application. The documentation for the server *app* should specify the application name.

The following options are supported:

`auto_transition`	Automatically transitions the user to other mechanisms when the user does a successful plain text authentication.
`auxprop_login`	Lists the name of auxiliary property plugins to use.
`canon_user_plugin`	Selects the canon_user plugin to use.
`mech_list`	Lists the mechanisms that are allowed to be used by the server application.
`pwcheck_method`	Lists the mechanisms used to verify passwords. Currently, `auxprop` is the only allowed value.
`reauth_timeout`	Sets the length of time, in minutes, that authentication information is cached for a fast reauthentication. This option is used by the DIGEST-MD5 plugin. Setting this option to 0 disables reauthentication.

The following options are not supported:

`plugin_list`	Lists available mechanisms. Not used because the option changes the behavior of the dynamic loading of plugins.
`saslauthd_path`	Defines the location of the saslauthd door, which is used for communicating with the `saslauthd` daemon. The `saslauthd` daemon is not included in the Oracle Solaris release. So, this option is also not included.
`keytab`	Defines the location of the `keytab` file used by the GSSAPI plugin. Use the `KRB5_KTNAME` environment variable instead to set the default `keytab` location.

The following options are options not found in Cyrus SASL. However, they have been added for the Oracle Solaris release:

`use_authid`	Acquires the client credentials rather than use the default credentials when creating the GSS client security context. By default, the default client Kerberos identity is used.
`log_level`	Sets the desired level of logging for a server.

♦ ♦ ♦ CHAPTER 7

Using Smart Cards for Multifactor Authentication in Oracle Solaris

This chapter provides information about how to use smart cards as a proof of identity when authenticating to an Oracle Solaris 11.3 system. The system cannot be running Sun Ray Software (SRS).

Note - For systems that are running SRS, follow the integrated Sun Ray smart card authentication procedures.

This chapter covers the following topics:

Two-Factor Authentication and Smart Cards

Smart cards provide a second proof of identity when logging in to sensitive computers and web sites.

About Two-Factor Authentication

Two-factor authentication (2FA) adds an independent authentication step, which strengthens login verification. 2FA is a form of Multi-Factor Authentication (MFA) in which a user must present several separate pieces of evidence to an authentication mechanism before the user can

gain access to a computer system. In 2FA, the user must supply two of the following categories of MFA evidence:

- Something you know, for example a personal identification number (PIN) or password
- Something you possess, for example a smart card, challenge response key fob, or token generator
- Something inherent to your body, for example a biometric fingerprint, retina scan, or voice print

In addition to the two proofs of identity, 2FA typically requires users to confirm that they themselves are the persons trying to log in to the account. An Oracle Solaris computer server enforcing 2FA would require two separate proofs of identity, a smart card (something you possess) and a PIN (something you know).

Smart cards, also called common access cards (CAC), are plastic cards with an embedded microchip that can provide personal identification, authentication, data storage, and application processing. In addition, they can enable the encryption and cryptographic signing of email and use of public key infrastructure (PKI) authentication tools. Logging in with a smart card in Oracle Solaris provides much stronger security than network login processes that depend on traditional passwords only.

U.S. Government Smart Cards

In this guide, a CAC is a U.S. Department of Defense (DoD) smart card that is used for 2FA. CACs are issued as standard identification for cleared government employees. Government employees use their CAC to access government buildings and computer networks. The CAC contains cardholder information, including a PKI certificate. Software in a smart card reader through standard Internet protocols can compare the cardholder information with data on a government server and either grant or deny access.

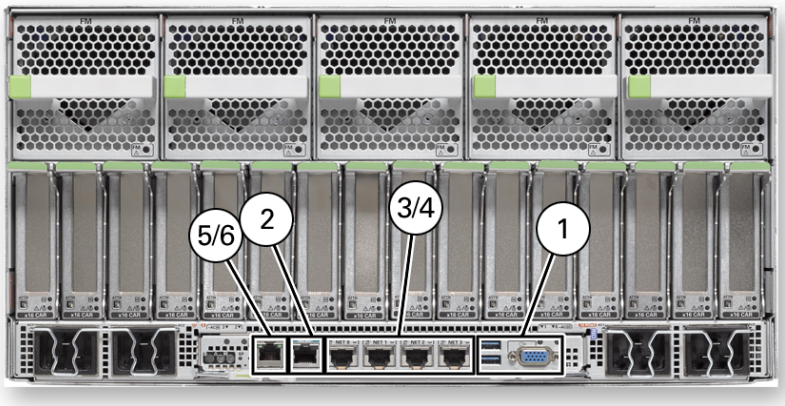
Oracle Solaris recognizes the four kinds of DoD CAC cards for computer authentication:

- Geneva Conventions Identification Card – For active duty/reserve armed forces and uniform service members
- Geneva Convention Accompany Forces Card – For emergency-essential civilian personnel
- ID and Privilege Common Access Card – For civilians residing on military installations
- ID card for DoD/Government Agency identification – For civilian employees and contractors

Local, Remote, and ILOM Smart Card Logins

The following figure illustrates the entry points for smart card logins.

FIGURE 7 Smart Card Entry Points



- 1 – Smart card reader directly attached to the system. Monitors and keyboards also use this entry point. See Figure 8, “Local Login With a Smart Card,” on page 116.
- 2 – Smart card reader directly attached to the system through a serial port. Consoles and terminal programs also use serial ports. See Figure 8, “Local Login With a Smart Card,” on page 116.
- 3 – Remote network access to smart card by using Secure Shell. See Figure 9, “Remote Login Over a Network With a Smart Card,” on page 117.
- 4 – Remote network access to smart card by using an X11 desktop. See Figure 9, “Remote Login Over a Network With a Smart Card,” on page 117.
- 5/6 – Integrated Lights Out Management (ILOM) port connects to smart card by using Secure Shell or `https`. See Figure 10, “ILOM Login With a Smart Card,” on page 117.

Smart cards and smart card readers in Oracle Solaris provide 2FA user authentication and nonrepudiation for three types of login: local login, remote login over the network, and remote login using Oracle Integrated Lights Out Manager (ILOM). After configuring their smart card login and authenticating to the server, users can also use secure web communication and secure email by configuring their web browser and mailer. For details, see “Enabling Your Web Browser and Email to Use Your Smart Card” on page 142.

The following figures illustrate 2FA logins that use a smart card.

FIGURE 8 Local Login With a Smart Card

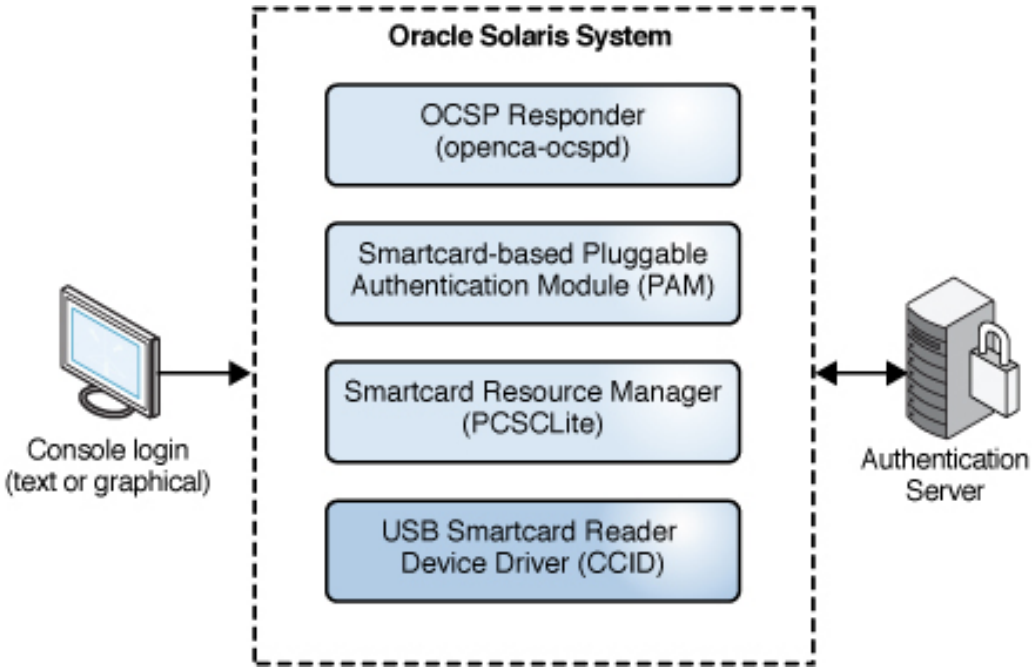


FIGURE 9 Remote Login Over a Network With a Smart Card

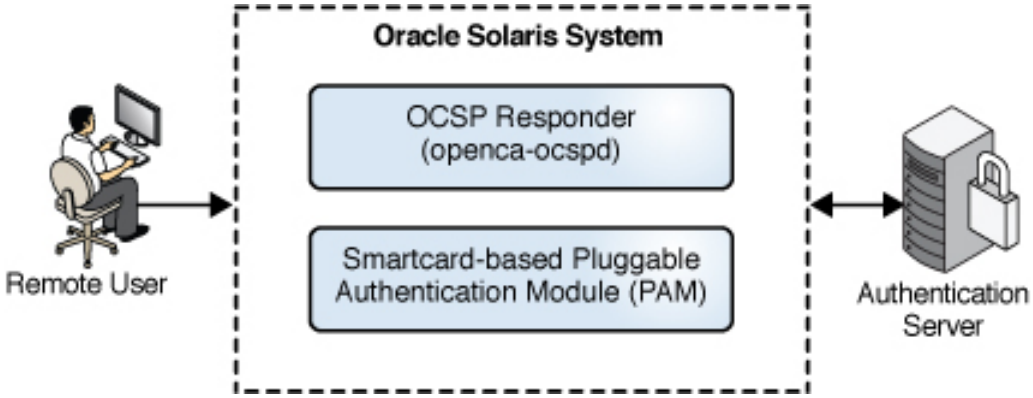
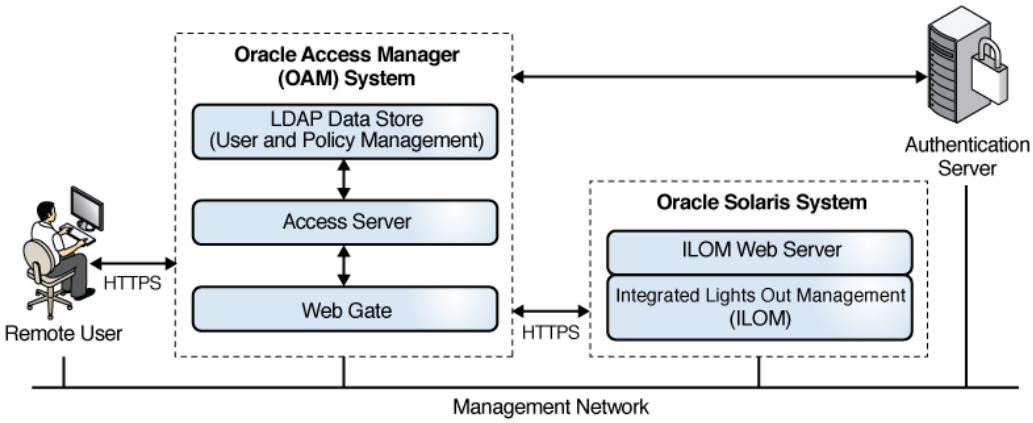


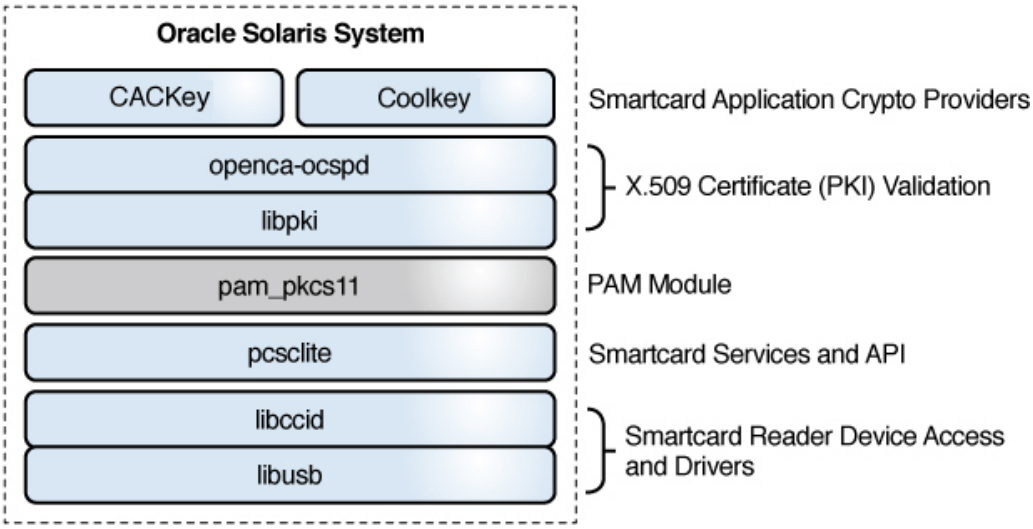
FIGURE 10 ILOM Login With a Smart Card



Implementation of Two-Factor Authentication in Oracle Solaris

Oracle Solaris implements 2FA with smart cards by using the following software stack. Most of the software is available in the `smartcard` package. The CACKey crypto provider is in a separate package. None of the IPS package groups install smart card packages, so you must install them.

FIGURE 11 Software Implementation of Two-Factor Authentication in Oracle Solaris



The 2FA software stack consists of the following modules:

- `libusb` – Open source library that enables access to USB devices. See `https://sourceforge.net/projects/libusb/files/libusb-1.0/`
- libccid – Open source library for generic USB CCID (Chip/Smart Card Interface Devices) driver and ICCD (Integrated Circuit Card Devices).
- libpki – Open source library that manages certificates from generation to validation.

  For documentation, see libpki Documentation.
- pcsclite – Provides the `pcscd` daemon as an SMF service that responds to requests to load drivers and handles runtime programs that are linked to the `libpcsclite.so` client library.

The `libpcsclite.so` client library connects a smart card driver to authentication software, in concert with the `pam_pkcs11` module.

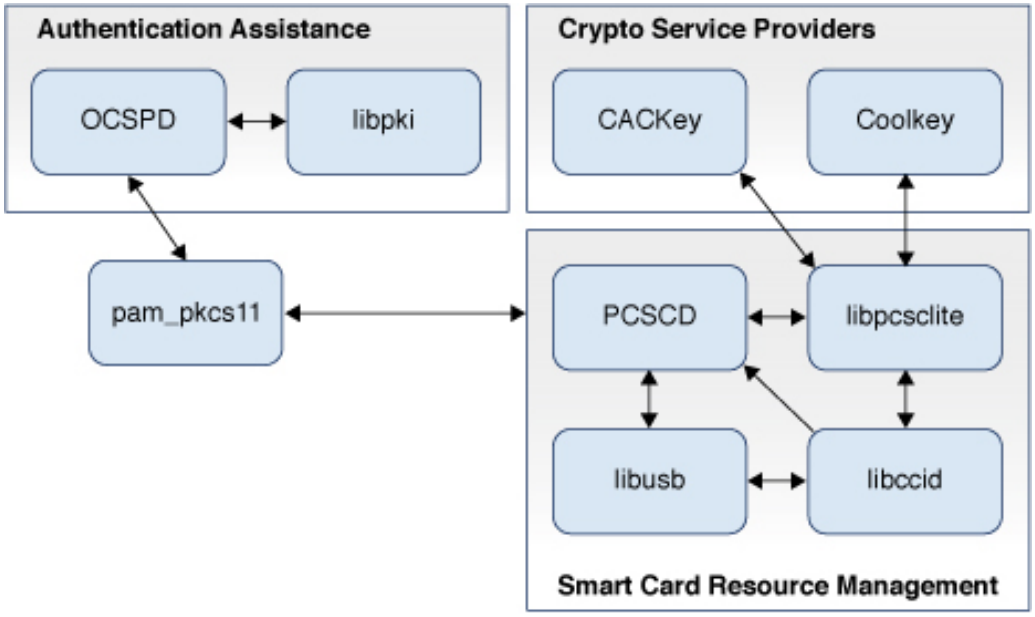
- For support of DoD CAC-enabled applications and web sites, a 2FA implementation requires CACKey software to link to the PKCS #11 module and web browser plugin.
- For support of PIV card-enabled applications and web sites, a 2FA implementation requires Coolkey software to link to the PKCS #11 module and web browser plugin.

- openca-ocspd – Open source Online Certificate Status Protocol (OCSP) responder

  OCSP is an Internet protocol for verifying whether an X.509 digital certificate is still valid. OCSP messages are encoded in ASN.1 and are usually communicated over HTTP. An OCSP server responds to requests for certificate verification, therefore are called OCSP responders.
- `pam_pkcs11(5)` – Pluggable authentication module (PAM) for the PKCS #11 token libraries that are used to authenticate users to an Oracle Solaris system.

The following figure illustrates the module connections for smart cards.

FIGURE 12 Software Connections for Two-Factor Authentication in Oracle Solaris



Software Cryptographic Providers for Smart Cards

As illustrated in Figure 11, “Software Implementation of Two-Factor Authentication in Oracle Solaris,” on page 118 and Figure 12, “Software Connections for Two-Factor Authentication in Oracle Solaris,” on page 119, Oracle Solaris supports smart card cryptography from two providers:

- Coolkey – Available from the `smartcard` package.

  Coolkey dynamically detects the presence of tokens when the `pcsclite` daemon (`pcscd`) is managing one or more PKI hardware token device interfaces, such as a smart card reader or other CCID supported devices.
- CACKey – Available from DISA for users with a security clearance and a Controlled Access Card. This software is also available from the `solaris` publisher.

  CACKey provides a standard PKCS #11 interface for smart cards that are connected to a PC/SC compliant reader. CACKey performs a similar function to Coolkey, but supports only U.S. Government smart cards that implement the Government Smart Card Interoperability Specification (GSC-IS) v2.1 or newer. To view the specification, go to NIST Computer Security Division web site and search the page for "6887". For a list of the cards, see “U.S. Government Smart Cards” on page 114.

Hardware Readers for Smart Cards

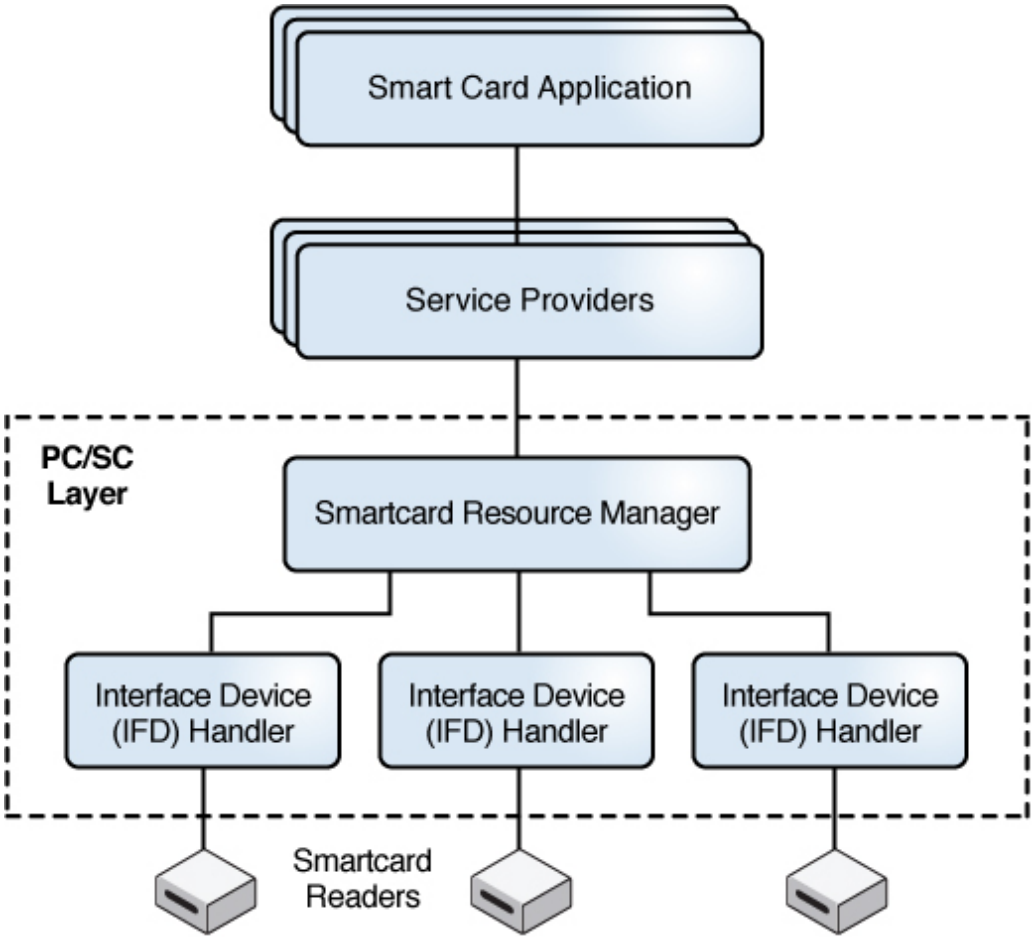
The following smart card hardware readers can be attached to an Oracle Solaris system for a local login with a smart card to authenticate to the system:

- HID/Omnikey, 3121
- Identive (formerly SCM Microsystems), SCR-3310v2 and SCR-3310
- ActivCard / ActivIdentity V3

Smart Card Architecture in Oracle Solaris

The following figure illustrates how Oracle Solaris connects to locally attached smart card readers and makes the resources on those smart cards available to cryptographic service providers and smart card-enabled applications.

FIGURE 13 PC/SC Layer Connecting Drivers to the Smart Card

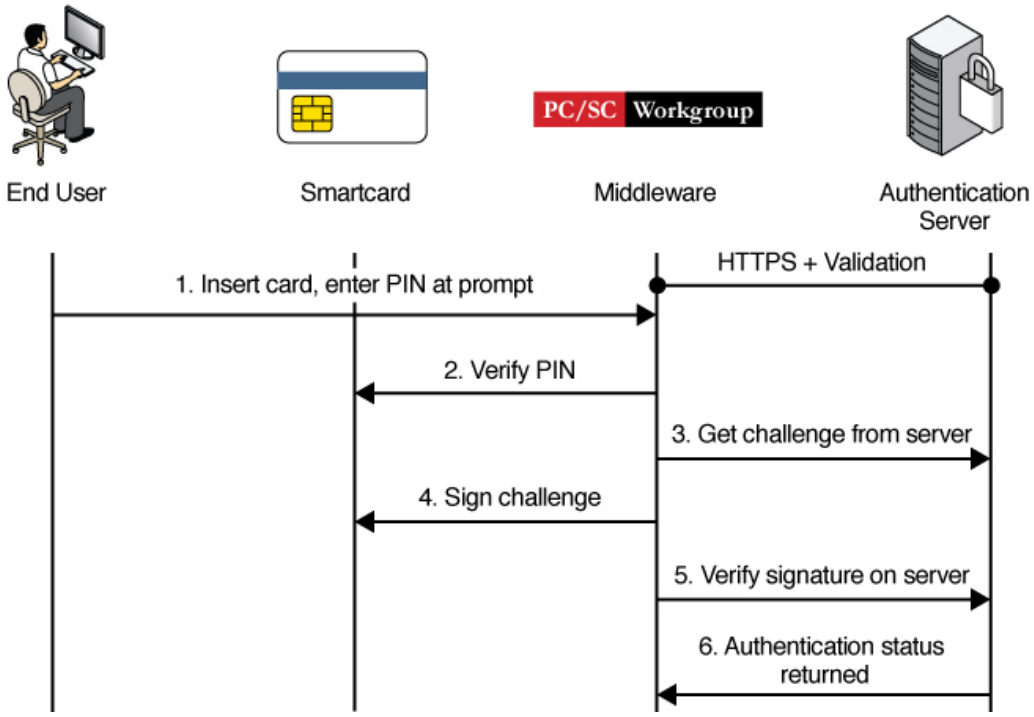


A smart card reader connects and communicates with a smart card on an Oracle Solaris system by using the PC/SC industry standard for accessing smart cards. The `pam_pkcs11` module integrates with the software in the `smartcard` package to provide 2FA authentication. Then, the OCSP responder communicates with an existing smart card Certificate Authentication (CA) server infrastructure to authenticate and verify the user-entered smart card PIN and verify the X.509 certificate that resides on the smart card.

Note - If you do not use OCSP, you can use local files to verify certificates and users.

The following figure illustrates how Oracle Solaris handles PKI authentication of a smart card.

FIGURE 14 PKI Authentication by Smart Card



Configuring an Oracle Solaris System for Smart Card Login

With appropriate hardware and software installed, an Oracle Solaris system can use smart cards to authenticate users. Figure 14, “PKI Authentication by Smart Card,” on page 122 shows how the PKI authenticates the smart card, and Figure 11, “Software Implementation of Two-Factor Authentication in Oracle Solaris,” on page 118 and Figure 12, “Software Connections for Two-Factor Authentication in Oracle Solaris,” on page 119 shows the software stack. The

assumption is that the site is already running an LDAP directory service where certificates can be enrolled.

Caution - These procedures do not work with Sun Rays. For systems that are running SRS, follow the integrated Sun Ray smart card authentication procedures.

Main Smart Card Configuration Tasks

The main configuration tasks are to install the `smartcard` package that contains the software, configure PAM, connect the LDAP server to the software, add and configure the software providers, register the smart cards and test. The steps in order are as follows:

1. Install the `smartcard` package. If you are a U.S. Government organization, also install the `pkcs11_cackey` package – "Installing Smart Card Packages" on page 124.
2. Run the OpenSSH version of Secure Shell on the smart card server and the smart card clients – "How to Use the OpenSSH Implementation of Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.3*.
3. (Optional) Review the `pcsclite` interface for communicating with smart cards – "Using `pcsclite` for Smart Cards" on page 124.
4. (Optional) Configure the `ccid` XML file if default configuration is insufficient – "Configuring `libccid` for Smart Card Readers" on page 125.
5. Configure a local or remote desktop for smart card users – "Configuring a Desktop for Users With Smart Cards" on page 126.
6. Create certificates and configure the OCSP responder for certificate validation "Configuring OCSP Certificates for Smart Cards" on page 128.

   OCSP relies on the `libpki` module from Oracle Solaris to manage the PKI certificates that OCSP validates. You can skip this step if you are storing certificates and using CRLs locally.
7. Configure `pam_pkcs11.conf` to obtain X.509 certificate information from a smart card – "How to Display a Smart Card's X.509 Certificate" on page 133.
8. Configure PAM to use the `pam_pkcs11` module – "How to Configure PAM for 2FA With Smart Cards" on page 136.
9. Register user smart cards and the root certificate with the `ssh` client – "How to Configure the Secure Shell Client for Smart Cards" on page 140.

Installing Smart Card Packages

The `smartcard` package installs all the necessary software to use the Coolkey cryptographic provider with smart cards on your system. The `pkcs11_cackey` package installs the CACKey cryptographic provider, which the U.S. Government requires for DoD access.

▼ How to Install the Smart Card Packages

Before You Begin You must become an administrator who is assigned the Software Installation rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Determine if the `smartcard` package is installed.**
   The following command output indicates that the package is not installed on your system.

   ```
   $ pkg info smartcard
   pkg: info: no packages matching the following patterns you specified are
   installed on the system.  Try querying remotely instead:

           smartcard
   ```

2. **Install the package.**

   ```
   $ pfbash pkg install smartcard
   ```

3. **If you belong to an organization that uses CACKey, install the `pkcs11_cackey` package.**

   ```
   $ pkg install pkcs11_cackey
   ```

Using `pcsclite` for Smart Cards

PC/SC Lite provides a Windows SCard interface for communicating with smart cards and readers. It uses the same `winscard` API as the Windows SCard. PC/SC Lite also uses the `libccid` library as its CCID-compliant smartcard reader driver. The default configuration of the `pcsclite` library (PC/SC Lite) is typically sufficient for smart card authentication.

In Oracle Solaris, the library and daemon are in the `pkg://library/security/pcsc/pcsclite` package. The version of the library is in the `Version` field of the package information:

```
$ pkg info pcsclite
```

```
            Name: library/security/pcsc/pcsclite
         Summary: Provides smart card services using the SCard API (PC/SC)
...
         Version: version
```

Oracle Solaris uses both the PC/SC Lite client (`libpcsclite.so`) and the PC/SC Lite server daemon (`pcscd`) for smart card authentication. PC/SC Lite is a service in Oracle Solaris, so you administer `pcscd` by using the `svcadm` command.

After installation of the `smartcard` group package, the PC/SC daemon remains disabled. After configuring the supporting software for smart cards, you will manually enable the `pcsc` service in "Enabling an Oracle Solaris System for Smart Card Login" on page 141.

Configuring `libccid` for Smart Card Readers

The default configuration of `libccid` is typically sufficient for smart card authentication. You might change the configuration during debugging.

The version of the library is in the `Version` field of the package information:

```
$ pkg info ccid
            Name: library/security/pcsc-lite/ccid
         Summary: Provides smart card reader drivers for pcsclite (PC/SC)
...
         Version: version
```

You can set the debug level and change the voltage level in `Info.plist`, the configuration file for the CCID driver. By default, it is installed in the `/usr/lib/$ISA/pcsc/drivers/ifd-ccid.bundle/Contents` directory.

The CCID driver uses the `pcsclite` debug function. Debug output goes to stdout or syslog depending on how you configure `pcsclite` debugging.

▼ How to Configure and Debug `libccid`

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. Set a useful debug level.

The debug level is set in the `ifdLogLevel` field. It is a binary `OR` combination of four different levels:

- 1 – Critical: Important error messages

- 2 – Info: Informative messages such as which reader was detected
- 4 – Comm: A dump of all the bytes exchanged between the host and the reader
- 8 – Periodic: A periodic log (every 1/10 of a second) of activity during a `pcscd` test if a card is present

By default the debug level is set to 3 (1 + 2) and corresponds to the critical and info levels.

2. **Set the voltage level to the tolerance of your smart card reader.**

   Modify the `ifdDriverOptions` field.

   The voltage level is a binary OR combination of 4 different levels.

   - 0 – Power on the card at 5V (default value)
   - 16 – Power on the card at 3V, and if 3V fails then power on the card at 5V
   - 32 – Power on the card at 1.8V, then 3V, and then 5V
   - 48 – Let the smart card reader decide the voltage level

   By default the voltage level is set to 0 and corresponds to 5V.

Caution - If your smart card reader requires low voltage, the reader will burn out and destroy smart cards if you do not lower the voltage.

3. **Restart the driver to read the modified `Info.plist` settings.**

   You have two options:

   - **Unplug all CCID readers, which unloads the CCID driver. Then, plug them in again.**
   - **Or, you can restart the `pcsc` service daemon.**

     ```
     # svcadm restart pcsc
     ```

Next Steps You can also configure version numbers and USB device numbers in this file.

Configuring a Desktop for Users With Smart Cards

Users who need to gain system desktop access must have a local or a remote X11 desktop configured for them by an administrator. For a local desktop, the administrator must install the `solaris-desktop` group package. For remote desktop access, the administrator must configure XDMCP.

How to Configure a Local Desktop

Before You Begin You must become an administrator who is assigned the Software Installation rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Verify that the `solaris-desktop` group package is installed.**

```
$ pkg search -H -o pkg.name solaris-desktop
group/system/solaris-desktop
```

2. **If the `solaris-desktop` group package is not installed, install it.**

```
$ pfexec pkg install solaris-desktop
```

This system can now be used by a smart card user who needs a desktop.

How to Configure a Remote X11 Desktop

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Verify that the `solaris-desktop` group package is installed.**

```
# pkg search -H -o pkg.name solaris-desktop
group/system/solaris-desktop
```

2. **Enable XDMCP connections in `gdm`.**

Set `Enable` to `true` in the `/etc/gdm/custom.conf` file.

```
[xdmcp] Enable=true
```

For more information, see the `gdm`(1M) man page.

3. **Before activating the changes, warn users to log off and save their work.**

The restart kills all current `gdm` sessions.

4. **Activate the configuration change by restarting the `gdm` desktop windowing service.**

```
# svcadm restart gdm
```

Users can now access a remote desktop with their smart cards. For the user procedure, see "How to Use a Smart Card to `ssh` to a Remote GNOME Desktop" on page 148.

Configuring OCSP Certificates for Smart Cards

The certificates on the smart card are used to for the second authentication factor. The smart card is "something you have" that contains certificates that have been verified by the root CA.

Note - You can skip this task if you are storing certificates and using CRLs locally.

▼ How to Configure and Validate Certificates

This procedure shows how to configure a root certificate for smart card authentication and test that the `ocspd` daemon can verify the status of the certificate found on a smart card. You will need two terminal windows, one window where you configure the Certificate Authority (CA) and another window where you test `ocspd` verification.

Before You Begin In Oracle Solaris, the `libpki` library is already linked against the OpenSSL preferred cryptographic provider and the OpenLDAP libraries. The `openca-ocspd` responder uses `libpki` to manage the PKI certificates from generation to validation.

You have assumed the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Set up a test Certificate Authority (CA).**

   For example, the following commands create a local CA:

   ```
   # cd /root
   # mkdir CertAuth
   # cd CertAuth
   # mkdir certs private
   # chmod g-rwx,o-rwx private
   # echo '01' > serial
   # touch index.txt
   ```

2. **Configure the `openssl.conf` file to point to this CA.**

   For example, the following `openssl.conf` file points to the `root` CA.

   ```
   # cat << 'EOF' > openssl.conf
   [ ca ]
   default_ca              = CertAuth

   [ CertAuth ]
   dir                     = /root/CertAuth
   certificate             = $dir/cacert.pem
   ```

```
database                = $dir/index.txt
new_certs_dir           = $dir/certs
private_key             = $dir/private/cakey.pem
serial                  = $dir/serial

default_crl_days        = 7
default_days            = 365
default_md              = sha256

policy                  = CertAuth_policy
x509_extensions         = certificate_extensions
copy_extensions         = copy

[ CertAuth_policy ]
commonName              = supplied
stateOrProvinceName     = optional
countryName             = optional
emailAddress            = optional
organizationName        = optional
organizationalUnitName  = optional

[ certificate_extensions ]
basicConstraints        = CA:false
extendedKeyUsage        = OCSPSigning

[ req ]
default_bits            = 2048
default_keyfile         = /root/CertAuth/private/cakey.pem
default_md              = sha256

prompt                  = no
distinguished_name      = root_ca_distinguished_name

x509_extensions         = root_ca_extensions

[ root_ca_distinguished_name ]
commonName              = CertAuth

[ root_ca_extensions ]
basicConstraints        = CA:true
EOF
```

3. **Declare the `OPENSSL_CONF` environment variable in your existing shell.**

Note - The CA must be accessible from your network.

```
# export OPENSSL_CONF=/root/CertAuth/openssl.conf
```

4. **Create a CA key and a CA certificate.**

```
# openssl genrsa -out private/cakey.pem 2048
Generating RSA private key, 2048 bit long modulus
...+++
.............................+++
e is 65537 (0x10001)

# openssl req -new -x509 -days 999 -key private/cakey.pem -out cacert.pem
```

5. **Create a test certificate signing request (CSR) in a new terminal.**

```
# unset OPENSSL_CONF
# cd /root
# mkdir test_client
# cd test_client

# openssl genrsa -out testkey.pem 2048

# openssl req -new -key testkey.pem -out testreq.pem
  Country Name (2 letter code) []:
  State or Province Name (full name) []:
  Locality Name (eg, city) []:
  Organization Name (eg, company) []:
  Organizational Unit Name (eg, section) []:
  Common Name (e.g. server FQDN or YOUR name) []:test
  Email Address []:
  A challenge password []:
  An optional company name []:
```

6. **Create a test certificate and verify that you can revoke it.**

   Change the terminal configuration to the CA.

```
# export OPENSSL_CONF=/root/CertAuth/openssl.conf

# cd /root/CertAuth

# openssl ca -in /root/test_client/testreq.pem
  Sign the certificate? [y/n]:y
  1 out of 1 certificate requests certified, commit? [y/n]y

# openssl verify -CAfile cacert.pem certs/01.pem

# cp certs/01.pem /root/test_client/testcert.pem

# openssl ca -revoke /root/test_client/testcert.pem

# openssl ca -gencrl -out crl.pem
```

7. Create a key and a certificate signing request for the `ocspd` daemon in a new terminal.

```
# unset OPENSSL_CONF

# cd /etc/ocspd

# ocspd-genreq.sh
  Please Enter the Server's Subject (eg., CN=OCSP Server, O=OpenCA, C=US):[Enter]
  Please Enter the Algorithm (default: RSA-SHA256):[Enter]
  Please Enter the Key Size (default: 2048):[Enter]

# cp /etc/ocspd/req.pem /root/CertAuth/ocspdreq.pem
# chmod a+r /root/CertAuth/ocspdreq.pem
```

Note - Use a password when prompted if you want the server key to be encrypted.

8. Create a certificate for `ocspd`.

You will copy the pem files to `/etc/ocspd`.

```
# export OPENSSL_CONF=/root/CertAuth/openssl.conf
# cd /root/CertAuth
# openssl ca -in ocspdreq.pem
  Sign the certificate? [y/n]:y
  1 out of 1 certificate requests certified, commit? [y/n]y

# openssl verify -CAfile cacert.pem certs/02.pem

# cp /root/CertAuth/certs/02.pem /etc/ocspd/certs/cert.pem
# cp /root/CertAuth/cacert.pem /etc/ocspd/certs
# cp /root/CertAuth/crl.pem /etc/ocspd/crls
```

9. Enable or restart the `ocsp` service.

By default, the `ocspd` daemon is disabled after installation of the `smartcard` group package. To use an OCSPD responder with smart card authentication in Oracle Solaris, the service must be online.

- **If the service is disabled, enable it.**

```
# svcs ocsp
STATE          STIME    FMRI
disabled       14:21:16 svc:/application/security/ocsp:default

# svcadm enable ocsp
```

- **If the service is already enabled, you must restart it.**

```
# svcs ocsp
STATE          STIME    FMRI
enabled       14:21:16 svc:/application/security/ocsp:default

# svcadm restart ocsp
```

10. **Verify that the `ocspd` daemon is running as `daemon`.**

```
# svcs ocsp
STATE          STIME    FMRI
online         14:27:13 svc:/application/security/ocsp:default

# ps -ef |grep ocspd
  daemon 22814   1   0 14:27:14 ?   0:00 /usr/lib/ocspd -c /etc/ocspd/ocspd.xml -d
```

11. **As a regular user, check the certificate revocation status.**

```
$ openssl ocsp -issuer /etc/ocspd/certs/cacert.pem \
   -CAfile /etc/ocspd/certs/cacert.pem -url http://localhost:2560/ -serial 1
Response verify OK
1: revoked
        This Update: Jun 12 21:03:32 2016 GMT
        Next Update: Jun 12 21:08:32 2016 GMT
        Revocation Time: Jun 12 20:49:22 2016 GMT

$ openssl ocsp -issuer /etc/ocspd/certs/cacert.pem \
  -CAfile /etc/ocspd/certs/cacert.pem -url http://localhost:2560/ -serial 2
Response verify OK
2: good
        This Update: Jun 12 21:03:54 2016 GMT
        Next Update: Jun 12 21:08:54 2016 GMT
```

Configuring PAM for Smart Cards

The `pam_pkcs11` login module enables X.509 certificate-based user authentication, the certificate that resides on the CACKey and Coolkey smart cards. The module uses the name service switch (NSS) to manage and validate PKCS #11 smart cards either from locally accessible certificate revocation lists (CRLs) or from the Online Certificate Status Protocol (OCSP).

All Oracle Solaris logins go through PAM. To enable smart card authentication for a user, you add information from the user's smart card to PAM files.

In the `/etc/security/pam_pkcs11` directory, you create or modify the following files:

- `pam_pkcs11.conf` – Identifies the CACKey or Coolkey cryptographic module, contains some information from the smart card, and points to mapping files
- `subject_mapping` – Maps the `subject` on a smart card's X.509 certificate to the card's login user or to an additional role that the user can assume, such as `root`
- `cn_map` – Maps the smart card's X.509 certificate name (CN) to the login user's CN or to the CN of an additional role that the login user can assume, such as `root`

Then, the `auth` PAM stack for all logins is modified to require a second authentication step. This second step uses the PKCS #11 library to verify the X.509 certificate on the smart card and requires the user to supply the smart card PIN.

▼ How to Display a Smart Card's X.509 Certificate

In this procedure, you configure `pam_pkcs11` to recognize a smart card that uses either CACKey or Coolkey as its cryptographic module. This configuration includes support for smart card authentication to Secure Shell.

After this preparation, you use this information to configure the user's smart card access in.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

You have completed "How to Use the OpenSSH Implementation of Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.3* and are running the OpenSSH version of Secure Shell. A smart card reader with a user's smart card in it is attached to your Oracle Solaris system. The system has the `pcsclite` and `ccid` packages installed.

1. **If you have not yet enabled the `pcsc` service, enable it.**

   ```
   # svcadm enable pcsc
   ```

   This service starts the `pcscd` daemon, which the `pam_pkcs11` module uses to communicate with the smart card.

2. **Copy the `pam_pkcs11.conf` file to `pam_pkcs11.conf.orig`.**

   ```
   # cd /etc/security/pam_pkcs11
   # cp pam_pkcs11.conf pam_pkcs11.conf.orig
   ```

3. **Configure PAM to use the CACKey or Coolkey cryptographic module for the smart card.**

   Add the appropriate module to the `pam_pkcs11.conf` file.

- **In the `pam_pkcs11.conf` file, find and change the `use_pkcs11_module` definition to CACKey.**

  ```
  # pfedit /etc/security/pam_pkcs11/pam_pkcs11.conf
  use_pkcs11_module = cackey;
  ```

  Following that line, add support for CACKey.

  ```
  # CACKey support
  pkcs11_module cackey {
       module = /usr/lib/$ISA/libcackey.so;
       description = "CACKey";
       slot_num = 0;
       support_threads = false;
       ca_dir =/etc/security/pam_pkcs11/cacerts;
       crl_dir =/etc/security/pam_pkcs11/crls;
       cert_policy = none;
       crl_policy = none;
    }
  ```

- **In the `pam_pkcs11.conf` file, find and change the `use_pkcs11_module` definition to Coolkey.**

  ```
  # pfedit /etc/security/pam_pkcs11/pam_pkcs11.conf
  use_pkcs11_module = coolkey;
  ```

  Following that line, add support for Coolkey.

  ```
  # Coolkey support
  pkcs11_module coolkey {
       module = /usr/lib/$ISA/libcoolkeypk11.so;
       description = "Coolkey";
       slot_num = 0;
       support_threads = false;
       ca_dir =/etc/security/pam_pkcs11/cacerts;
       crl_dir =/etc/security/pam_pkcs11/crls;
       cert_policy = none;
       crl_policy = none;
    }
  ```

4. **Still in the `pam_pkcs11.conf` file, find and change the `use_mappers` definition.**

   This entry indicates the certificate parameters that can verify the certificate.

   ```
   use_mappers = cn, subject, openssh, null;
   ```

   A full list of supported mappers is in the `pam_pkcs11.conf` file.

5. **Find and change the `mapper subject` definition.**

```
# Certificate Subject to login based mapper
# provided file stores one or more "Subject -> login" lines
mapper subject {
     debug = false;
     module = internal;
     ignorecase = false;
     mapfile = file:///etc/security/pam_pkcs11/subject_mapping;
}
```

You will create the `subject_mapping` mapfile in "How to Configure PAM for 2FA With Smart Cards" on page 136.

6. **Find and change the `mapper cn` definition.**

```
mapper cn {
       debug = true;
       module = internal;
       ignorecase = true;
       mapfile = file:///etc/security/pam_pkcs11/cn_map;
  }
```

You will create the `cn_map` mapfile in "How to Configure PAM for 2FA With Smart Cards" on page 136.

7. **Find and change the `mapper openssh` definition.**

```
# Search public keys from user's $HOME/.ssh/authorized_keys for match
mapper openssh {
     debug = false;
     module = /usr/lib/pam_pkcs11/$ISA/openssh_mapper.so;
}
```

8. **Exit the `pam_pkcs11.conf` file.**

9. **Set restrictive permissions on the `pam_pkcs11.conf` file.**

```
# chmod 644 pam_pkcs11.conf
```

10. **Verify that you can view the information on the user's smart card.**

    a. **In a terminal window, run the `pkcs11_inspect` command.**

    ```
    # /usr/lib/pam_pkcs11/pkcs11_inspect
    ```

    b. **Type the user's smart card PIN at the prompt.**

After you type the PIN, X.509 certificate information from the user's smart card should appear. For sample output, see Step 1 in "How to Configure PAM for 2FA With Smart Cards" on page 136.

Next Steps Continue with "How to Configure PAM for 2FA With Smart Cards" on page 136 to complete PAM configuration for smart card authentication.

▼ How to Configure PAM for 2FA With Smart Cards

This procedure shows how to complete the configuration of the `pam_pkcs11` module to authenticate smart card users. The example in the procedure is of U.S. Government-issued CACKeys. You must follow these steps for every smart card user.

Before You Begin You have completed "How to Display a Smart Card's X.509 Certificate" on page 133.

You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. Display the information on the user's smart card.

```
# /usr/lib/pam_pkcs11/pkcs11_inspect
```

After you type the PIN, X.509 certificate information from the user's smart card should appear similar to the following:

```
PIN for token:
Printing data for mapper cn:
LNAME.FNAME.ID
Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID
Printing data for mapper openssh:
ssh-rsa AAAAB3NzaC1yc2EAAAA ...
... fname.lname@example.org
Printing data for mapper cn:
LNAME.FNAME.ID
Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID
Printing data for mapper openssh:
ssh-rsa AAAAB3NzaC1yc2EAAAA ...
... fname.lname@example.org
...
Printing data for mapper cn:
DoD Root CA ...
...
Printing data for mapper subject:
```

```
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD Root CA

Printing data for mapper cn:
DOD CA-30

Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-30
...
```

2. Create the `subject_mapping` file.

Copy the file from `/etc/security/pam_pkcs11/subject_mapping.example`.

```
# cd /etc/security/pam_pkcs11
# cp subject_mapping.example subject_mapping
```

3. In the `subject_mapping` file, map the user's `subject` value from their X.509 certificate to their login name.

The `format` line describes the mapping format.

Use the value from the line that follows the first instance of `Printing data for mapper subject:`, for example:

```
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID
```

```
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID -> login
...
## Root certificate authority
...
```

Smart cards that are not issued by the U.S. government have different values for certificate subjects.

Note - If the `root` account has a unique smart card, treat `root` as a user and map `root`'s certificate name from the X.509 certificate to the `root` login name.

---

4. (Optional) If the smart card user is allowed to assume other roles, such as `root`, map the correct certificate from the card to the correct identity.

- In this example, the certificate for the `root` CA is the certificate for the `root` role.

```
# pfedit subject_mapping
# Mapping file for Certificate Subject
```

```
# format: Certificate Subject -> login
#
## User certificates
...
## Certificate name mapped to the root account
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-3 -> root
```

- In this example, the certificate for the `root` role is different from the root CA certificate.

```
# pfedit subject_mapping
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
...
## Certificate name mapped to the root account
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-30 -> root
```

- In this example, the `DOD CA-29` certificate subject maps to the `sysadmin` role.

```
# pfedit subject_mapping
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
...
## Certificate name mapped to the sysadmin role
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-29 -> sysadmin
```

Smart cards that are not issued by the U.S. government have different values for certificate names.

5. Map `CN` values in the `cn_map` file.

- Create the `/etc/security/pam_pkcs11/cn_map` file.
- Map the user's certificate name from the X.509 certificate to the user's login name.
- If the user can assume a role, map the appropriate certificate name to the role.

```
# pfedit cn_map
# Mapping file for Certificate Name
# format: Certificate Name -> login
#
## User certificate names
LNAME.FNAME.ID -> login
… many user entries

## Certificate name mapped to the root account
```

```
DOD CA-3 -> root
```

6. **Set restrictive permissions on the mapping files.**

```
# chmod 644 cn_map subject_mapping
```

7. **Add `pam_pkcs11` as the first module in the `auth` stack of the `login` PAM configuration file.**

```
# cd /etc/pam.d
# cp login login.orig
# pfedit login
# login service (explicit because of pam_dial_auth)
#
## pam_pkcs11 enables smart card logins
auth sufficient         pam_pkcs11.so
auth definitive         pam_user_policy.so.1
...
```

8. **Modify the `other` file similarly.**

```
# cp other other.orig
# pfedit other
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
## pam_pkcs11 enables smart card logins
auth sufficient         pam_pkcs11.so
auth definitive         pam_user_policy.so.1
...
```

9. **Test the PAM configuration.**

   a. **Log in to a local desktop.**

   b. **Log in to a remote desktop.**

   c. **Log in by using the `ssh` command.**

   d. **Log in by using a local console.**

For more information about PAM and testing, see Chapter 1, “Using Pluggable Authentication Modules” in *Managing Kerberos and Other Authentication Services in Oracle Solaris 11.3*.

Configuring Secure Shell Clients for Smart Cards

The smart cards must contact the Secure Shell server for certificate validation. Secure Shell is based on OpenSSH and provides the necessary PKCS #11 support for clients, so requires no additional configuration for smart cards. The Secure Shell server can be running any OpenSSH version. See "How to Use the OpenSSH Implementation of Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.3*.

From a Secure Shell client, the private key on the user's smart card authenticates to a remote Secure Shell server by using key-based authentication. The user must configure the keys.

▼ How to Configure the Secure Shell Client for Smart Cards

In this procedure, you, the smart card user, obtain the public key from your smart card, use that key to identify the card to Secure Shell, then configure Secure Shell to recognize it.

Before You Begin A smart card reader with your smart card in it is attached to your Oracle Solaris system. The system has the `pcsclite` and `ccid` packages installed and the `pcscd` daemon enabled.

1. **Obtain the public key from your smart card.**
   For the procedure, see "How to Display a Smart Card's X.509 Certificate" on page 133.

2. **Add the public key portion to the `.ssh` directory in your home directory.**

   a. **Create an `authorized_keys` file in your `$HOME/.ssh` directory.**

   ```
   $ cd ; mkdir .ssh ; chmod 755 .ssh
   $ cd .ssh ; touch authorized_keys
   ```

   b. **Copy the public key information from the preceding output into the `authorized_keys` file.**

   Append the first public key in the output into the `authorized_keys` file, as in:

   ```
   Printing data for mapper openssh:
   ssh-rsa AAAAB3NzaC1yc2EAAAA ...
   ... fname.lname@example.org
   ```

   The key starts with `ssh-`*key-signing-algorithm* and ends with your email address. Do not introduce spaces when copying and pasting it.

   c. **Verify that the permissions on the `authorized_keys` file are 600.**

```
$ chmod 600 authorized_keys
```

3. **Test your access.**

   From a PC or workstation that has a CCID-compliant smart card reader attached, type `ssh` to connect to the smart card server.

   ```
   $ ssh username@SSH-server
   ```

   You are authenticated by your X.509 certificate-based CAC or smart card and PIN.

   Note - At the `ssh` login prompt, the system should now require your PIN instead of your password, because the system is now authenticating from the X.509 certificate information on the smart card.

4. **Log out immediately after the test.**

   Your Secure Shell connection is a secure trusted link into the server. To prevent a possible attack from their local PC or workstation, users must log out of the server or remove their smart card or CAC when not actively working.

Enabling an Oracle Solaris System for Smart Card Login

Configuration is the time-consuming part of enabling a system for smart card use. After completing the configuration tasks in "Configuring an Oracle Solaris System for Smart Card Login" on page 122, you enable smart card use by enabling the `svc://system/security/pcsc` smart card service.

▼ How to Enable Smart Card Authentication

Before You Begin You have assumed the `root` role. You have completed the tasks in "Configuring an Oracle Solaris System for Smart Card Login" on page 122. For a review, see "Main Smart Card Configuration Tasks" on page 123.

1. **On the smart card server, import the root CA certificates for OCSP.**

   Note - You can skip this step if you are storing certificates and using CRLs locally.

You configured and tested the certificates in "How to Configure and Validate Certificates" on page 128.

```
# pktool import "Root CA certificates" -i CACert.pem
```

where *CACert.pem* is the base-64 format root CA certificate file.

2. **Enable the smart card utility.**

```
# svcadm enable pcsc
```

Enabling Your Web Browser and Email to Use Your Smart Card

Users can access a secure browser and email by using the appropriate certificate from the smart card. The certificate relies on a hierarchy of certificates. Typically, the top level certificate signs the intermediate certificate and the intermediate certificate signs the site's certificate. By importing and trusting the top level certificate, you do not have to install the others.

The certificate authenticates to the web sites using SSL/PKI, and signs and encrypts emails. To authenticate to your Firefox web browser and Thunderbird email with your smart card, you configure the browser and email to read your client certificates from your CAC card.

▼ How to Download Smart Card Certificates for Web and Email Use

In this procedure, you download certificates that authenticate you to applications that require a smart card for access. You need the entire chain or hierarchy of certificates. To use the certificates, continue with "How to Configure Firefox to Use Your Smart Card for Authentication" on page 143 and "How to Configure Thunderbird to Use Your Smart Card for Signing and Encrypting Emails" on page 144.

This example uses U.S. Government certificates. Other organizations should follow the directions of their security administrator to locate and install the certificates.

1. **Open a web browser.**

2. **Go to DoD Cyber Exchange NIPR (cyber.mil).**

3. **In the PKI and PKE Tools section, naviagate to the PKI CA Certificate Bundles: PKCS#7 section.**

   Click to download the For DoD PKI Only - Version *n.n* URL.

4. **Unzip the files and follow the directions in `README.txt`.**

How to Configure Firefox to Use Your Smart Card for Authentication

In this procedure, you configure Firefox to authenticate with sites that require smart card authentication.

Note - Smart card software works with the 32-bit Firefox browser. It does not work with the 64-bit browser.

Before You Begin You must have completed "How to Download Smart Card Certificates for Web and Email Use" on page 142.

1. **Download the certificates for this application only.**

2. **Insert your CAC smart card.**

   The green light should flash.

3. **Add the CAC module to Firefox as a security device.**

   a. **From the Firefox Preferences Menu, navigate to the Advanced Section, click the Security Devices button, then the Load button.**

   b. **Type "CAC Module" as the module name and browse to the appropriate CAC module.**

      - For CACKey, select `/usr/lib/libcackey.so`.
      - For Coolkey, select `/usr/lib/libcoolkeypk11.so`.

   Firefox can now read the client certificates from your CAC card.

4. **In Firefox, test your configuration by navigating to a CAC-enabled website.**

   If you are prompted to enter your PIN and the site reports "Your PKI Certificate has been detected", the configuration is correct.

▼ How to Configure Thunderbird to Use Your Smart Card for Signing and Encrypting Emails

In this procedure, you configure Thunderbird to encrypt and sign emails with the certificate from your smart card.

Before You Begin You have completed "How to Download Smart Card Certificates for Web and Email Use" on page 142.

1. **Download the certificates for this application only.**
2. **Insert your CAC smart card.**
   The green light should flash.
3. **Add the CAC module to Thunderbird as a security device.**
   a. **From the Thunderbird Preferences Menu, navigate to the Advanced Section, Certificates tab, click the Security Devices button, then the Load button.**
   b. **Type "CAC Module" as the module name and browse to the appropriate CAC module.**
      - For CACKey, select `/usr/lib/libcackey.so`.
      - For Coolkey, select `/usr/lib/libcoolkeypk11.so`.

   Thunderbird can now read the client certificates from your CAC card.
4. **In Thunderbird, test your configuration by sending an email to yourself.**
   When composing the email message, pull down the Security menu and choose "S/MIME Sign" and "S/MIME Encrypt" before sending the message.

Using a Smart Card

Caution - These procedures do not work with Sun Rays. For systems that are running SRS, follow the integrated Sun Ray smart card authentication procedures.

Smart cards use personal identification numbers (PINs) rather than passwords. The smart card is protected from misuse by the PIN, which is known only to the smart card's owner. To use the smart card, you insert the card in a smart card reader that is attached to a computer and, when

prompted, type the PIN. The smart card can be used only by someone who possesses the smart card and knows the PIN.

For computer use, a CAC, PIV or X.509 certificate-based smart card should remain in the reader for the duration of the session. When the smart card is removed from the reader, the credentials are unavailable in the existing login session to any applications that require re-authentication.

Caution - Log out during periods of inactivity. An authenticated smart card is a secure trusted link into the server. To prevent a possible attack from your local system, you must log out or remove your smart card or CAC when not actively working.

The following tasks describe how to use the various entry points to a system to log in to Oracle Solaris with a smart card.

- "How to Log In From a Local Console With Smart Card Authentication" on page 145
- "How to Log In as a Role With Smart Card Authentication" on page 146
- "How to Log In Remotely by Using `ssh` With Smart Card Authentication" on page 147
- "How to Use a Smart Card to `ssh` to a Remote GNOME Desktop" on page 148
- "How to Use a Smart Card to Log In to Your Local GNOME Desktop" on page 149
- "How to Authenticate With a Smart Card on a Screensaver" on page 153

▼ How to Log In From a Local Console With Smart Card Authentication

Before You Begin You have inserted a smart card into the CCID-compliant smart card reader that is attached to a PC or workstation.

1. At the prompt, type your username.

```
smartcard console login: username
Smartcard authentication starts
Smart card found.
Welcome username!
```

2. Type the smart card PIN.

```
Smart card PIN: nnnnnnnn
verifying certificate
Last login: ...
Oracle Corporation   SunOS 5.11
You have new mail.
```

```
username@server:~$
```

3. **If you typed the wrong PIN, type the PIN again.**

```
Error 2320: Wrong smartcard PIN
date smartcard login: open_pkcs11_login() failed x0000000A0
Login incorrect
Smartcard authentication starts
Smart card found.
Please insert your smart card or enter your username.
Smartcard console login: nnnnnnnn
verifying certificate
Oracle Corporation   SunOS 5.11
You have new mail.
username@server:~$
```

4. **(Optional) To assume a role, `su` to the role.**

   For example, assume the `root` role.

```
$ su -
```

   The terminal displays the progress of smart card authentication.

```
Smartcard authentication starts
Smart card found.
Welcome root!
```

5. **Type the smart card PIN.**

```
Smart card PIN: nnnnnnnn
verifying certificate
Oracle Corporation   SunOS 5.11
You have new mail.
root@server: ~#
```

   If you typed the correct PIN, you are logged in.

6. **Type `Control-D` to log out of the session.**

How to Log In as a Role With Smart Card Authentication

Before You Begin You are logged in to a PC or workstation that has a CCID-compliant smart card reader attached. The `root` account has its own certificate. In Step 3 in "How to Configure PAM for 2FA With Smart Cards" on page 136, you started to locate and map certificates for privileged users.

1. **Open a terminal window.**

2. **Switch user to the role.**
   In this example, you switch to the `root` role.

   ```
   $ su -
   ```

   The terminal displays the progress of smart card authentication.

   ```
   Smartcard authentication starts
   Smart card found.
   Welcome root!
   ```

3. **Type the smart card PIN.**
   If you typed the correct PIN, a series of "`verifying certificate`" messages display, and you are logged in.

   ```
   Smart card PIN: nnnnnnnn
   verifying certificate
   verifying certificate
   ...
   Oracle Corporation   SunOS 5.11
   You have new mail.
   root@server: ~#
   ```

4. **If you typed the wrong PIN, you are logged in as your username, not the role.**

   ```
   Error 2320: Wrong smartcard PIN
   su: Authentication failed
   username@server: ~$
   ```

   To switch to the role, repeat the procedure.

5. **Type `exit` to log out of the session.**

How to Log In Remotely by Using `ssh` With Smart Card Authentication

Before You Begin You have a Secure Shell server that is configured for smart cards. You have inserted a smart card into the CCID-compliant smart card reader that is attached to a PC or workstation.

1. **Open a terminal window.**

2. **Use `ssh` to reach the remote server.**

- **If you are connecting from an Oracle Solaris Secure Shell client to an Oracle Solaris Secure Shell server, type the following command:**

  ```
  $ ssh username@SSH-server
  ```

- **If either the Secure Shell client or server are not Oracle Solaris, provide the full path to the PKCS #11 library.**

  Confirm with your administrator which cryptographic module to use.

  ```
  $ ssh -I /usr/lib/libcackey.so username@SSH-server
  $ ssh -I /usr/lib/libcoolkeypk11.so username@SSH-server
  ```

The terminal displays the progress of smart card authentication.

```
Smartcard authentication starts
Smart card found.
Welcome LNAME.FNAME.ID!
```

3. **Type your smart card PIN.**

   ```
   Smart card PIN: nnnnnnnn
   ```

   - If you typed the correct PIN, a "`verifying certificate`" message displays and you are logged in.
   - If you typed the wrong PIN, an error message displays: `Error 2320: Wrong smartcard PIN`. Type the correct PIN when smart card authentication restarts.

4. **To quit the `ssh` session, type `Control-D`.**

How to Use a Smart Card to `ssh` to a Remote GNOME Desktop

Before You Begin Your administrator has completed "How to Configure a Remote X11 Desktop" on page 127.

- **You can log in from the command line or from a client application that calls the XDMCP chooser.**

  - **Use the `-X` option to the `ssh` command.**

    ```
    $ ssh -X GNOME-desktop-server
    ```

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

- **Call the XDMCP chooser from a client application for remote X11 desktop display.**

  Possible client applications include Hummingbird.

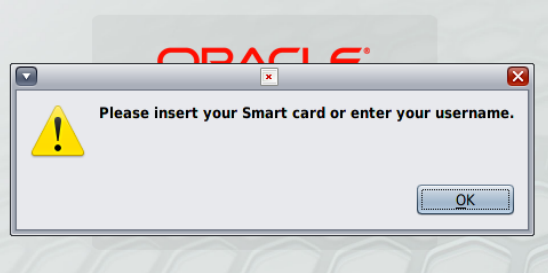
▼ How to Use a Smart Card to Log In to Your Local GNOME Desktop

Before You Begin Your administrator has completed "How to Configure a Local Desktop" on page 127. You have inserted a smart card into the CCID-compliant smart card reader that is attached to a PC or workstation.

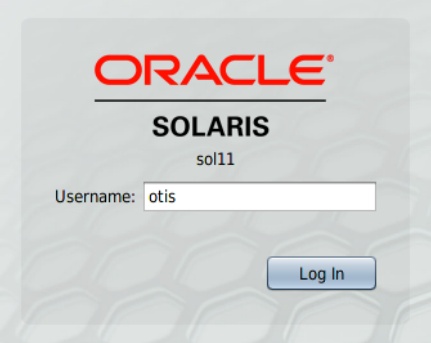
1. **Log in.**

   You are presented with two smart card dialog boxes that you must confirm by pressing the `OK` button.

2. **Press the OK key, then either insert your smart card or type your username.**



3. **Press the OK key and type your username.**

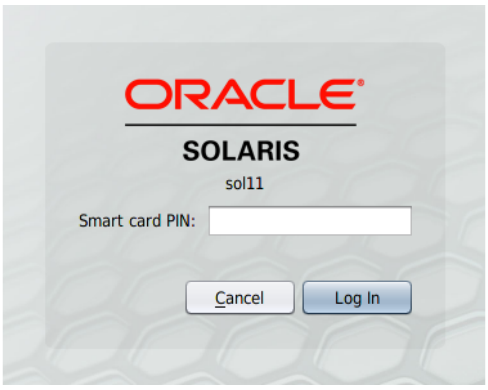


The system looks for the smart card and displays "`Smart card found.`".

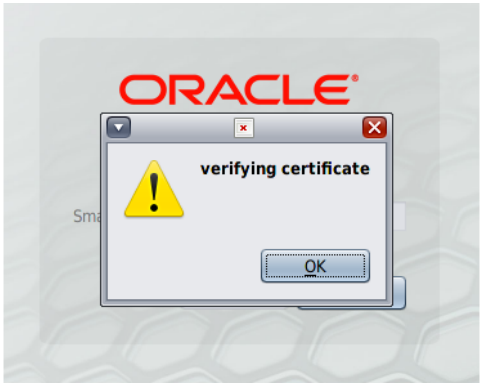
4. **Press the `OK` key.**

   Then press it again on the Welcome dialog box.

5. **Type your smart card PIN at the prompt and press the Log In button.**



After the system verifies the certificate, it logs you in.



6. **If any of the preceding steps fail, take appropriate action.**

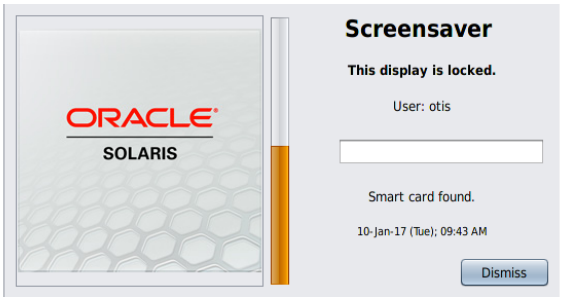
The system displays an error dialog box that describes the error.

- `Error 2312: open PKCS#11 session failed` – The system cannot find a smart card or the `subject_mapping` ID on the card did not match your username. Contact your administrator.
- `Error 2320: Wrong smartcard PIN` – You typed an incorrect PIN. Press the `OK` button and type the PIN again.

How to Authenticate With a Smart Card on a Screensaver

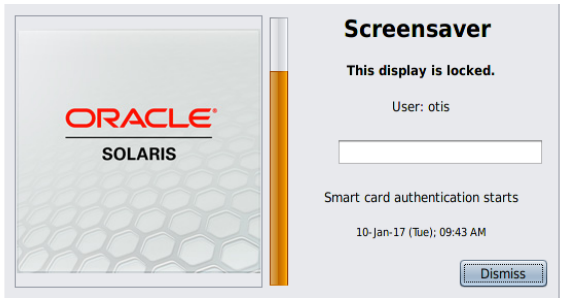
Before You Begin The GNOME screensaver is displaying on a desktop that you logged in to with a smart card PIN.

1. **Insert the cursor into the entry field and type your smart card PIN.**



Then press the `Enter` or `Return` key.

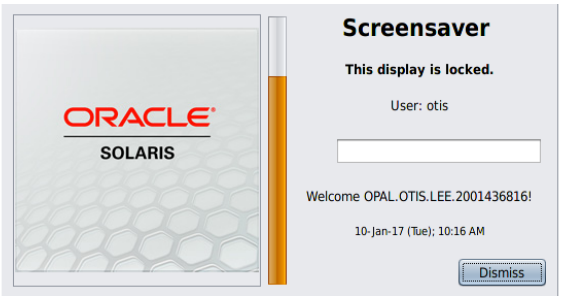
Note - Do not press the `Dismiss` button.



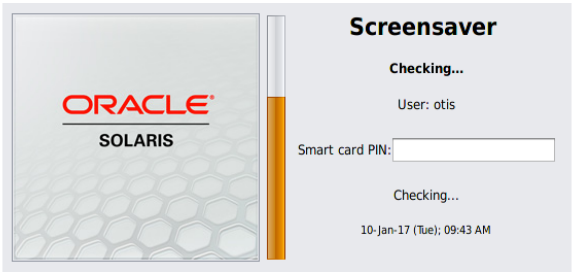
Note - Press the `Dismiss` button only if you need a different account to unlock the screen. Then you would insert the card for that different account, type the username in the entry field and press the `Enter` or `Return` key. The system will attempt to read the card for the certificate and prompt for the smart card PIN for this different account.

The smart card reader's green LED blinks as the card is being read and the `subject_mapping` field of the certificate is verified.

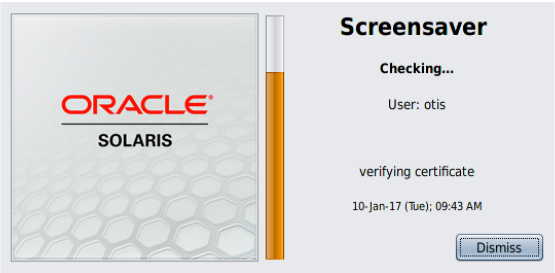
2. **After the dialog box displays a welcome message, press the `Dismiss` button.**



3. **At the smart card PIN prompt, type your PIN and press the `Enter` or `Return` key.**



A "`verifying certificate`" dialog box displays.



If the dialog box displays `Error 2320: Wrong smartcard PIN`, press the `Dismiss` button and type the PIN again.

4. **Press the `Dismiss` button to log in.**

♦ ♦ ♦ CHAPTER 8

Using One-Time Passwords for Multifactor Authentication in Oracle Solaris

This chapter provides information about how to configure Oracle Solaris to require selected users to use the one-time password (OTP) from their mobile authenticator when logging in to an Oracle Solaris system. This chapter covers the following topics:

About OTP in Oracle Solaris

OTP provides a second proof of identity when logging in to Oracle Solaris. The use of a second proof of identity is called two-factor authentication (2FA). The system first prompts you for your UNIX password, then for the OTP from your mobile authenticator app. After you the system verifies these two authentications, it logs you in. For more information, see "About Two-Factor Authentication" on page 113.

OTP in Oracle Solaris conforms to the specifications for HMAC-based and time-based OTPs in HOTP: An HMAC-Based One-Time Password Algorithm, RFC 4226 and TOTP: Time-Based One-Time Password Algorithm, RFC 6238, so should be able to work with any authenticator that conforms to these specifications.

Oracle Solaris delivers OTP in the `system/security/otp` IPS package. The `solaris-small-server`, `solaris-large-server`, and `solaris-desktop` groups deliver this package, which contains the following items:

- OTP PAM module – `pam_otp_auth` implements OTP. When `pam_otp_auth` is a module in a login PAM stack, users must provide an OTP. For more information, see the `pam_otp_auth`(5) man page.

- OTP PAM policy configuration files – `otp` and `otp_strict` are two per-user PAM configuration files that the OTP package provides.
- OTP administrative command – `otpadm` is the command you use to configure OTP authentication for users. Users can manage their own keys with this command. For more information, see the `otpadm`(1M) man page.
- OTP rights profile – OTP Auth Manage All Users rights profile enables administrators to administer OTP through the `otpadm` command.

To assign OTP to individual users, administrators use the `-K pam_policy=otp` option to the `useradd` or `usermod` command. For the procedures, see "Configuring and Using OTP in Oracle Solaris" on page 158.

OTP Administration in Oracle Solaris

Administrators with the OTP Auth Manage All Users rights profile can use the `otpadm` command to manage the attributes that affect users' OTPs. Regular users can manage the keys for their own account, but an administrator must install the OTP package and assign an `otp` PAM policy to the user.

To configure per-user OTP, you must be assigned the User Management rights profile.

The administrator is responsible for configuring any OTP attributes that should not inherit the default values. Users who set their own secret keys can also set their OTP attributes.

Note - The mobile authenticator app and the `otpadm` utility in Oracle Solaris constrain the attributes that a user can customize. For example, a user's OTP code must be at least 6 digits long.

Typically, the default attribute values are adequate. For a description of the attributes, see the `otpadm`(1M) man page.

Configuring and Using OTP in Oracle Solaris

Configuring OTP requires coordination between the administrator and OTP users. After configuration, OTP users can log in to the server by using their UNIX login and the OTP that is displayed on their mobile authenticator app.

Caution - Users can be locked out if the secret key is not on their mobile device before they are required to use OTP.

Administrator responsibilities:

1. Ensure that the `otp` package is installed on the login server.
2. Ensure that the login server can keep accurate time.

   The server should be a client of a Precision Time Protocol (PTP) or Network Time Protocol (NTP) clock synchronization service. For more information, see *Enhancing System Performance Using Clock Synchronization and Web Caching in Oracle Solaris 11.3*.
3. Ensure that the user has a secret key. Either the user or you can create the user's secret key.
4. Assign the `otp` per-user PAM policy to the user.

Responsibilities of the user with the mobile authentication app:

1. Download a mobile authenticator app to their mobile device.
2. Create a secret key or coordinate with the administrator when the administrator creates a secret key for their authenticator app.
3. Ensure that OTP configuration on the authenticator app matches the configuration on the login server.
4. Type the secret key into their mobile authenticator app before the administrator assigns the `otp` PAM policy to them.
5. Test that a prompt appears for a OTP, and that the OTP logs them in.

TABLE 5 Task Map: Using OTP in Oracle Solaris

Task	Description	For Instructions
Prepare for OTP.	The administrator ensures that the login server has the modules that support OTP.	"How to Configure OTP" on page 160
Create and confirm a secret key.	The user creates a secret key on the login server and configures the mobile authenticator with the secret.	"How to Configure and Confirm the Secret Key for Your OTP" on page 160
Create and send a secret key to the user.	Alternatively, the administrator creates a secret key for the user.	"How to Set a Secret Key for a OTP User" on page 162
Enable OTP.	The administrator requires a OTP at login from users.	"How to Require a UNIX Password and a OTP to Log In to an Oracle Solaris System" on page 163

How to Configure OTP

Before You Begin You must become an administrator with the Software Installation rights profile to install the `otp` package. You must become an administrator with the OTP Auth Manage All Users rights profile to enable you to manage OTP. The `root` role has all of these rights. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Ensure that the login server is a client of a clock synchronization service.**
   For instructions, see *Enhancing System Performance Using Clock Synchronization and Web Caching in Oracle Solaris 11.3*.

2. **Ensure that the `otp` package is installed on the login server.**

   ```
   login-server $ pkg search -H -o pkg.name -l \<otp\>
   pkg:/system/security/otp
   ```

3. **(Optional) Determine whether the defaults are sufficient for your site policy.**

   ```
   $ otpadm get
                   mode=timer
              algorithm=hmac-sha1
                 digits=6
                   ...
   ```

4. **If users should use different values, send them instructions.**

   - Include the attributes that the users must set on their mobile devices.
   - If users set their own secrets, send them instructions. See Example 12, "Users Changing to a Longer OTP and a Stronger Algorithm," on page 161 and Example 13, "Setting and Displaying a Hexadecimal Secret Key," on page 162.

How to Configure and Confirm the Secret Key for Your OTP

Before You Begin You are the owner of a mobile device that connects to the Internet, and your administrator has completed "How to Configure OTP" on page 160.

1. **Download a mobile authenticator app to your device and open the application.**
   Search for Authenticator in your app store. See Exploring Oracle Mobile Authenticator and Its Applications for more information.

2. **On the login server, create a secret key.**

```
$ otpadm set secret
XXXX nnnn XXXX XXXX nnnn nnnn nnnn XXXX
Enter current code from authenticator: nnnnnnnn
```

The server displays the secret and prompts for a code from the mobile application.

3. **Type the displayed secret into the mobile authenticator.**

For example, on the Oracle Mobile Authenticator screen, press the plus (+) button in the upper right corner of the screen, choose "Enter provided key", pick a name for the account (*username@login-server*), and type the secret key under "Key".

4. **Generate a code on the mobile application and type it into the `otpadm` prompt.**

After the `otpadm` prompt accepts a valid code from the authenticator, OTP is configured and ready to use.

5. **Test the OTP.**

After the administrator completes "How to Require a UNIX Password and a OTP to Log In to an Oracle Solaris System" on page 163, log in to the server. You should be prompted first for your server login, then for the OTP. After you type the OTP, you should be logged in.

Example 12 Users Changing to a Longer OTP and a Stronger Algorithm

The administrator notifies OTP users to change to a SHA2 algorithm and an 8-digit password.

1. By email, the administrator instructs them to follow the new guidelines.

```
Users,
We are changing the mobile authenticator to use a longer password and
a stronger algorithm.  Please complete the changeover by Friday.
On the server, open a terminal window and issue the following commands:
otpadm set algorithm=hmac-sha256 digits=8 secret
Respond to the prompts and instructions.
If you have difficulty, notify the administrator.
```

2. In their mobile authenticator app, users select the `hmac-sha256` algorithm and set digits to `8`.
3. On the login server, each user runs the commands from the email.

```
$ otpadm set algorithm=hmac-sha256 digits=8 secret
1234 abcd 1234 abcd 1234 1234 1234 abcd
Enter current code from authenticator: nnnnnnnn
```

4. Each user types the secret into their mobile authenticator app.

5. After the user generates a code on the app and types it at the login server prompt, the app and the server are synchronized and configuration is complete.

Example 13 Setting and Displaying a Hexadecimal Secret Key

Users own a mobile authenticator that prompts for a secret in hexadecimal format. They create a secret key that displays in hexadecimal format.

```
$ otpadm -f hex set secret
 7DDF B236 7023 82A6 F70F 0001 C8B7 F0BE A76C 3F31
```

Troubleshooting If the OTP password fails, wait and try the second OTP that displays.

If the login server does not accept the OTP, verify with the administrator that the clocks on the mobile device and the server are synchronized.

If the times on the login server and the mobile authenticator do not synchronize, you and your administrator could configure a counter-based OTP rather than a time-based OTP. See Example 15, “Using a Counter Rather Than a Timer for OTP Authentication,” on page 165.

▼ How to Set a Secret Key for a OTP User

Before You Begin You have completed “How to Configure OTP” on page 160.

You must become an administrator with the OTP Auth Manage All Users rights profile. For more information, see “Using Your Assigned Administrative Rights” in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **(Optional) Determine if the defaults are sufficient for your site policy.**

```
$ otpadm get
                mode=timer
           algorithm=hmac-sha1
              digits=6
                ...
```

2. **Create a secret key for the user.**

```
$ pfexec otpadm -u username -f [base32 | hex] set attributes secret
```

For example, use the default OTP attributes:

```
$ pfexec otpadm -u jdoe set secret
```

For example, require a longer OTP:

```
$ pfexec otpadm -u jdoe set digits=8 secret
```

For example, set counter mode:

```
$ pfexec otpadm -u jdoe set mode=counter secret
```

By default, the OTP secret is displayed in Base32 format. Most authenticators accept this format, but some expect hexadecimal format. To change the format for the OTP secret, see Example 13, “Setting and Displaying a Hexadecimal Secret Key,” on page 162.

3. **Get the secret key to the user.**

   - **Send the secret to users out of band.**

     a. **Display the secret.**

        ```
        $ pfexec otpadm -u username get secret
        CBA6 5JBR M73T XGZK CNAB 36HG QLE5 PFCR
        ```

     b. **Send it over a secure channel, such as encrypted email.**

   - **Instruct the user to log in, display the secret key, and type it into their mobile authenticator app.**

     ```
     username $ otpadm get secret
     CBA6 5JBR M73T XGZK CNAB 36HG QLE5 PFCR
     ```

How to Require a UNIX Password and a OTP to Log In to an Oracle Solaris System

Before You Begin

You have completed “How to Configure OTP” on page 160.

You must become an administrator with the following rights profiles to complete the steps in this task:

- User Management rights profile – For assigning PAM policy to users
- OTP Auth Manage All Users rights profile – For managing OTP

The `root` role has all of these rights. For more information, see “Using Your Assigned Administrative Rights” in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Ensure that the user has typed and confirmed the secret key in their mobile authenticator app.**

   You and the user have finished the following tasks:

   - "How to Configure and Confirm the Secret Key for Your OTP" on page 160
   - "How to Set a Secret Key for a OTP User" on page 162

2. **As an administrator with the User Management rights profile, change the user's PAM policy to `otp`.**

   ```
   $ pfexec usermod -K pam_policy=otp username
   ```

3. **Instruct the OTP users to test their logins.**

   The users should be prompted first for their regular login password, then for the OTP.

4. **(Optional) To disable OTP for a user, perform two steps:**

   a. **As an administrator with the User Management rights profile, remove `otp` as the user's PAM policy.**

      ```
      $ pfexec usermod -K pam_policy=  username
      ```

   b. **As an administrator with the OTP Auth Manage All Users rights profile, remove the inactive OTP configuration files from the user's home directory.**

      ```
      $ pfexec otpadm -u username expunge
      ```

Example 14 Enforcing the Change to a Longer OTP and a Stronger Algorithm

The authenticator app in use at a company can handle a very strong algorithm and a long password. To implement a stronger security policy, the administrator notifies OTP users to change to a SHA2 algorithm and an 8-digit password. Then, the administrator audits their change. The email and user responsibilities are shown in Example 12, "Users Changing to a Longer OTP and a Stronger Algorithm," on page 161.

1. After allowing time for the users to change their OTP attributes, the administrator audits every OTP user.

   ```
   $ pfexec otpadm -u username get algorithm digits
                digits=8
           algorithm=hmac-sha256
   ```

2. If a user's configuration is different from the preceding output, the administrator sends a warning email that specifies the date that the user will be locked out.

3. On the specified date, the administrator locks out OTP users who have not changed to the new OTP configuration.

   ```
   $ pfexec usermod -e date username
   ```

Example 15 Using a Counter Rather Than a Timer for OTP Authentication

In this example, the user is using a mobile authenticator that supports counter mode. The administrator sets the OTP mode to `counter` when the mobile authenticator does not synchronize with the login server. To prevent using any existing codes, the administrator sets a new secret.

```
$ otpadm -u username set mode=counter secret
$ otpadm -u username get secret
VOCJ YHTV 2C4O DTDN R34X CGM4 YZVM JJFI
```

The administrator sends the secret to the user out of band. Before typing in the secret, the user sets the mobile authenticator app to use counter mode.

Troubleshooting If the authenticator does not confirm the user's OTP, users should wait and try the second OTP that displays.

If the login server does not accept the OTP, make sure that the clocks on the mobile device and the server are synchronized.

♦ ♦ ♦ CHAPTER 9

Configuring Network Services Authentication

This chapter provides information about how to use Secure RPC to authenticate a host and a user across an NFS mount, and covers the following topics:

About Secure RPC

Secure RPC (Remote Procedure Call) protects remote procedures with an authentication mechanism. The Diffie-Hellman authentication mechanism authenticates both the host and the user who is making a request for a service. The authentication mechanism uses Data Encryption Standard (DES) encryption. Applications that use Secure RPC include NFS and the NIS naming service.

NFS Services and Secure RPC

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

The NFS service uses Secure RPC to authenticate users who make requests over the network. This process is known as *Secure NFS*. The Diffie-Hellman authentication mechanism, `AUTH_DH`,

uses DES encryption to ensure authorized access. The `AUTH_DH` mechanism has also been called `AUTH_DES`. For more information, see the following:

- To set up and administer Secure NFS, see "Administering the Secure NFS System" in *Managing Network File Systems in Oracle Solaris 11.3*.

Kerberos Authentication

Kerberos is an authentication system that was developed at MIT. A client-side and server-side implementation of Kerberos V5, which uses RPCSEC_GSS, is included with this release. For more information, see "How to Configure Kerberos NFS Servers" on page 95.

DES Encryption With Secure NFS

The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt data. If two credential users or principals know the same DES key, they can communicate in private by using the key to encipher and decipher text. DES is a relatively fast encryption mechanism.

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

Diffie-Hellman Authentication and Secure RPC

The Diffie-Hellman (DH) method of authenticating a user is nontrivial for an intruder to crack. The client and the server have their own private key, which they use with the public key to devise a common key. The private key is also known as the *secret key*. The client and the server use the common key to communicate with each other. The common key is encrypted with an agreed-upon encryption function, such as DES.

Authentication is based on the ability of the sending system to use the common key to encrypt the current time. Then, the receiving system can decrypt and check against its current time. The time on the client and the server must be synchronized. The Network Time Protocol (NTP) can be used to synchronize clocks. NTP public domain software from the University of Delaware is included in the Oracle Solaris software. Documentation is available from the NTP Documentation web site.

The public keys and private keys are stored in an NIS database. NIS stores the keys in the `publickey` map. This file contains the public key and the private key for all potential users.

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

Administering Authentication With Secure RPC

By requiring authentication for use of mounted NFS file systems, you increase the security of your network.

The following task map points to procedures that configure Secure RPC for NIS, and NFS.

TABLE 6 Task Map: Administering Authentication With Secure RPC

Task	Description	For Instructions
1. Start the keyserver.	Ensures that keys can be created so that users can be authenticated.	"How to Restart the Secure RPC Keyserver" on page 169
2. Set up credentials on an NIS host.	Ensures that the `root` user on a host can be authenticated in an NIS environment.	"How to Set Up a Diffie-Hellman Key for an NIS Host" on page 170
3. Give an NIS user a key.	Enables a user to be authenticated in an NIS environment.	"How to Set Up a Diffie-Hellman Key for an NIS User" on page 171
4. Share NFS files with authentication.	Enables an NFS server to securely protect shared file systems using authentication.	"How to Share NFS Files With Diffie-Hellman Authentication" on page 172

▼ How to Restart the Secure RPC Keyserver

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Verify that the `keyserv` daemon is running.**

```
# svcs \*keyserv\*
STATE    STIME   FMRI
disabled Dec_14  svc:/network/rpc/keyserv
```

2. **Ensure that the `nobody` user cannot use default keys.**

```
$ pfedit /etc/default/keyserv
```

```
# Change the default value of ENABLE_NOBODY_KEYS to NO.
#
#ENABLE_NOBODY_KEYS=YES
ENABLE_NOBODY_KEYS=NO
```

3. **Enable the keyserver service if the service is not online.**

```
# svcadm enable network/rpc/keyserv
```

How to Set Up a Diffie-Hellman Key for an NIS Host

Perform this procedure on every host in the NIS domain.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **If the default naming service is not NIS, add the `publickey` map to the naming service.**

   a. **Verify that the value of `config/default` for the naming service is not `nis`.**

```
# 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 nis dns"
config/printer              astring      "user files nis"
```

   If the value of `config/default` is `nis`, you can stop here.

   b. **Set the naming service for `publickey` to `nis`.**

```
# svccfg -s name-service/switch setprop config/publickey = astring: "nis"
# svccfg -s name-service/switch:default refresh
```

   c. **Confirm the `publickey` value.**

```
# svccfg -s name-service/switch listprop
config                      application
config/value_authorization  astring      solaris.smf.value.name-service.switch
config/default              astring      files
config/host                 astring      "files nis dns"
```

```
config/printer              astring       "user files nis"
config/publickey            astring       nis
```

On this system, the value of `publickey` is listed because it differs from the default, `files`.

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

```
# newkey -h hostname
```

where *hostname* is the name of the client.

Example 16 Setting Up a New Key for `root` on an NIS Client

In the following example, and administrator with the Name Service Security rights profile sets up `earth` as a secure NIS client.

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

How to Set Up a Diffie-Hellman Key for an NIS User

Perform this procedure for every user in the NIS domain.

Before You Begin You must be logged in to the NIS master server to generate a new key for a user. You must be assigned the Name Service Security rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **Create a new key for a user.**

```
# newkey -u username
```

where *username* is the name of the user. The system prompts for a password. You can type a generic password. The private key is stored in an encrypted form by using the generic password.

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

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

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

Example 17 Setting Up and Encrypting a New User Key in NIS

In this example, superuser sets up the key.

```
# newkey -u jdoe
Adding new key for unix.12345@example.com
New Password: xxxxxxxx
Retype password: xxxxxxxx
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

Then the user `jdoe` re-encrypts the key with a private password.

```
% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@example.com
Please enter the Secure-RPC password for jdoe: xxxxxxxx
Please enter the login password for jdoe: xxxxxxxx
Sending key change request to centralexample...
```

How to Share NFS Files With Diffie-Hellman Authentication

This procedure protects shared file systems on an NFS server by requiring authentication for access.

Before You Begin Diffie-Hellman public key authentication must be enabled on the network. To enable authentication on the network, complete "How to Set Up a Diffie-Hellman Key for an NIS Host" on page 170.

You must become an administrator who is assigned the System Management rights profile to perform this task. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

1. **On the NFS server, share a file system with Diffie-Hellman authentication.**

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

where *filesystem* is the file system that is being shared.

The `-o sec=dh` option means that `AUTH_DH` authentication is now required to access the file system.

2. **On an NFS client, mount a file system with Diffie-Hellman authentication.**

```
# mount -F nfs -o sec=dh server:filesystem mount-point
```

server	Is the name of the system that is sharing *filesystem*
filesystem	Is the name of the file system that is being shared, such as `opt`
mount-point	Is the name of the mount point, such as `/opt`

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

Authentication Services Glossary

These glossary entries cover words that have different meanings in different parts of the operating system, or have different meanings in the authentication services of Oracle Solaris. For definitions of Kerberos components, see the documentation on the MIT Kerberos web site (`http://web.mit.edu/kerberos/`).

authorization 1. In Kerberos, the process of determining if a principal can use a service, which objects the principal is allowed to access, and the type of access that is allowed for each object.

2. In Oracle Solaris rights-based access control (RBAC), a right that can be assigned to a role or user (or as part of 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.

instance 1. In Kerberos, the second part of a principal name. An instance qualifies the principal's primary. In the case of a service principal, the instance is required. The instance is the host's fully qualified domain name, as in `host/central.example.com`. For user principals, an instance is optional. Note, however, that `jdoe` and `jdoe/admin` are unique principals.

2. In Oracle Solaris, a specific service of a class of System Management Facility (SMF) services. For example, the `compliance:default` instance and the `compliance:generate-guide` instance are separate instances of the `compliance` SMF service.

Kerberos policy A set of rules that governs password usage in the Kerberos service. Policies can regulate principals' accesses, or ticket parameters, such as lifetime.

password policy The requirements for passwords, such as "following industry standards such as password length". Security policy requires passwords. Password policy might dictate password strength, frequency of change, and permitted alphabet, number, and keyboard modifier keys.

policy Generally, a plan or course of action that influences or determines decisions and actions. For computer systems, policy typically means security policy. Your site's security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access.

See also Kerberos policy and password policy.

privilege In general, a power or capability to perform an operation on a computer system that is beyond the powers of a regular user. A privileged user or privileged application is a user or application that has been granted additional rights.

1. In Kerberos, a right granted to a principal by an entry in the `kadm5.acl` file.

2. In Oracle Solaris, privileges are discrete rights on processes, including user processes. Privileges are also called *process privileges* or *kernel privileges*. For a full description of privileges, see the `privileges(5)` man page.

relation In Kerberos, a configuration variable or relationship that is defined in the `kdc.conf` or `krb5.conf` files. In the Oracle Solaris OS, relations are typically called *variables* or *keyword-value pairs*.

security policy See policy.

service 1. In Kerberos, a resource that is provided to network clients, often by more than one server. For example, if you `ssh` to the `central.example.com` host, then that host is the server that provides the `ssh` service.

2. In Oracle Solaris, a program that is managed by the SMF as a service. Services can be enabled, disabled, refreshed, and restarted through SMF commands. The status of services is constantly monitored and logged for ease in tracking and troubleshooting. In Oracle Solaris, the KDC, the Kerberos client, OTP, SASL, and smart card are SMF services.

Index

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X