在 Oracle® Solaris 11.2 中管理 Kerberos 和其他验证服务

文件号码 E53971-02
2014 年 9 月

ORACLE

目录

表

示例

使用本文档

- **概述** – 介绍如何管理一个或多个 Oracle Solaris 系统上的安全验证。本指南涵盖可插拔验证模块 (Pluggable Authentication Module, PAM)、Kerberos、简单验证和安全层 (Simple Authentication and Security Layer, SASL) 以及适用于 NFS 和 NIS 的安全 RPC。本指南中的附录通过一些使用示例来介绍用于 Kerberos 的 DTrace 探测器。
- **目标读者** – 系统管理员、安全管理员和网络安全管理员。
- **必备知识** – 了解访问要求和网络安全要求。

产品文档库

位于 http://www.oracle.com/pls/topic/lookup?ctx=E56344 的文档库中包含此产品的最新信息和已知问题。

获得 Oracle 支持

Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

反馈

可以在 http://www.oracle.com/goto/docfeedback 上提供有关本文档的反馈。

♦ ♦ ♦ 第 1 章

1 使用可插拔验证模块

本章介绍可插拔验证模块 (Pluggable Authentication Module, PAM)。PAM 提供了一个框架，用于插入对 Oracle Solaris OS 上的应用程序用户的检查。PAM 提供了一个管理应用程序使用的中央框架，包括对用户进行验证、管理口令更改、关闭和打开用户会话以及跟踪帐户限制（例如每日时间段限制）。PAM 可扩展到第三方应用程序，因此能够对系统服务访问提供无缝管理。

本章包含以下主题：

Oracle Solaris 11.2 中新增的验证功能

本节重点向现有客户介绍此发行版中 PAM 和 Kerberos 技术新增的重要功能。

PAM 的新增功能

Oracle Solaris 11.2 发行版中的 PAM 框架包括以下新增功能：

- `pam_unix_account` 模块支持对所有或指定 PAM 服务实施基于时间的访问控制。因此，调用 PAM 的安全相关操作可以限定到特定日期和时间。或者，您也可以提供一个时区。请参见 `pam_unix_account(5)` 手册页。
- 可以将 `access_times` 和 `access_tz` 这两个关键字分配给用户和角色。与所有用户安全属性一样，这些关键字在所有名称服务中均受支持。有关更多信息，请参见 `user_attr(4)` 手册页。

Kerberos 的新增功能

Oracle Solaris 11.2 发行版中的 Kerberos 包括以下新增功能：

- Kerberos 支持在任意时间运行经过验证的批处理作业。支持延迟执行的命令（at、batch 和 cron 命令）可以使用 Kerberos 服务来提供验证而无需手动干预。有关更多信息，请参见“为 Kerberos 服务访问配置延迟执行” [109]。
- 可以使用自动化安装程序 (Automated Installer, AI) 自动配置 Kerberos 客户机。这些客户机可以立即用于托管基于 Kerberos 的服务。通过在安装期间自动置备 Kerberos 客户机系统，管理员能快速而轻松地部署安全基础结构，从而将宝贵的系统管理员资源解放出来。
  - 安装的客户机系统不需要执行其他步骤便可指定 Kerberos 配置并为系统创建 Kerberos 服务密钥。
  - 为客户机系统提供 Kerberos 服务密钥时，有不同选项可用。

  有关通过自动化安装程序配置 Kerberos 的更多信息，请参见《安装 Oracle Solaris 11.2 系统 》中的“如何使用 AI 配置 Kerberos 客户机”。
- 缺省情况下，ktkt_warn 服务处于禁用状态。必须显式启用该服务，才能警告用户 TGT 初始凭证过期或自动更新用户的初始凭证。有关更多信息，请参见“自动更新所有票证授予票证” [100]。

有关 Oracle Solaris 中的 Kerberos 历史记录的信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“各种 Kerberos 发行版的组件”。

关于 PAM

PAM 为应用程序提供了一个用于执行各项验证功能的框架。它为应用程序（包括系统程序）的用户提供集中验证、会话管理、口令管理和帐户限制功能。PAM 使 login、su 和 ssh 这类程序能够在用户管理详细信息更改时保持不变。站点应用程序可以使用 PAM 来管理自己的帐户、凭证、会话和口令要求。PAM 以插件的形式嵌入到这些应用程序。

PAM 框架介绍

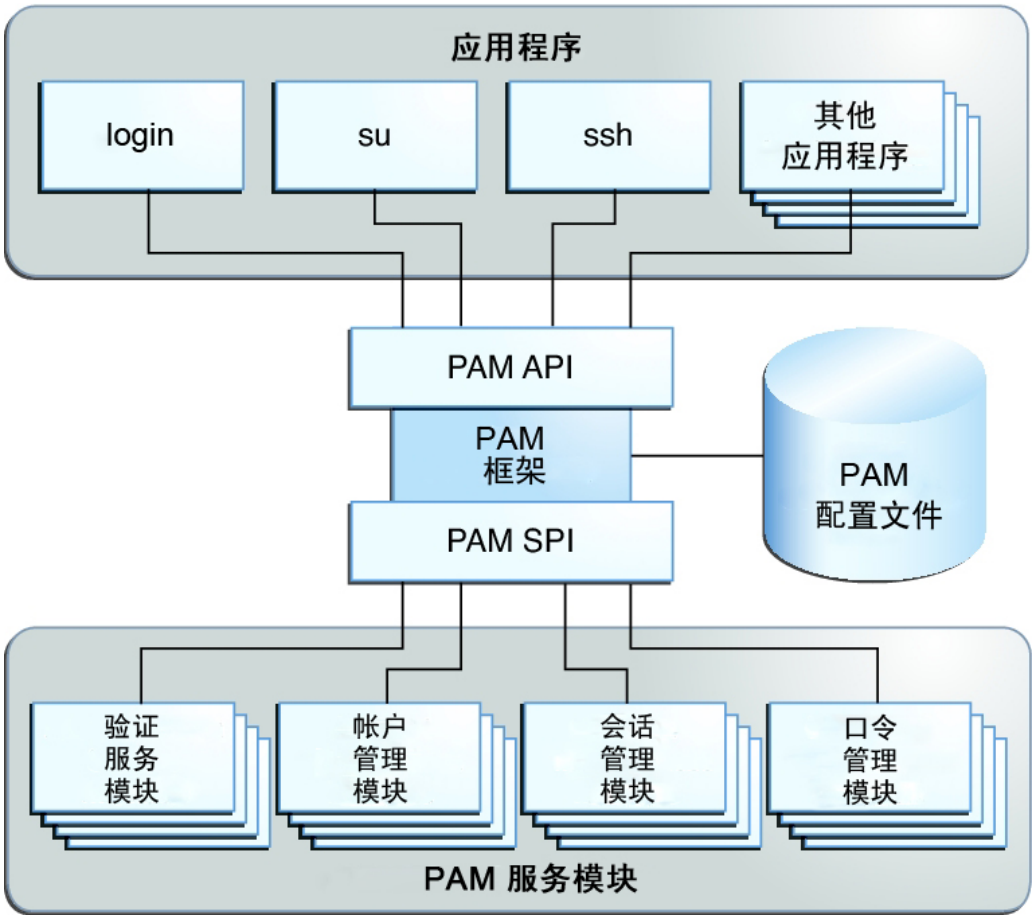
PAM 框架由四个部分组成：

- 使用 PAM 的应用程序
- PAM 框架
- PAM 服务模块
- PAM 配置（包括模块选择和用户分配）

该框架为与验证相关的活动提供了统一的执行方式。采用该方式，应用程序开发者不必了解验证策略的语义即可使用 PAM 服务。借助 PAM，管理员可以根据特定系统的需要调整验证过程，而不必更改任何应用程序。管理员只需调整 PAM 配置即可。

下图说明了 PAM 体系结构。

图 1-1 PAM 体系结构



该体系结构的工作原理如下：

- 应用程序通过 PAM 应用编程接口 (Application Programming Interface, API) 与 PAM 框架进行通信。

  有关使用 API 的信息，请参见 pam(3PAM) 手册页和《面向开发者的 Oracle Solaris 11 安全性指南》中的第 3 章 “编写 PAM 应用程序和服务”。

- PAM 服务模块通过 PAM 服务提供者接口 (Service Provider Interface, SPI) 与 PAM 框架进行通信。有关更多信息，请参见 pam_sm(3PAM) 手册页。

  有关所选服务模块的简要说明，请参见“PAM 服务模块” [34]以及 pam.conf(4) 和 pam_user_policy(5) 手册页。

管理员可以配置一个或多个模块系列来管理站点要求。该模块系列称为 PAM 栈。该栈将按顺序进行评估。如果应用程序需要多个 PAM 栈，应用程序开发者必须创建多个服务名称。例如，sshd 守护进程提供了几个用于 PAM 的服务名称，这些服务名称是 PAM 必需的。有关 sshd 守护进程的 PAM 服务名称的列表，请在 sshd(1M) 手册页中搜索单词 PAM。有关 PAM 栈的详细信息，请参见“PAM 栈” [30]。“PAM 栈示例” [33]中逐步介绍了 PAM 验证栈。

使用 PAM 的益处

通过 PAM 框架，您可以配置用户使用应用程序必须满足的要求。以下是 PAM 提供的部分益处：

- 灵活的 PAM 配置策略
  - 基于服务名称的验证策略
  - 站点范围 PAM 策略和基于用户的 PAM 策略
  - 管理员可以选择缺省验证策略
  - 在安全级别高的系统上强制执行多重用户要求
- 易于最终用户使用
  - 无需为不同的验证服务重新键入完全相同的口令
  - 通过多个验证服务向用户发出提示，而不要求用户键入多个命令
- 方便管理员进行配置
  - 能够将选项传递到 PAM 服务模块
  - 能够实施特定于站点的安全策略，而无需更改应用程序

规划特定于站点的 PAM 配置

交付时提供的 PAM 配置实施的标准安全策略涵盖了需要验证的系统服务（例如，login 和 ssh）。如果需要为部分系统服务实施不同的安全策略或需要为第三方应用程序创建策略，请考虑以下问题：

- 确定所提供的配置文件不满足要求。

  测试缺省配置。测试 /etc/security/pam_policy 目录中基于用户的文件。测试缺省服务名称 other 是否能够满足要求。“PAM 栈示例” [33]完整介绍了 other 栈。
- 识别栈需要修改的任何服务名称。有关修改服务名称的 PAM 栈的示例，请参见如何创建特定于站点的 PAM 配置文件 [20]。

- 对于通过编码方式使用 PAM 框架的任何第三方应用程序，请确定该应用程序所使用的 PAM 服务名称。
- 对于每个服务名称，请确定要使用的 PAM 模块。

  有关 PAM 模块的信息，请查看手册页的第 5 部分。这些手册页介绍每个模块的工作方式、可用的选项以及堆叠模块之间的交互。有关所选模块的简要说明，请参见“PAM 服务模块” [34]。PAM 模块还可以从外部资源获取。
- 基于服务名称确定模块的运行顺序。
- 选择每个模块的控制标志。有关控制标志的更多信息，请参见“PAM 栈” [30]。请注意，控制标志可能会有安全隐患。

  有关直观图示，请参见图 1-2 “PAM 栈：控制标志的作用” 和图 1-3 “PAM 栈：如何确定集成值”。
- 选择每个模块必需的选项。每个模块的手册页列出了可用于该模块的选项。
- 使用 PAM 配置测试应用程序使用情况。分别以 `root` 角色、其他角色、特权用户和一般用户进行测试。如果部分用户无权使用应用程序，则对这些用户进行测试。

分配基于用户的 PAM 策略

`pam_user_policy` PAM 模块允许系统管理员基于用户分配特定 PAM 配置。当此模块为 PAM 栈中的第一个模块，并且用户的 `pam_policy` 安全属性指定了 PAM 配置文件时，该文件将指定该用户的 PAM 策略。

有关更多信息，请参见以下内容：

- `pam_user_policy(5)` 手册页
- “PAM 栈” [30]
- 如何创建特定于站点的 PAM 配置文件 [20]
- 例 1-3 “使用权限配置文件设置基于用户的 PAM 策略”

配置 PAM

可以按原样使用 PAM。本节介绍缺省情况下未生效的 PAM 配置的示例。

表 1-1 PAM 任务列表

任务	说明	有关说明
规划 PAM 安装。	介绍如何为站点规划 PAM 定制。	“规划特定于站点的 PAM 配置” [18]
向用户指定新的 PAM 策略。	为多个服务定制基于用户的验证要求。	如何创建特定于站点的 PAM 配置文件 [20]
为用户创建加密的起始目录。	修改 PAM 栈以创建加密的起始目录。	例 1-1 “使用修改后的 PAM 栈创建加密的起始目录”

任务	说明	有关说明
添加新的 PAM 模块。	说明如何安装和测试定制的 PAM 模块。	如何添加 PAM 模块 [22]
将非缺省 PAM 策略分配给用户。	说明如何将 PAM 策略添加到权限配置文件中，以便分配给使用 Kerberos、LDAP 或登录组合的站点上的一系列用户。	如何分配修改后的 PAM 策略 [24]
将非缺省 PAM 策略分配给用户。	将定制的 PAM 栈分发到所有系统映像。	如何分配修改后的 PAM 策略 [24]
启动错误日志记录。	通过 `syslog` 记录 PAM 错误消息。	如何记录 PAM 错误报告 [26]
排除 PAM 错误。	提供用于查找、解决和测试 PAM 错误配置的步骤。	如何排除 PAM 配置错误 [27]

▼ 如何创建特定于站点的 PAM 配置文件

在缺省配置中，`ssh` 和 `telnet` 登录服务涵盖在 `other` 服务名称下。以下过程中的 PAM 配置文件会更改 `ssh` 和 `telnet` 的要求。

开始之前 您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **创建一个新的 PAM 策略配置文件。**

   使用 `pfedit` 命令创建文件。将该文件放到站点配置目录，例如 `/opt`。也可以将该文件放到 `/etc/security/pam_policy` 目录。

   注 - 请勿修改 `/etc/security/pam_policy` 目录中的现有文件。

   在该文件中添加解释性注释。

```
# 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. **保护该文件。**

   使用 `root` 所有权和 `444` 权限保护该文件。

```
# ls -l /opt/local_pam

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

3. 分配策略。
   请参见如何分配修改后的 PAM 策略 [24]。

例 1-1 使用修改后的 PAM 栈创建加密的起始目录

缺省情况下，zfs_pam_key 模块不在 /etc/security/pam_policy/unix 文件中。在本示例中，管理员创建了一个 unix 版本的基于用户的 PAM 策略，然后使用新版本创建起始目录加密的用户。

```
# 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
```

管理员添加用户时使用此策略文件。请注意，加密不能添加到文件系统。必须在启用加密的情况下创建文件系统。有关更多信息，请参见 zfs_encrypt(1M)。

管理员创建用户并分配口令，

```
# 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
```

然后以该用户身份登录来创建加密的起始目录。

```
# 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      11.2    July 2014

# logout
```

有关 ZFS 服务模块的选项，请参见 pam_zfs_key(5) 手册页。

最后，管理员将验证新的起始目录是否为加密的文件系统。

```
# 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
```

▼ 如何添加 PAM 模块

此过程说明如何添加、保护和测试新的 PAM 模块。您可能需要将新模块用于特定于站点的安全策略，或用于支持第三方应用程序。要创建 PAM 模块，请参见《面向开发者的 Oracle Solaris 11 安全性指南 》中的第 3 章 “编写 PAM 应用程序和服务”。

注 - 必须安装 32 位版本和 64 位版本的 PAM 服务模块。

开始之前 完成“规划特定于站点的 PAM 配置” [18]中所述的步骤。

您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **在磁盘上安装两个版本的 PAM 服务模块，然后保护这些模块。**

   确保保护模块文件的所有权和权限是 root 所有权和 444 权限。

```
# 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
```

   32 位模块位于 /opt/pam_modules 目录中，而 64 位模块位于 64 子目录中。

2. **将模块添加到相应的 PAM 配置文件。**

   在以下示例中，模块用于新的应用程序 (app1)。其服务名称与应用程序名称相同。在 /etc/pam.d 目录中创建 app1 服务名称文件。该文件中的第一项使 app1 服务能够分配给单个用户。

```
# 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
```

模块路径中的 $ISA 标记根据调用应用程序将 PAM 框架定向到相应的 32 位或 64 位体系结构版本的服务模块。对于 32 位应用程序，*/a/b/*$ISA/*module*.so 将变为 */a/b/module*.so，对于 64 位应用程序，将变为 */a/b/*64/*module*.so。在本示例中，在 /opt/pam_modules 目录中安装了 32 位的 pam.app1.so.1 服务模块，在 /opt/pam_modules/64 目录中安装了 64 位的模块。

有关更多信息，请参见 pfedit(1M) 和 pam.conf(4) 手册页。

要将 app1 PAM 策略限定到所选用户，请参见例 1-2 “将新模块添加到基于用户的 PAM 策略文件”。

3. **测试新服务。**

   直接使用 login 或 ssh 登录。然后，运行受新模块影响的命令。分别对允许和拒绝使用受影响命令的用户进行测试。有关故障排除帮助，请参见如何排除 PAM 配置错误 [27]。

4. **分配策略。**

   请参见如何分配修改后的 PAM 策略 [24]。

例 1-2 将新模块添加到基于用户的 PAM 策略文件

在本示例中，app1 服务并非供所有用户使用，因此管理员可将该服务添加为基于用户的策略。

```
# 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
```

管理员将 app1 文件从 pam.d 目录中删除，

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

然后将 app1-conf 策略添加到系统管理员的 PAM 策略。

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

例 1-3 使用权限配置文件设置基于用户的 PAM 策略

本示例使用 pam_policy 安全属性对来自不同命名服务的用户进行验证。any PAM 策略文件在 /etc/security/pam_policy 目录中提供。该文件中的注释对此策略进行了描述。

请勿修改此目录中的文件。

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

要分配此权限配置文件，请参见如何分配修改后的 PAM 策略 [24]。

▼ 如何分配修改后的 PAM 策略

在此过程中，您将在所有系统映像上配置非缺省 PAM 策略。复制所有文件后，可以将新的或修改后的 PAM 策略分配给单个或所有用户。

开始之前 您已对实施新策略的 PAM 配置文件进行了修改和测试。

您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **将非缺省 PAM 文件添加到所有系统映像。**
   必须将所有新的 PAM 模块以及新的和修改后的 PAM 配置文件添加到所有系统映像。

   a. **首先，将任何新的 PAM 模块添加到每个系统映像。**

      i. **将 32 位 PAM 模块添加到相应体系结构的的目录。**

      ii. **将 64 位 PAM 模块添加到相应体系结构的的目录。**

      有关目录设置的示例，请参见如何添加 PAM 模块 [22]中的步骤 1。

   b. **接着，将任何新的 PAM 配置文件添加到每个系统映像。**
      例如，将 `/opt/local_pam/ssh-telnet-conf` 文件添加到每个系统映像。

   c. **然后，将任何修改后的 PAM 配置文件复制到每个系统映像。**
      例如，将修改后的 `/etc/pam.conf` 文件以及任何修改后的 `/etc/pam.d/`*service-name-files* 复制到每个系统映像。

2. **将非缺省 PAM 策略分配给所有用户。**

   a. **通过以下方法之一修改 `policy.conf` 文件：**

      - **将 PAM 配置文件添加到 `policy.conf` 文件中的 `PAM_POLICY` 关键字。**

        ```
        # pfedit /etc/security/policy.conf
        ```

```
...
# PAM_POLICY=
PAM_POLICY=/opt/local_pam/ssh-telnet-conf
...
```

- 将权限配置文件添加到 **policy.conf** 文件中的 **PROFS_GRANTED** 关键字。

  例如，分配例 1-3 “使用权限配置文件设置基于用户的 PAM 策略” 中在 "Any"（任何）权限配置文件内设置的基于用户的 PAM 策略。

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

b. 将修改后的 **policy.conf** 文件复制到每个系统映像。

3. 要将非缺省 PAM 策略分配给单个用户，可以直接将该策略分配给用户，也可以将该策略添加到已分配给用户的权限配置文件。

   - 直接将 PAM 策略分配给单个用户。

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

   - 将 PAM 策略包含到权限配置文件，然后将该配置文件分配给单个用户。

     本示例使用的是 `ldap` PAM 策略。

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

     然后将权限配置文件分配给用户。

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

例 1-4 将 `ktelnet` PAM 栈限定到所选用户

管理员希望只有有限数量的用户能在 Kerberos 领域中使用 `telnet`。因此，启用 `telnet` 服务之前，管理员应更改缺省的 `ktelnet` 配置文件，并将缺省的 `ktelnet` 文件放到 `pam_policy` 目录。

首先，管理员配置一个基于用户的 `ktelnet` 文件。

```
# 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
```

管理员使用 444 权限保护该文件。

```
# 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
```

然后，管理员修改 pam.d 目录中的 ktelnet 文件。

- 第一项启用基于用户的分配。
- 第二项拒绝 ktelnet 使用，除非管理员为您分配了 pam_policy=ktelnet。

```
# 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
```

管理员分别以特权用户、一般用户和 root 角色对配置进行测试。配置通过测试后，管理员将启用 telnet 服务，并将基于用户的策略分配给 Kerberos 管理员。

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

管理员将修改后的文件复制到所有 Kerberos 服务器，并在这些服务器上启用 telnet。

▼ 如何记录 PAM 错误报告

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **确定联机的 system-log 服务实例。**

```
# 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. **配置 syslog.conf 文件以设置您需要的日志记录级别。**

有关日志记录级别的信息，请参见 syslog.conf(4) 手册页的“描述”一节。大多数 PAM 错误报告由 LOG_AUTH 工具执行。

例如，创建调试输出文件。

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

然后添加 `syslog.conf` 项，以将调试输出发送到该文件。

注 - 如果 `rsyslog` 服务实例联机，请修改 `rsyslog.conf` 文件。

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

3. 刷新 **system-log** 服务的配置信息。

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

注 - 如果 `rsyslog` 服务联机，请刷新 `system-log:rsyslog` 服务实例。

▼ 如何排除 PAM 配置错误

开始之前 您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 为要进行故障排除的每个 PAM 项添加 **debug** 选项。

   例如，`/etc/pam.d/cron` 文件中的以下项将创建该服务的调试输出。

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

2. 记录相应级别的 PAM 错误并刷新 **syslog** 守护进程。

   有关详细信息，请参见如何记录 PAM 错误报告 [26]。

3. 如果问题在于 PAM 配置损坏，请执行以下操作：

   a. 从一个终端窗口运行应用程序，在另一个窗口中修改 PAM 配置文件。

   b. 通过在应用程序窗口中对更改进行测试，验证错误是否已更正。

4. 如果问题在于 PAM 配置损坏而无法登录，请以单用户模式引导，然后依次执行以下操作：更正文件、重新引导和测试。

   - 要引导 SPARC 系统，请在 PROM 提示符下键入以下命令：

```
ok > boot -s
```

- **要引导 x86 系统，请在 GRUB 菜单中将 -s 选项添加到内核选项行。**

有关更多信息，请参见 boot(1M) 和 grub(5) 手册页。

5. **验证错误是否已更正。**

直接使用 login 或 ssh 登录。分别测试一般用户、特权用户和角色是否能够使用受影响的命令。

PAM 配置参考信息

本节提供有关 PAM（包括 PAM 栈）的其他详细信息。

PAM 配置文件

使用 PAM 框架的系统应用程序（例如，login 和 ssh）在 /etc/pam.d 目录的 PAM 配置文件中进行配置。也可以使用 /etc/pam.conf 文件。对这些文件所做的更改将影响系统上的所有用户。

此外，/etc/security/pam_policy 目录也保留 PAM 配置文件。这些文件涵盖多个服务，专为基于用户的分配而设计。不得修改此目录中的文件。

- **/etc/pam.d 目录** – 包含特定于服务的 PAM 配置文件，并且包括通配符文件 (other)。要为应用程序添加服务，请添加一个 *service-name* 文件，该文件以应用程序所使用的服务名称命名。如果适用，应用程序可以使用 other 文件中的 PAM 栈。

  /etc/pam.d 目录中的服务文件提供了适用于大多数 PAM 实现的缺省配置。这些文件使用 IPS 机制进行自汇编，如 pkg(5) 手册页中所述。此缺省设置可简化与其他跨平台 PAM 应用程序的互操作性。有关更多信息，请参见 pam.conf(4) 手册页。
- **/etc/pam.conf 文件** – 传统 PAM 配置和策略文件。此文件交付时为空。配置 PAM 的首选机制是使用 /etc/pam.d 目录中的文件。有关更多信息，请参见 pam.conf(4) 手册页。
- **/etc/security/pam_policy 目录** – 包含 PAM 策略文件（其中包含用于多个服务的策略）。可以根据需要将这些文件分配给单个用户、一组用户或所有用户。此类分配将覆盖 pam.conf 或 /etc/pam.d 目录中的系统 PAM 配置文件。请勿修改这些文件。要添加基于用户的文件，请参见如何创建特定于站点的 PAM 配置文件 [20]。有关基于用户的文件的信息，请参见 pam_user_policy(5) 手册页。

由安全管理员管理所有 PAM 配置文件。项顺序不正确（也就是 PAM 栈不正确）会导致无法预料的负面影响。例如，如果文件配置错误，则可能会将多个用户锁定，这样必须

通过单用户模式才能进行修复。有关帮助，请参见“PAM 栈” [30]和如何排除 PAM 配置错误 [27]。

PAM 配置搜索顺序

对 PAM 框架的应用程序调用按以下顺序搜索已配置的 PAM 服务：

1. 在 `/etc/pam.conf` 文件中查找服务名称。
2. 使用 `/etc/pam.d/`*service-name* 文件中的特定服务。
3. 在 `/etc/pam.conf` 文件中检查服务名称 `other`。
4. 使用 `/etc/pam.d/other` 文件。

按照此顺序，现有的 `/etc/pam.conf` 文件能够与 `/etc/pam.d` 目录中基于服务的 PAM 配置文件协同工作。

PAM 配置文件语法

`pam.conf` 文件和 PAM 基于用户的文件所使用的语法与 `pam.d` 目录中特定于服务的文件不同。

- `/etc/pam.conf` 文件和 `/etc/security/pam_policy` 文件中的项采用以下两种格式之一：

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

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

- `/etc/pam.d` 目录的 *service-name* 文件中的项省略服务名称。文件名提供了服务名称。

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

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

PAM 配置文件的语法项如下所示：

service-name

服务的名称（不区分大小写），例如 `login` 或 `ssh`。应用程序可以针对其提供的各服务使用不同的服务名。例如，在 `sshd(1M)` 手册页中搜索单词 `PAM` 可查找 `sshd` 守护进程所提供的各个服务的服务名称。

如果未提供任何特定的服务配置，则预定义的服务名称 (`other`) 为缺省服务名称。

module-type

指示服务类型，即 `auth`、`account`、`session` 或 `password`。

control-flag

指示模块在确定服务的成功失败值时所起的作用。有效的控制标志在“PAM 栈” [30]中有介绍。

module-path

此路径指向实现模块类型的模块。如果路径名不是绝对的，则认为该路径相对于 `/usr/lib/security/$ISA/` 路径。`$ISA` 宏或标记指示 PAM 框架查找模块路径特定于体系结构的目录。

module-options

可传递到服务模块的选项，例如 `nowarn` 和 `debug`。模块的手册页介绍了可用于该模块的选项。

path-to-included-PAM-configuration

指定 PAM 配置文件的完整路径，或者指定相对于 `/usr/lib/security` 目录的文件名。

PAM 栈

当应用程序调用以下函数之一时，PAM 框架将读取 PAM 配置文件以确定实现该应用程序的 PAM 服务名称的模块：

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

如果配置文件仅包含一个模块，则该模块的结果将决定操作的结果。例如，`passwd` 应用程序的缺省验证操作包含一个模块，即 `/etc/pam.d/passwd` 文件中的 `pam_passwd_auth.so.1`。

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

另一方面，如果由多个模块实现一个服务，则这些模块被视为堆叠，也就是说，该服务名称存在 PAM 栈。例如，考虑 `/etc/pam.d/login` 服务示例中的以下项：

```
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
```

这些项将为 `login` 服务名称创建一个 `auth` 栈。要确定此栈的结果，各个模块的结果代码需要进行集成处理。

在集成过程中，模块按照其在文件中的顺序执行。每个成功/失败代码都会根据模块的控制标志集成到整体结果中。控制标志可能会导致栈很早就终止。例如，`requisite` 或 `definitive` 模块失败会终止栈。即使先前没有任何模块失败，`sufficient`、`definitive` 或 `binding` 模块的成功也会终止栈。处理栈之后，各个结果会合并成单个整体结果，然后传送给应用程序。有关该流程的图形，请参见图 1-2 “PAM 栈：控制标志的作用”和图 1-3 “PAM 栈：如何确定集成值”。

控制标志指示 PAM 模块在确定成功失败时所起的作用。控制标志及其作用如下所述：

- Binding – 如果先前未记录到任何模块失败，则成功满足 binding 模块的要求后，会立即向应用程序返回成功信息。如果满足这些条件，则不会进一步执行模块。

  如果失败，则会记录必需的失败信息并继续处理模块。
- Definitive – 如果先前未记录到任何模块失败，则成功满足 definitive 模块的要求后，会立即向应用程序返回成功信息。

  如果先前记录到某个模块失败，则会立即向应用程序返回该失败信息，而不会进一步执行模块。如果失败，则会立即返回错误信息，而不会进一步执行模块。
- Include – 添加单独的 PAM 配置文件中的行，以便在 PAM 栈中的此点使用。此标志不控制成功或失败行为。读取一个新文件时，PAM `include` 栈的值会递增。当新文件中的栈检查完成时，`include` 栈的值会递减。当到达文件末尾，并且 PAM `include` 栈的值为 `0` 时，栈处理将结束。PAM `include` 栈的最大数为 `32`。
- Optional – 不必成功满足 optional 模块的要求即可使用服务。

  如果失败，则会记录可选的失败信息。
- Required – 必须成功满足 required 模块的要求才能使该栈成功。仅当 binding 模块或 required 模块均没有报告失败时，才会返回该栈最终成功的信息。

  如果失败，则执行该服务的其余模块后会返回错误信息。
- Requisite – 必须成功满足 requisite 模块的要求才能使该栈成功。仅当栈中的所有 requisite 模块都返回成功信息时，该栈才能向应用程序返回成功信息。

  如果失败，则会立即返回错误信息，而不会进一步执行模块。
- Sufficient – 如果先前未记录到任何 required 模块失败，则成功满足 sufficient 模块的要求后，会立即向应用程序返回成功信息，而不会进一步执行模块。

  如果失败，则会记录可选的失败信息。

以下两个相连的图显示了在集成过程中如何确定结果。

- 第一个图显示了每种类型的控制标志如何记录成功失败信息。结果显示在第二个图中。
- 第二个图说明如何确定集成值。Optional 模块失败和 required 模块失败会返回失败代码，成功则会返回成功代码。应用程序将决定如何处理这些返回码。

图 1-2 PAM 栈：控制标志的作用

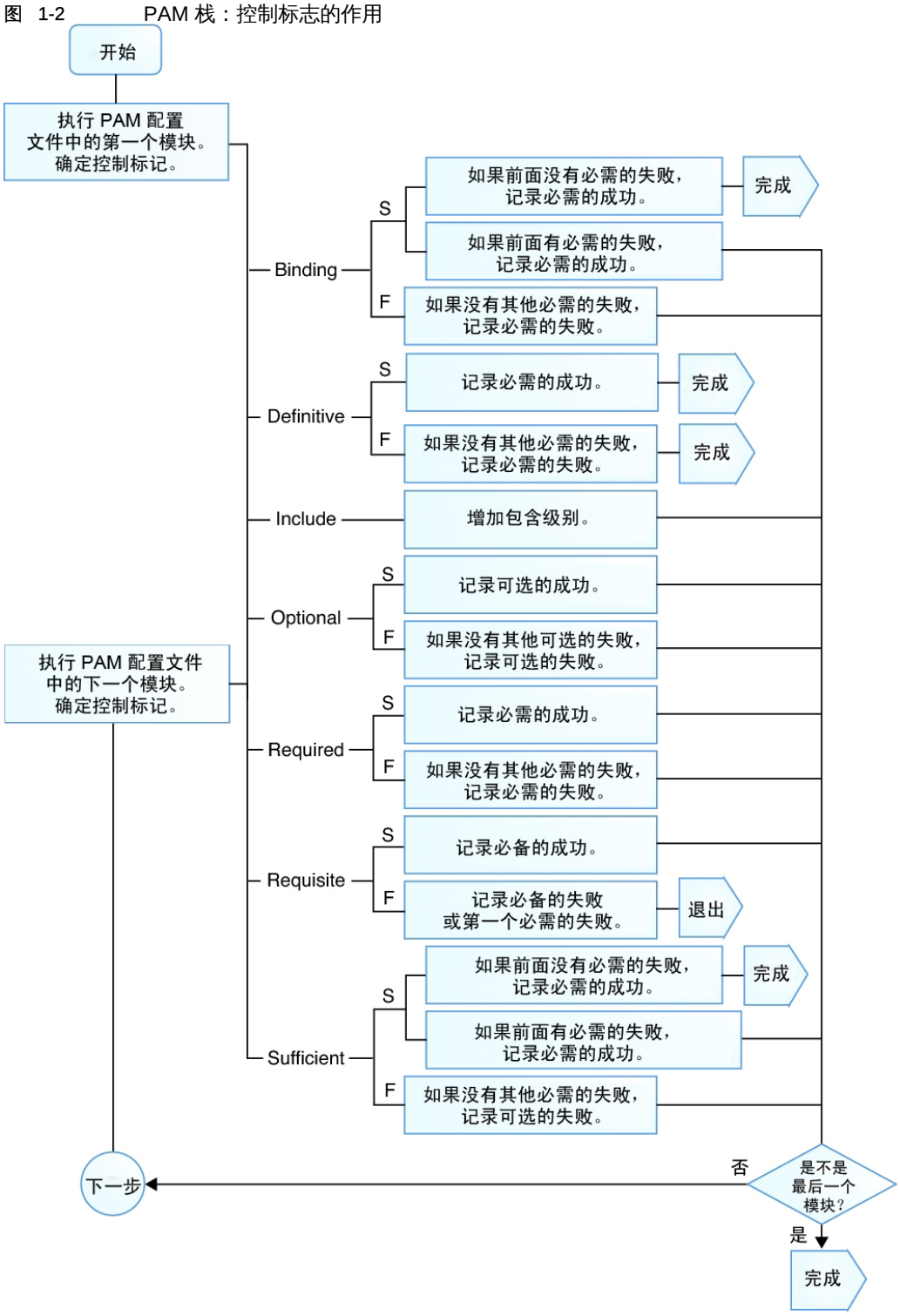
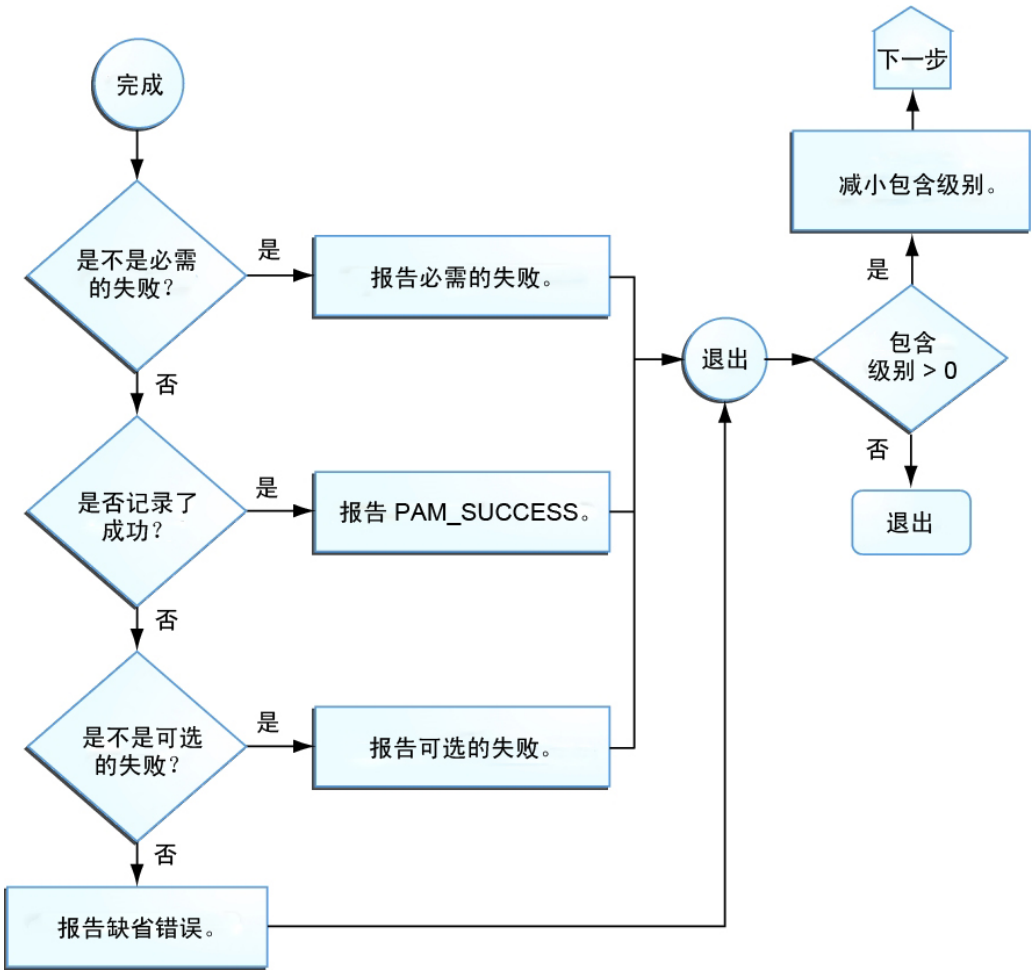


图 1-3 PAM 栈：如何确定集成值



PAM 栈示例

以下示例显示了 `/etc/pam.d/other` 文件样例中的缺省验证管理定义。如果未配置任何特定于服务的验证定义，则将这些定义用于验证。

```
##
# 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
```

首先，使用 pam_user_policy.so 模块检查用户的 PAM 策略。definitive 控制标志指示当配置的 PAM 栈评估结果为成功时，向应用程序返回成功代码，因为此时未检查其他模块。如果配置的 PAM 栈评估结果为失败，则会向应用程序返回失败代码，并且不会进一步检查。如果未向此用户分配任何基于用户的 PAM 策略，则将执行下一个模块。

如果未向此用户分配基于用户的 PAM 策略，则将执行 pam_authtok_get 模块。此模块的控制标志设置为 requisite。如果 pam_authtok_get 失败，则验证过程将结束并向应用程序返回失败代码。

如果 pam_authtok_get 没有失败，则将执行接下来的三个模块。这些模块配置有 required 控制标志，因此无论是否返回了单个失败代码，集成处理过程都将继续进行。执行 pam_unix_cred 后，不再执行剩余的任何模块。此时，如果所有模块均已成功，则会向应用程序返回成功代码。如果 pam_dhkeys、pam_unix_auth 或 pam_unix_cred 中的任意一个返回了失败代码，则会向应用程序返回失败代码。

PAM 服务模块

本节有选择地列出了一些 PAM 服务模块。这些模块按手册页列出，后面附有在何时何地使用这些模块的简要说明。有关更多信息，请阅读手册页。

有关 Oracle Solaris 所提供的所有 PAM 服务模块的列表，请参见手册页的第 5 部分。Oracle 公司会定期添加新模块。例如，此发行版添加了许多用于在 Windows 系统中进行验证的模块。您的站点可能还会添加来自第三方的 PAM 模块。

pam_allow(5)	为所有调用返回 PAM_SUCCESS。另请参见 pam_deny(5) 手册页。
pam_authtok_check(5)	验证口令更改的口令令牌。
pam_authtok_get(5)	向 PAM 栈提供口令提示功能。
pam_authtok_store(5)	更新 PAM_USER 的口令令牌。
pam_deny(5)	为所有调用返回模块类型的缺省失败返回码。另请参见 pam_allow(5) 手册页。
pam_dhkeys(5)	向以下两个 PAM 服务提供功能：安全 RPC 验证和安全 RPC 验证令牌管理。

pam_krb5(5)	提供以下两项功能：验证 Kerberos 用户身份和管理 Kerberos 凭证高速缓存。
pam_krb5_migrate(5)	帮助将 PAM_USER 迁移到客户机的本地 Kerberos 领域。
pam_ldap(5)	为配置的 LDAP 目录服务器所使用的 PAM 验证和帐户管理栈提供功能。
pam_list(5)	提供验证此主机上的用户帐户的功能。验证基于主机上的用户和网络组列表。
pam_passwd_auth(5)	向口令栈提供验证功能。
pam_pkcs11(5)	允许用户使用 X.509 证书及其存储在 PKCS#11 令牌中的专用私钥登录到某个系统。
pam_roles(5)	验证用户是否已获得承担某个角色的授权，并防止直接以某个角色登录。
pam_smb_passwd(5)	支持更改或添加本地 Oracle Solaris 用户的 SMB 口令。另请参见 smb(4) 手册页。
pam_smbfs_login(5)	在 Oracle Solaris 客户机及其 CIFS/SMB 服务器之间同步口令。
pam_tsol_account(5)	验证 Trusted Extensions 帐户限制是否与标签相关。
pam_tty_tickets(5)	提供一种机制来检查先前的成功验证所创建的票证。
pam_unix_account(5)	提供以下功能：验证用户帐户是否未锁定或未失效；验证用户口令是否不需要更改。 包括对 access_times 和 access_tz 进行检查。
pam_unix_auth(5)	提供验证口令是否为 PAM_USER 的正确口令的功能。
pam_unix_cred(5)	提供建立用户凭证信息的功能。通过此模块，可以独立于凭证功能替换验证功能。
pam_unix_session(5)	打开和关闭会话，同时更新 /var/adm/lastlog 文件。
pam_user_policy(5)	调用特定于用户的 PAM 配置。
pam_zfs_key(5)	提供装入和更改用户加密起始目录的 ZFS 加密口令短语的功能。

♦ ♦ ♦ 第 2 章

2 关于 Kerberos 服务

本章介绍 Kerberos 服务。本章包含以下信息：

什么是 Kerberos 服务？

Kerberos 服务是一种客户机/服务器体系结构，用于在网络上提供安全事务。该服务可提供功能强大的用户验证以及完整性和保密性。通过验证，可保证网络事务的发送者和接收者的身份真实。该服务还可以检验来回传递的数据的有效性（完整性），并在传输过程中对数据进行加密（保密性）。使用 Kerberos 服务，可以安全登录到其他计算机、执行命令、交换数据以及传输文件。此外，该服务还提供授权服务，这样，管理员便可限制对服务和计算机的访问。而且，作为 Kerberos 用户，您还可以控制其他用户对您帐户的访问。

Kerberos 服务是单点登录系统，这意味着您对于每个会话只需要向服务进行一次自我验证。会话过程中的所有后续事务将自动受到保护。在您通过该服务的验证后，便不需要在每次使用基于 Kerberos 的命令（如 `ftp` 或 `ssh`）或访问 NFS 文件系统上数据的命令时，都证明您自己的身份。因此，无需在每次使用这些服务时都在网络上发送口令（口令在网络上可能会被拦截）。

Oracle Solaris 中的 Kerberos 服务基于麻省理工学院 (Massachusetts Institute of Technology, MIT) 开发的 Kerberos V5 网络验证协议。使用过 Kerberos V5 产品的用户会感觉对 Oracle Solaris 版本非常熟悉。由于 Kerberos V5 协议是实际的网络安全行业标准，因此 Oracle Solaris 版本可在异构网络上实现安全事务。此外，该服务还会在各个域之间以及单个域内提供验证和安全服务。

通过 Kerberos 服务可灵活运行 Oracle Solaris 应用程序。可以将该服务配置为允许向网络服务（如 NFS 服务和 `ftp`）发出基于 Kerberos 的请求和不基于 Kerberos 的请求。因此，即使当前应用程序运行的系统上未启用 Kerberos 服务，这些应用程序仍能工作。当然，也可以将 Kerberos 服务配置为仅允许基于 Kerberos 的网络请求。

通过 Kerberos 服务安全机制，在使用采用通用安全服务应用编程接口 (Generic Security Service Application Programming Interface, GSS-API) 的应用程序时，可使用 Kerberos 提供验证、完整性和保密性服务。但是，如果开发了其他安全机制，则应用程序无需继续承诺使用 Kerberos 服务。因为该服务设计为以模块形式集成到 GSS-API 中，所以采用 GSS-API 的应用程序可以选择最符合其需求的安全机制。

Kerberos 服务的工作原理

本节概述 Kerberos 验证系统。有关更多详细说明，请参见“Kerberos 凭证如何提供对服务的访问” [48]。

从用户的角度来看，启动 Kerberos 会话后，Kerberos 服务通常不可见。`ssh` 或 `ftp` 等命令也是如此。初始化 Kerberos 会话通常仅包括登录和提供 Kerberos 口令。

Kerberos 系统围绕票证概念展开。票证是一组标识用户或服务（如 NFS 服务）的电子信息。正如您的驾驶证可标识您的身份并表明您的驾驶级别一样，票证也可标识您的身份以及您的网络访问特权。执行基于 Kerberos 的事务（例如，请求已挂载 NFS 的文件）时，您将透明地向密钥分发中心 (Key Distribution Center, KDC) 发送票证请求。KDC 会访问数据库验证您的身份，并返回向您授予 NFS 服务器访问权限的票证。“透明”指的是您不需要显式请求票证。尝试访问服务器时就会发送请求。因为只有通过验证的客户机可以获取特定服务的票证，所以其他客户机不能以虚假身份访问 NFS 服务器。

票证具有与之关联的特定属性。例如，票证可以是可转发的，这意味着它可以在其他计算机上使用，而不必进行新的验证。票证也可以是以后生效的，这意味着它要到指定时间后才会生效。票证的使用方式（例如，指定允许哪些用户获取哪些类型的票证）由策略设置。策略在安装或管理 Kerberos 服务时确定。

注 - 您可能会经常看到术语凭证和票证。在更为广泛的 Kerberos 范围内，两者通常可互换使用。但是，从技术上讲，凭证指的是票证和会话的会话密钥。“Kerberos 凭证如何提供对服务的访问” [48]中对这一区别进行了更详细的说明。

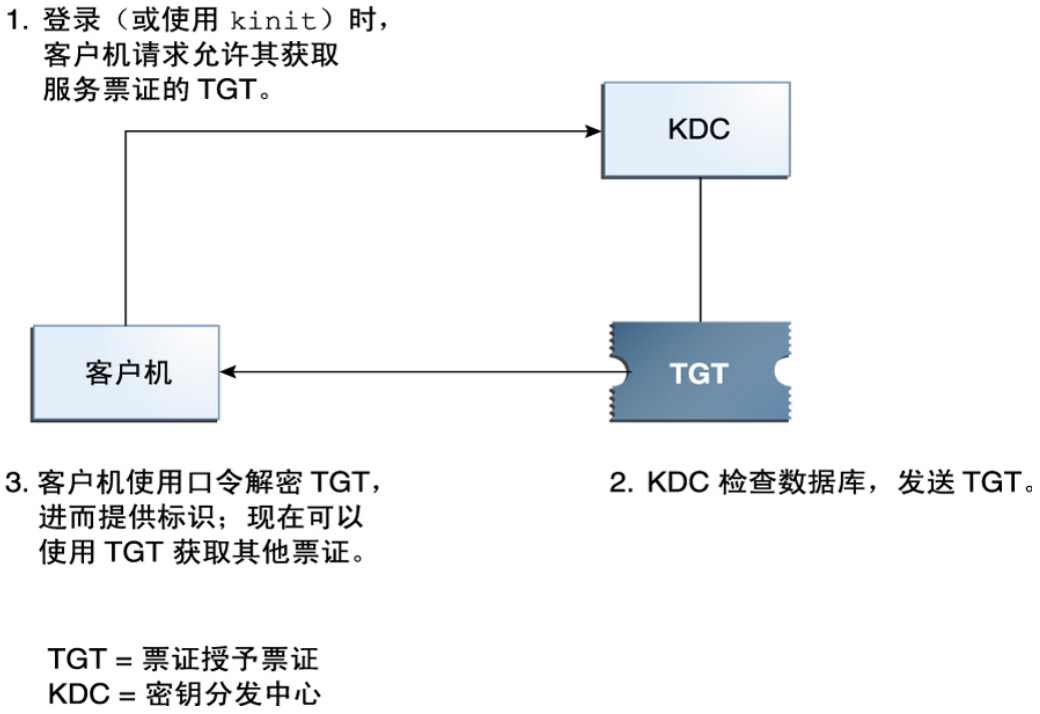
以下各节将进一步说明 Kerberos 验证过程。

初始验证：票证授予票证

Kerberos 验证分为两个阶段：允许进行后续验证的初始验证以及所有后续验证本身。

下图显示了如何进行初始验证。

图 2-1 Kerberos 会话的初始验证



1. 客户机（用户或 NFS 等服务）通过从密钥分发中心 (Key Distribution Center, KDC) 请求票证授予票证 (ticket-granting ticket, TGT) 开始 Kerberos 会话。此请求通常在登录时自动完成。

   要获取特定服务的其他票证，需要票证授予票证。票证授予票证类似于护照。与护照一样，票证授予票证可标识您的身份并允许您获取多个“签证”（票证），此处的“签证”不是出国所用，而是用于访问远程计算机或网络服务。与护照和签证一样，票证授予票证和其他各种票证具有有限的生命周期。区别在于基于 Kerberos 的命令会通知您拥有护照并为您取得签证。您不必亲自执行该事务。

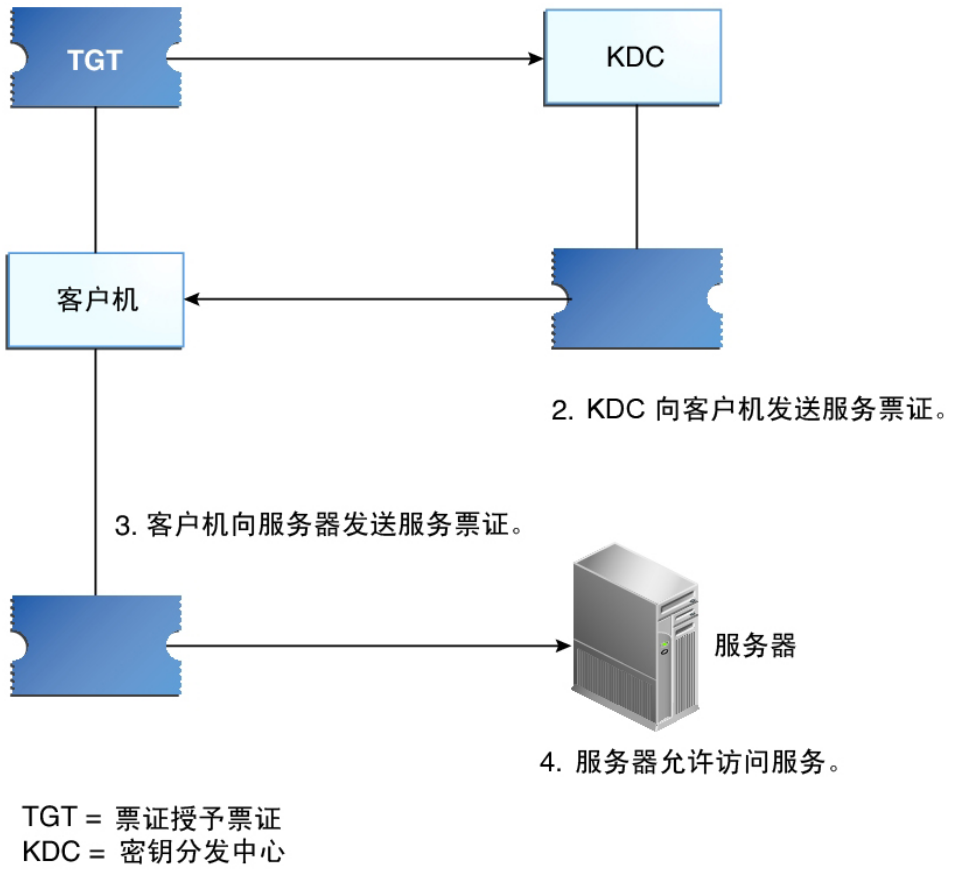
   与票证授予票证类似的另一种情况是可以在四个不同的滑雪场使用的三天滑雪入场卷。只要入场券未过期，您就可以在决定要去的任意一个滑雪场出示入场卷，并获取该滑雪场提供的缆车票。获取缆车票后，即可在该滑雪场随意滑雪。如果第二天去另一个滑雪场，您需要再次出示入场卷，并获取新滑雪场的另一张缆车票。区别在于基于 Kerberos 的命令会注意到您持有滑雪场的周末入场券并为您获取缆车票，这样您就不必亲自执行这些事务。
2. KDC 可创建票证授予票证，并采用加密形式将其发送回客户机。客户机使用其口令来解密票证授予票证。

3. 拥有有效的票证授予票证后，只要该票证授予票证未过期，客户机便可以请求所有类型的网络操作（如 `nfs` 或 `ssh`）的票证。此票证的有效期通常为几个小时。每次客户机执行唯一的网络操作时，都将从 KDC 请求该操作的票证。

后续 Kerberos 验证

客户机收到初始验证后，每个后续验证都按下图所示的模式进行。

图 2-2 使用 Kerberos 验证获取对服务的访问权



1. 客户机通过向 KDC 发送其票证授予票证作为身份证明，从 KDC 请求特定服务（例如，远程登录到另一台计算机）的票证。
2. KDC 将该特定服务的票证发送到客户机。

   假设用户 `jdoe` 要访问一个 NFS 文件系统，该文件系统已通过所要求的 `krb5` 验证共享。由于 `jdoe` 已经通过验证（也就是说，`jdoe` 已经拥有票证授予票证），因此当 `jdoe` 尝试访问文件时，NFS 客户机系统将自动以透明方式从 KDC 获取 NFS 服务的票证。要使用其他基于 Kerberos 的服务，`jdoe` 需要获取另一个票证，如步骤 1 中所述。
3. 客户机将票证发送到服务器。

   使用 NFS 服务时，NFS 客户机会自动透明地将 NFS 服务的票证发送到 NFS 服务器。
4. 服务器允许此客户机进行访问。

虽然这些步骤意味着服务器从不与 KDC 通信，但服务器确实会像第一个客户机那样向 KDC 注册自身。为了简洁起见，已省略该节。

批处理作业的 Kerberos 验证

批处理作业（例如，cron、at 和 batch）是延迟执行进程。在 Kerberos 环境中，包括延迟执行进程在内的所有进程都需要凭证。但是，用户凭证的有效期相对较短。缺省情况下，用户凭证的有效期为 8 小时，最多可将其更新到一周。这些时间设置用于限制敏感密钥暴露给恶意用户，但会导致无法在任意时间执行作业。

在 Oracle Solaris 中，访问 Kerberos 服务的批处理作业可在不曝光用户的长期密钥的情况下运行。此解决方案涉及将以下凭证信息存储到基于会话的用户凭证高速缓存：Kerberos 服务、用户名和客户机主机名。PAM 模块用于验证批处理作业。可以将主机能够获取其票证的服务集中存储到 LDAP 目录服务器。

有关更多信息，请参见 `pam_krb5_keytab(5)` 和 `pam_gss_s4u(5)` 手册页以及“为 Kerberos 服务访问配置延迟执行” [109]。

Kerberos、DNS 和命名服务

Kerberos 服务编译为使用 DNS 解析主机名。解析主机名时根本不会检查 `nsswitch` 服务。

Kerberos 组件

Kerberos 包含领域、网络程序、主体、服务器及其他组件。有关命令和模块的列表，请参见“Kerberos 实用程序” [46]。

Kerberos 网络程序

注 - 此 Oracle Solaris 发行版已弃用除 ssh 之外的所有远程登录命令。如有必要，可以使用以下已弃用的命令之一连接到旧版本系统。由于已弃用的命令用于传统脚本，因此，如果要在此系统上使用此类命令，您必须为该命令启用 SMF 服务，例如 svcadm enable login:rlogin。或者，您也可以修改脚本以使用 ssh 命令。同样地，要在旧版本系统中使用已弃用的命令连接到此系统，必须在此系统上启用该命令的服务。

已弃用的命令（例如 telnet）可能要求弱加密密钥。缺省情况下，不允许在 krb5.conf 文件中配置此类密钥。有关更多信息，请参见“Kerberos 中支持的加密类型” [56]和 krb5.conf(4) 手册页。

有关更多信息，请参见《在 Oracle Solaris 11.2 中确保系统和连接设备的安全 》中的“控制对计算机资源的访问”。另请参见“Kerberos 用户命令” [151]中列出的手册页。

用户可以使用的基于 Kerberos 的（即 "Kerberized"）命令包括：

- ftp
- rcp、rlogin、rsh
- ssh、scp、sftp
- telnet

这些应用程序与同名的 Oracle Solaris 应用程序相同。但是，它们已扩展为使用 Kerberos 主体来验证事务，因此会提供基于 Kerberos 的安全性。有关主体的信息，请参见“Kerberos 主体” [42]。

“Kerberos 用户命令” [151]中将进一步介绍这些命令。

Kerberos 主体

Kerberos 服务中的客户机由其主体标识。主体是 KDC 可以为其分配票证的唯一标识。主体可以是用户（如 jdoe）或服务（如 nfs）。

根据约定，主体名称分为三个部分：主名称、实例和领域。例如，jdoe/admin@CORP.EXAMPLE.COM 是一个典型的 Kerberos 主体。在本示例中：

- jdoe 是主名称。主名称可以是用户名（如此处所示）或服务（如 nfs）。主名称还可以是单词 host，这表示此主体是设置用于提供各种网络服务（如 ftp、scp 和 ssh 等）的服务主体。
- admin 是实例。对于用户主体，实例是可选的；但对于服务主体，实例则是必需的。例如，如果用户 jdoe 有时充当系统管理员，则主体 jdoe/admin 可将管理员与其用户身份区分开来。同样，如果 jdoe 在两台不同的主机上拥有帐户，这些帐户

可以使用具有不同实例的两个主体名称，例如 jdoe/denver.example.com 和 jdoe/boston.example.com。请注意，Kerberos 服务会将 jdoe 和 jdoe/admin 视为两个完全不同的主体。

对于服务主体，实例是全限定主机名。例如，bigmachine.corp.example.com 就是这种实例。此示例的主名称/实例可以为 ftp/bigmachine.corp.example.com 或 host/bigmachine.corp.example.com。

- CORP.EXAMPLE.COM 是 Kerberos 领域。领域将在“Kerberos 领域” [43]中介绍。

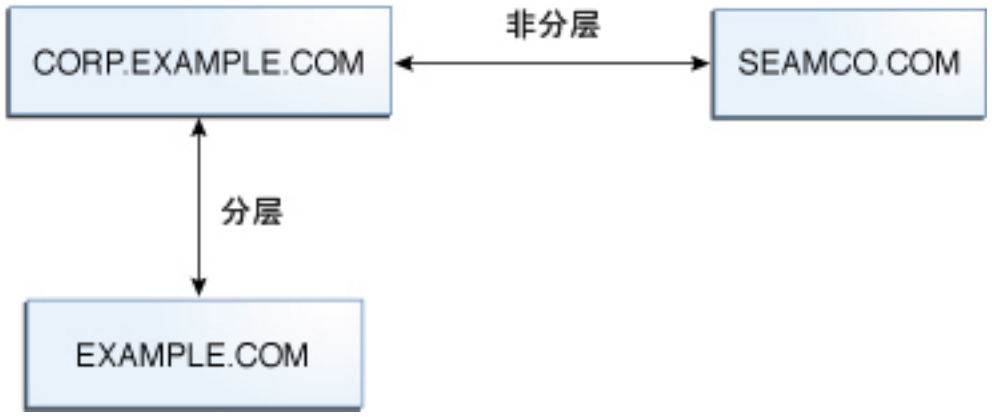
以下都是有效的主体名称：

- jdoe
- jdoe/admin
- jdoe/admin@CORP.EXAMPLE.COM
- nfs/host.corp.example.com@CORP.EXAMPLE.COM
- host/corp.example.com@CORP.EXAMPLE.COM

Kerberos 领域

领域是定义属于同一主 *KDC* 的一组系统的逻辑网络，类似于域。图 2-3 “Kerberos 领域” 显示了各领域相互之间的关系。有些领域是层次化的，其中，一个领域是另一个领域的超集。另外一些领域是非层次化（或“直接”）的，必须定义两个领域之间的映射。Kerberos 跨领域验证可在多个领域中提供验证。每个领域只需在其 KDC 中有对应于另一个领域的主体项即可。

图 2-3 Kerberos 领域



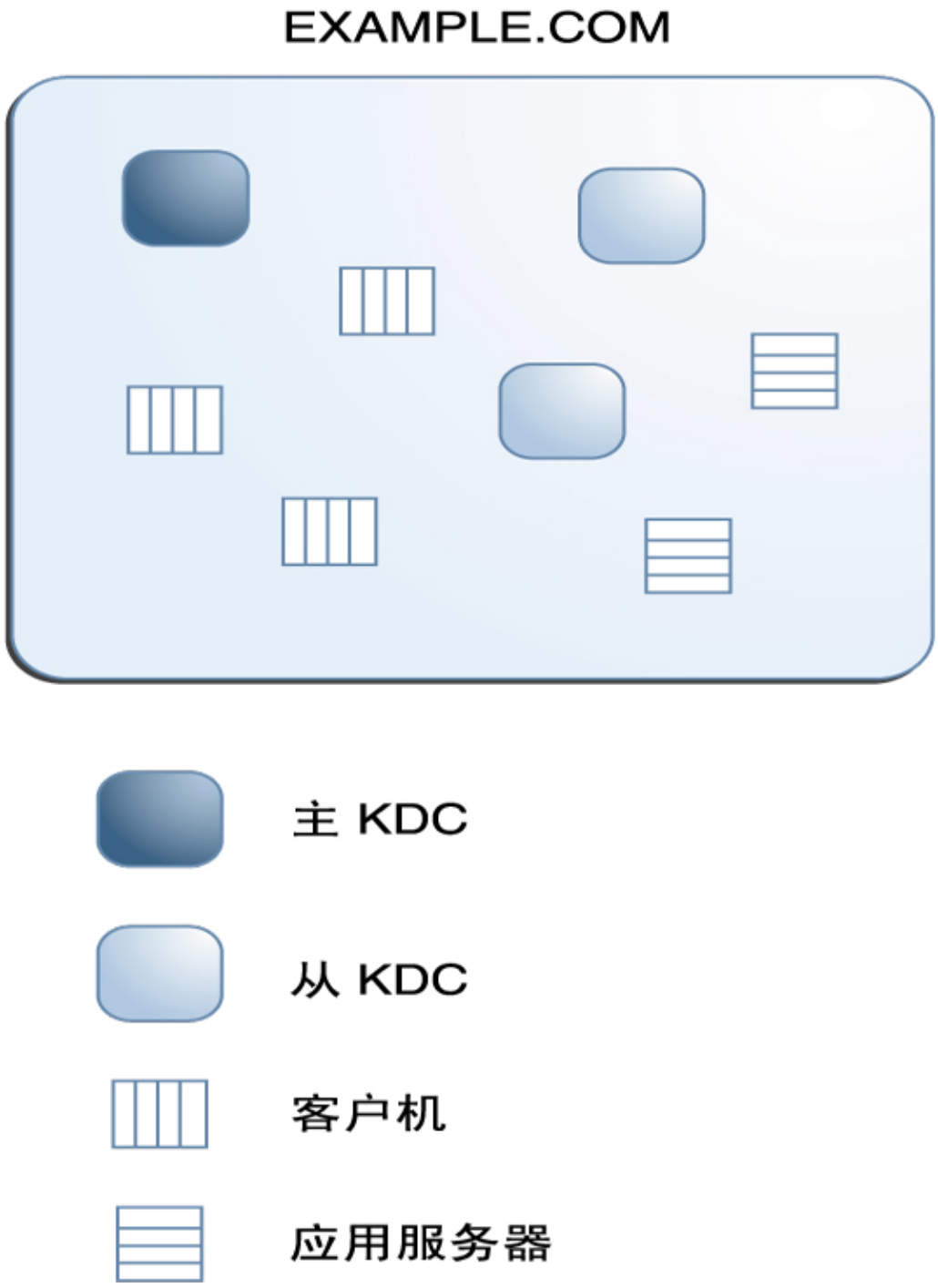
Kerberos 服务器

每个领域都必须包括一台用于维护主体数据库主副本的服务器。此服务器称为主 *KDC* 服务器。此外，每个领域还应至少包含一台从 *KDC* 服务器，该服务器包含主体数据库的副本。主 KDC 服务器和从 KDC 服务器都可创建用于建立验证的票证。

领域还可以包含 Kerberos 应用服务器。该服务器提供对基于 Kerberos 的服务（如 `ftp`、`ssh` 和 NFS）的访问。

下图显示了一个假设的领域可能包含的内容。

图 2-4 典型的 Kerberos 领域



Kerberos 实用程序

与 MIT 发布的 Kerberos V5 产品类似，Oracle Solaris 发行版的 Kerberos 服务也包括以下内容：

- 密钥分发中心 (Key Distribution Center, KDC)：
  - Kerberos 数据库管理守护进程 – `kadmind`。
  - Kerberos 票证处理守护进程 – `krb5kdc`。
  - 数据库管理程序 – `kadmin`（仅主 KDC）、`kadmin.local` 和 `kdb5_util`。
  - 数据库传播软件 – `kprop`（仅从 KDC）和 `kpropd`。
- 用于管理凭证的用户程序 – `kinit`、`klist` 和 `kdestroy`。
- 用于更改 Kerberos 口令的用户程序 – `kpasswd`。
- 网络应用程序 – `ftp`、`rcp`、`rlogin`、`rsh`、`scp`、`sftp`、`ssh` 和 `telnet`。
- 远程应用程序守护进程 – `ftpd`、`rlogind`、`rshd`、`sshd` 和 `telnetd`。
- 密钥表管理实用程序 – `ktutil`。
- 通用安全服务应用编程接口 (Generic Security Service Application Programming Interface, GSS-API) – 允许应用程序使用多种安全机制，并且不需要在每次添加新机制时重新编译应用程序。GSS-API 使用标准接口，这些接口允许将应用程序移植到多种操作系统上。使用 GSS-API，应用程序可包括完整性服务、保密性安全服务和验证。`ftp` 和 `ssh` 均使用 GSS-API。
- RPCSEC_GSS 应用编程接口 (Application Programming Interface, API) – 允许 NFS 服务使用 Kerberos 验证。RPCSEC_GSS API 可提供与要使用的机制无关的安全服务。RPCSEC_GSS 位于 GSS-API 层的顶部。使用 RPCSEC_GSS 的应用程序可以使用所有基于可插拔 GSS_API 的安全机制。

此外，Oracle Solaris 中的 Kerberos 服务包括：

- PAM 的 Kerberos V5 服务模块 – 为 Kerberos 服务提供验证、帐户管理、会话管理和口令管理。这些模块可使 Kerberos 验证对用户透明。
- Kerberos V5 基于用户的 PAM 栈 – 在 `/etc/security/pam_policy` 目录中提供适用于不同方案的 PAM 配置文件。
- 内核模块 – 提供 Kerberos 服务的基于内核的实现以供 NFS 服务使用，这可大大提高性能。
- Kerberos 管理 GUI (`gkadmin`) – 允许您在基于 Java™ 技术的 GUI 中管理主体和主体策略，以此作为 `kadmin` 命令的替代方法。

有关更多信息，请参见第 7 章 Kerberos 服务参考信息。

Kerberos 安全服务

除了提供安全的用户验证外，Kerberos 服务还会提供两种安全服务：

- **完整性** – 正如验证服务可确保网络中客户机的身份真实可靠一样，完整性服务可确保客户机发送的数据有效并且在传输过程中未被篡改。完整性服务通过数据的加密校验和计算来实现。完整性还包括用户验证。
- **保密性** – 保密性服务可进一步增强安全性。保密性服务不仅包括对所传输数据的完整性进行验证，还会在传输之前加密数据，防止数据遭到窃听。保密性服务也对用户进行验证。

Kerberos 加密类型

加密类型可标识执行加密操作时要使用的加密算法和模式。有关支持的加密类型的列表，请参见 krb5.conf(4) 和 kdb5_util(1M) 手册页。

客户机向 KDC 请求票证时，KDC 必须使用其加密类型与客户机和服务器都兼容的密钥。Kerberos 协议允许客户机请求 KDC 将特定加密类型用于该客户机的票证回复部分，但不允许服务器向 KDC 指定加密类型。

更改加密类型之前，请考虑以下问题：

- KDC 假定服务器支持与主体数据库中的服务器主体项关联的第一个密钥/加密类型。
- 在 KDC 上，必须确保为主体生成的密钥与对该主体进行验证的系统兼容。缺省情况下，`kadmin` 命令将为所有支持的加密类型创建密钥。如果使用主体的系统不支持该缺省加密类型集，则您在创建主体时应限制这些加密类型。限制加密类型的两种建议方法是：在 `kadmin addprinc` 中使用 `-e` 标志；将 `kdc.conf` 文件中的 `supported_enctypes` 参数设置为该子集。如果 Kerberos 领域中的大多数系统都支持缺省加密类型集的子集，则应使用 `supported_enctypes` 参数。如果设置 `supported_enctypes`，则将指定 `kadmin addprinc` 在为特定领域创建主体时使用的缺省加密类型集。
- 确定系统支持的加密类型时，请考虑系统中运行的 Kerberos 版本，以及为其创建服务器主体的服务器应用程序所支持的加密算法。例如，创建 `nfs/hostname` 服务主体时，应将加密类型限制为该主机上的 NFS 服务器支持的类型。
- `kdc.conf` 文件中的 `master_key_enctype` 参数可用于控制加密主体数据库中各项的主密钥的加密类型。如果已创建 KDC 主体数据库，请勿使用此参数。可在创建数据库时使用 `master_key_enctype` 参数，以便将缺省主密钥加密类型从 `aes256-cts-hmac-sha1-96` 更改为其他加密类型。配置从 KDC 时，请确保所有从 KDC 都支持所选加密类型，并且其 `kdc.conf` 文件都包含完全相同的 `master_key_enctype` 项。另外，如果在 `kdc.conf` 中设置了 `supported_enctypes`，则还应确保将 `master_key_enctype` 设置为 `supported_enctypes` 中的加密类型之一。如果未正确处理其中任一问题，则主 KDC 可能无法使用从 KDC。
- 在客户机上，可以通过 `krb5.conf` 文件中的参数控制客户机从 KDC 请求的加密类型。`default_tkt_enctypes` 参数用于指定客户机从 KDC 请求票证授予票证 (Ticket Granting Ticket, TGT) 时要使用的加密类型。客户机可使用 TGT 以一种更有效的方式获取其他服务器票证。如果客户机使用 TGT 请求服务器票证（称为 TGS 请求），则设置 `default_tkt_enctypes` 将提供客户机对加密类型（用于保护客户机和

KDC 之间的通信）的部分控制。请注意，`default_tkt_enctypes` 中指定的加密类型必须至少与 KDC 上存储的主体数据库中的一种主体密钥加密类型匹配。否则，TGT 请求将会失败。在大多数情况下，最好不要设置 `default_tkt_enctypes`，因为此参数可能会导致互操作性问题。缺省情况下，客户机代码会请求所有受支持的加密类型，而 KDC 会基于在主体数据库中找到的密钥来选择加密类型。

- `default_tgs_enctypes` 参数可限制客户机在其 TGS 请求（用于获取服务器票证）中请求的加密类型。另外，此参数还可限制 KDC 在创建客户机和服务器共享的会话密钥时使用的加密类型。例如，如果客户机要在执行安全 NFS 时仅使用 3DES 加密，则应将 `default_tgs_enctypes` 设置为 `des3-cbc-sha1`。请确保客户机主体和服务器主体在主体数据库中具有 `des-3-cbc-sha1` 密钥。与 `default_tkt_enctype` 一样，在大多数情况下，最好不要设置此参数，因为当 KDC 和服务器上均未正确设置凭证时，此参数可能会导致互操作性问题。
- 在服务器上，可使用 `kdc.conf` 文件中的 `permitted_enctypes` 参数来控制该服务器可接受的加密类型。此外，还可以指定该服务器在创建密钥表项时使用的加密类型。应避免使用上述任一方法来控制加密类型，而由 KDC 确定要使用的加密类型，因为 KDC 不会与服务器应用程序进行通信来确定要使用的密钥或加密类型。

FIPS 140 算法和 Kerberos 加密类型

可以在 Oracle Solaris 中配置 Kerberos 以 FIPS 140 模式运行。如果领域中包含不符合 FIPS 140 的传统应用程序或系统，则该领域无法以 FIPS 140 模式运行。

以 FIPS 140 模式运行时，Kerberos 成为 FIPS 140 提供者的使用者。在 Oracle Solaris 中，提供者为加密框架。加密框架唯一通过 FIPS 140 验证的 Kerberos 加密类型是 `des3-cbc-sha1`。该值不是缺省值。有关说明，请参见如何配置 Kerberos 以 FIPS 140 模式运行 [64]。

注 - 如果有严格的要求，只能使用 FIPS 140-2 验证的加密，则必须运行 Oracle Solaris 11.1 SRU 5.5 发行版或 Oracle Solaris 11.1 SRU 3 发行版。Oracle 已在这两个特定发行版中针对加密框架完成了 FIPS 140-2 验证。Oracle Solaris 11.2 基于此验证的基础而构建并在性能、功能和可靠性方面进行了软件改进。应当尽可能在 FIPS 140-2 模式下配置 Oracle Solaris 11.2，以利用这些改进。

Kerberos 凭证如何提供对服务的访问

如果能够提供证明身份的票证和匹配的会话密钥，则允许访问远程服务。会话密钥包含特定于用户以及要访问的服务的信息。所有用户首次登录时，KDC 都会为其创建票证和会话密钥。票证和匹配的会话密钥即组成了凭证。使用多个网络服务时，用户可以收集许多凭证。对于在特定服务器上运行的每个服务，用户都需要拥有一个凭证。例如，访

问名为 boston 的服务器上的 nfs 服务需要一个凭证。访问其他服务器上的 nfs 服务需要另外的凭证。

要访问特定服务器上的特定服务，用户必须获取两个凭证。第一个凭证用于票证授予票证 (Ticket Granting Ticket, TGT)。票证授予服务解密此凭证后，该服务即可为用户请求访问的服务器创建第二个凭证。然后，可使用第二个凭证来请求访问该服务器中的相应服务。该服务器成功解密第二个凭证后，便会授予用户访问权限。

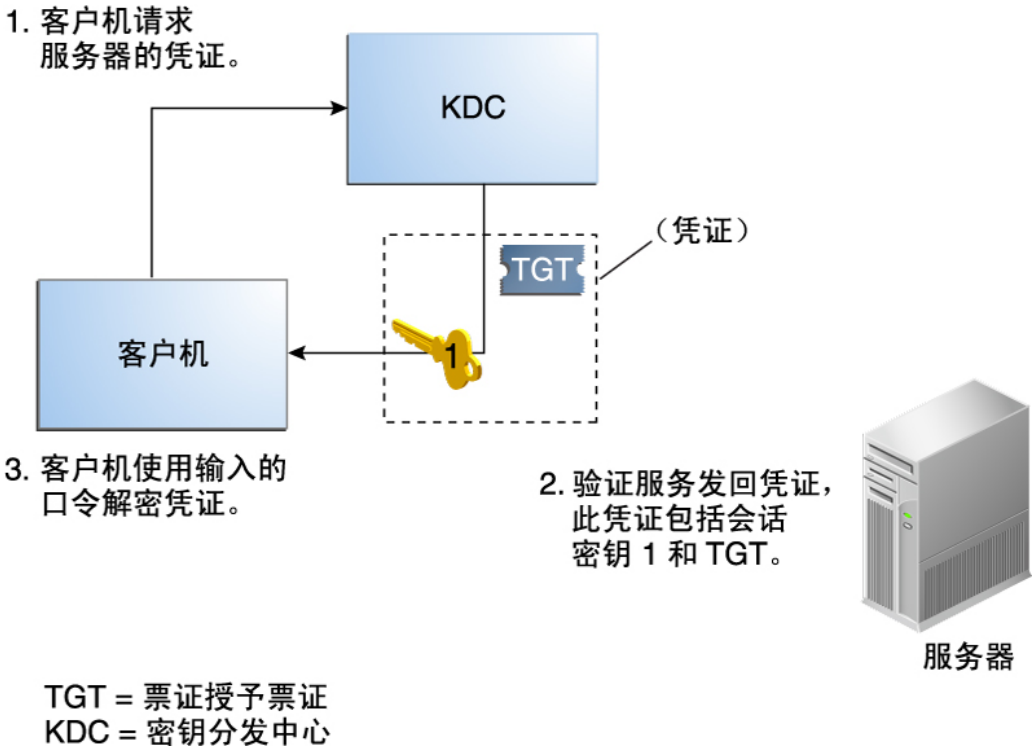
创建和存储凭证的过程是透明的。凭证是由将凭证发送到请求程序的 KDC 创建的。收到凭证后，会将其存储在凭证高速缓存中。

获取用于票证授予服务的凭证

1. 要启动验证过程，客户机需要向验证服务器发送针对特定用户主体的验证请求。该请求在发送时未加密。由于请求中不包含安全信息，因此无需使用加密。
2. 验证服务收到请求后，将在 KDC 数据库中查找该用户的主体名称。如果主体与数据库中的项匹配，验证服务会获取该主体的私钥。然后，验证服务将生成一个供客户机和票证授予服务使用的会话密钥（称为会话密钥 1），以及一个用于票证授予服务的票证（称为票证 1）。此票证也称作票证授予票证 (ticket-granting ticket, TGT)。会话密钥和票证均使用该用户的私钥进行加密，并且会将信息发回客户机。
3. 客户机通过使用该用户主体的私钥，借助此信息对会话密钥 1 和票证 1 进行解密。由于该私钥仅对此用户和 KDC 数据库公开，因此包中的信息应该是安全的。客户机将该信息存储在凭证高速缓存中。

在此过程中，通常会提示用户输入口令。如果用户指定的口令与用于生成存储在 KDC 数据库中的私钥的口令相同，则客户机可以成功解密验证服务发送的信息。这样，客户机便拥有了用于票证授予服务的凭证，并已准备好请求获取某个服务器的凭证。

图 2-5 获取用于票证授予服务的凭证

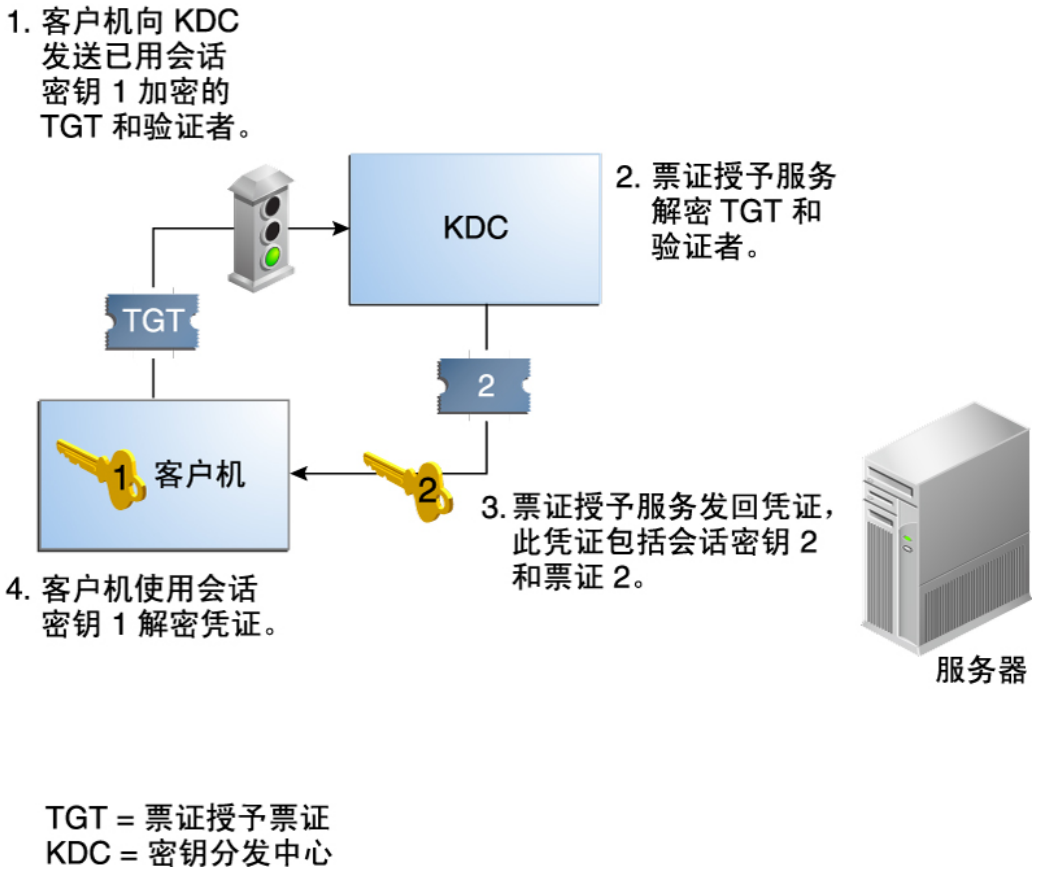


获取基于 Kerberos 的服务器的凭证

1. 要请求对特定服务器的访问权限，客户机首先必须向验证服务获取该服务器凭证。请参见“获取用于票证授予服务的凭证” [49]。客户机随后会向票证授予服务发送请求，其中包括服务主体名称、票证 1 和以会话密钥 1 加密的验证者。票证 1 最初是使用票证授予服务的服务密钥，由验证服务进行加密的。
2. 由于票证授予服务的服务密钥对票证授予服务公开，因此可以对票证 1 进行解密。票证 1 中的信息包括会话密钥 1，这样，票证授予服务可以解密验证者。在此情况下，由票证授予服务对用户主体进行验证。
3. 一旦验证成功，票证授予服务会为用户主体和服务器生成会话密钥（称为会话密钥 2），为服务器生成票证（称为票证 2）。然后，会使用会话密钥 1 对会话密钥 2 和票证 2 进行加密。因为会话密钥 1 仅为客户机和票证授予服务所知，因此该信息是安全的，且可以通过网络安全发送。

4. 客户机收到该信息包后，会使用会话密钥 1 解密信息，该密钥储存在凭证高速缓存中。客户机即获取用于服务器的凭证。现在，客户机可以请求访问该服务器中的特定服务。

图 2-6 获取服务器凭证

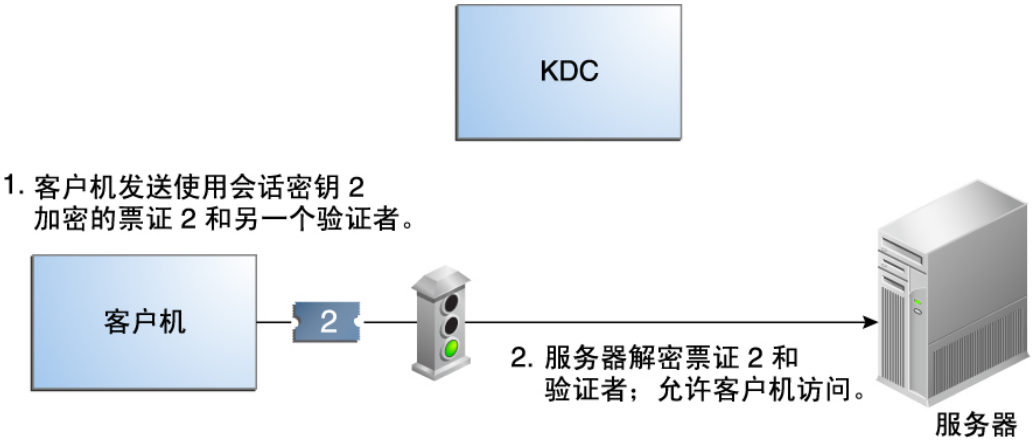


获取对特定 Kerberos 服务的访问权限

1. 要请求对特定服务的访问权限，客户机必须首先从验证服务器获取用于票证授予服务的凭证，然后从票证授予服务获取服务器凭证。请参见“获取用于票证授予服务的凭证” [49]和“获取基于 Kerberos 的服务器的凭证” [50]。然后，客户机可将包含票证 2 和另一个验证者的请求发送到该服务器。验证者使用会话密钥 2 加密。

2. 票证 2 是由票证授予服务使用该服务的服务密钥进行加密的。由于服务密钥对服务主体公开，因此该服务可以解密票证 2 并获取会话密钥 2。然后，可使用会话密钥 2 解密验证者。如果成功解密验证者，则可授予客户机对该服务的访问权限。

图 2-7 获取对特定服务的访问权限



Oracle Solaris Kerberos 和 MIT Kerberos 之间的显著差异

Oracle Solaris 包含 MIT Kerberos 所不具备的以下增强功能：

- Oracle Solaris 远程应用程序的 Kerberos 支持
- KDC 数据库的增量传播
- 客户机配置脚本
- 本地化的错误消息
- Oracle Solaris 审计记录支持
- 通过使用 GSS-API 实现 Kerberos 安全使用线程
- 使用加密框架进行加密

♦ ♦ ♦ 第 3 章

规划 Kerberos 服务

本章介绍管理员在配置或部署 Kerberos 服务之前必须确定的一些安装和配置选项：

规划 Kerberos 部署

安装 Kerberos 服务之前，必须先解决几个配置问题。虽然可以在初始安装后更改配置，但有些更改可能会难以实现。此外，有些更改需要重建 KDC，所以应在部署 Kerberos 前考虑长期目标。

部署 Kerberos 基础结构涉及到安装 KDC、为主机创建密钥和迁移用户等任务。重新配置 Kerberos 部署会像执行初始部署一样难，因此，请仔细规划部署，以避免将来不得不重新配置。

规划 Kerberos 领域

领域是类似于域的逻辑网络，用于定义相同主 KDC 下的一组系统。与建立 DNS 域名一样，必须在配置 Kerberos 服务之前解决领域名称、领域数和每个领域的大小，以及各领域之间的关系（用于跨领域验证）等相关问题。

Kerberos 领域名称

领域名称可以包括任意 ASCII 字符串。通常，领域名称与 DNS 域名相同，但领域名称采用大写。使用此约定可帮助您区分 Kerberos 服务发生的问题与 DNS 名称空间发生的

问题，同时仍使用熟悉的名称。可以使用任何字符串，但这样可能会增加配置和维护任务的工作量。因此，请使用符合标准 Internet 命名结构的领域名称。

Kerberos 领域数量

安装所需的领域数量由以下因素决定：

- 支持的客户机数量。如果一个领域内客户机过多，管理起来会非常困难，最终仍然需要您分割领域。确定可以支持的客户机数量的主要因素如下：
  - 每台客户机产生的 Kerberos 流量
  - 物理网络带宽
  - 主机速度

  因为每种安装都有不同的限制，所以没有任何规则可以确定客户机的最大数量。
- 客户机间的距离。如果客户机分别位于不同的地理区域，设置若干个小领域可能更有效果。
- 可作为 KDC 安装的主机数量。规划时，应为每个领域创建至少两个 KDC 服务器（一个主服务器以及至少一个从服务器）。

建议将 Kerberos 领域与管理域结合使用。请注意，Kerberos V 领域可以跨其对应的 DNS 域的多个子域。

Kerberos 领域分层结构

为进行跨领域验证而配置多个领域时，需要决定如何将这些领域捆绑在一起。可以在这些领域之间建立层次化关系，以便提供指向相关域的自动路径。如果分层结构链中的所有领域均配置正确，这些自动路径将能减轻管理工作。但是，如果存在许多域层次，您可能不希望使用自动路径，因为这涉及到太多的事务。

您也可以选择直接建立信任关系。当两个层次化领域之间存在的层次太多或不存在任何层次化关系时，直接信任关系很有用。由于必须在所有主机上的 `/etc/krb5/krb5.conf` 文件中定义所使用的连接，因此还需要执行一些其他工作。直接信任关系又称为可传递关系。有关图示，请参见“Kerberos 领域” [43]。有关配置过程，请参见“配置跨领域验证” [110]。

将主机名映射到 Kerberos 领域

主机名到领域名称的映射在 `krb5.conf` 文件的 `domain_realm` 部分中定义。可以根据需要对整个域和单个主机定义这些映射。

也可以使用 DNS 来查找有关 KDC 的信息。通过使用 DNS，您将能更轻松地更改信息，因为不需要更改所有 Kerberos 客户机上的 `krb5.conf` 文件以应用每项更改。

注 - 本指南中的过程假定使用 DNS。

有关更多信息，请参见 `krb5.conf(4)` 手册页。

Oracle Solaris 中的 Kerberos 客户机可与 Active Directory 服务器良好地互操作。可以将 Active Directory 服务器配置为提供领域到主机的映射。

Kerberos 客户机与服务主体名称

Oracle Solaris 中的 Kerberos 不使用 `name-service/switch` 服务。Kerberos 服务使用 DNS 来解析主机名。因此，必须在所有主机上启用 DNS。使用 DNS 时，主体必须包含每台主机的全限定域名 (Fully Qualified Domain Name, FQDN)。例如，如果主机名是 `boston`，DNS 域名是 `example.com`，领域名称是 `EXAMPLE.COM`，那么主机的主体名称将是 `host/boston.example.com@EXAMPLE.COM`。本指南中的示例假定已配置 DNS 并将 FQDN 用于每台主机。

在为关联服务构建服务主体时，Kerberos 服务将通过 DNS 标准化主机别名，并使用标准化的格式 (cname)。因此，创建服务主体时，服务主体名称的主机名部分采用提供服务的系统的主机名标准格式。

以下示例显示了 Kerberos 服务如何将主机名标准化。如果用户运行 `ssh alpha.example.com` 命令（其中 `alpha.example.com` 是 cname `beta.example.com` 的 DNS 主机别名），Kerberos 服务会将 `alpha.example.com` 标准化为 `beta.example.com`。KDC 将票证处理为服务主体 *host*`/beta.example.com` 的请求。

对于包括主机 FQDN 的主体名称，务必使其与 `/etc/resolv.conf` 文件中描述 DNS 域名的字符串匹配。为主体指定 FQDN 时，Kerberos 服务要求 DNS 域名使用小写字母。DNS 域名可以包括大写和小写字母，但是创建 host 主体时只能使用小写字母。例如，DNS 域名可以是 `example.com`、`Example.COM` 或其他任何变体。主机的主体名称仍将是 `host/boston.example.com@EXAMPLE.COM`。

此外，还配置了服务管理工具 (Service Management Facility, SMF)，以便未运行 DNS 客户机服务时，不会启动许多守护进程或命令。`kdb5_util`、`kadmind` 和 `kpropd` 守护进程，以及 `kprop` 命令都配置为依赖于 DNS 服务。要充分利用 Kerberos 服务和 SMF 提供的功能，必须在所有主机上启用 DNS 客户机服务。

Kerberos 领域内的时钟同步

所有参与 Kerberos 验证系统的主机都必须使其内部时钟的时间偏差保持在指定的最大时间内。该功能称为时钟相位差，它提供另一种 Kerberos 安全检查方法。如果任意两台参与主机之间的时间偏差超过了时钟相位差，则请求会被拒绝。

一种同步所有时钟的方法是使用网络时间协议 (Network Time Protocol, NTP) 软件。有关更多信息，请参见“同步 KDC 与 Kerberos 客户机的时钟” [112]。也可以使用其他方法同步时钟，但必须采用某种同步形式。

Kerberos 中支持的加密类型

加密类型是指定 Kerberos 服务中使用的加密算法、加密模式和散列算法的标识符。Kerberos 服务中的密钥具有关联的加密类型，用于指定服务使用该密钥执行加密操作时所用的加密算法和模式。有关支持的加密类型的列表，请参见 `krb5.conf(4)` 和 `kdb5_util(1M)` 手册页。

如果要更改加密类型，可在创建新的主体数据库时进行更改。由于 KDC（服务器）和客户机之间存在交互，因此很难在现有数据库上更改加密类型。有关更多信息，请参见“Kerberos 加密类型” [47]。

缺省情况下不允许使用弱加密类型（例如 `des`）。如果需要使用弱加密类型以实现向后兼容性或互操作性，请将 `/etc/krb5/krb5.conf` 文件的 `libdefaults` 部分中的 `allow_weak_crypto` 项设置为 `true`。

规划 KDC

KDC 使用特定端口，需要额外的服务器来处理增加的票证负载，而且需要利用传播技术来使服务器保持同步。此外，加密类型集中进行管理。对 KDC 进行初始配置时，有若干选项可供选择。

KDC 端口和管理服务端口

缺省情况下，端口 `88` 和端口 `750` 供 KDC 使用，端口 `749` 供 KDC 管理守护进程使用。可以使用不同的端口号。但是，如果更改了端口号，则必须更改每台客户机上的 `/etc/services` 和 `/etc/krb5/krb5.conf` 文件。此外，还必须更新每个 KDC 上的 `/etc/krb5/kdc.conf` 文件。

从 KDC 数量

与主 KDC 一样，从 KDC 也能为客户机生成凭证。如果主 KDC 不可用，从 KDC 将会提供备份。规划时，应为每个领域创建至少一个从 KDC。

可能需要更多从 KDC，视以下因素而定：

- 领域中的物理段数。通常，应至少将网络设置为领域中其他组件不可用情况下，每个段仍可工作。为此，必须能够从每个段访问 KDC。该实例中的 KDC 可以是主 KDC 或从 KDC。
- 领域中的客户机数量。添加更多的从 KDC 服务器可以减少当前服务器的负荷。

应避免添加太多从 KDC。由于 KDC 数据库必须传播到每台服务器，因此，安装的 KDC 服务器越多，更新整个领域中的数据所用的时间就越长。另外，因为每个从 KDC 都会保存 KDC 数据库的副本，所以增加从 KDC 会增大安全违规的风险。

应配置一个或多个从 KDC 与主 KDC 交换。以这种方式配置至少一个从 KDC 的好处是：一旦主 KDC 由于任何原因而出现故障，可以将预先配置的系统变为主 KDC。有关说明，请参见“交换主 KDC 服务器与从 KDC 服务器” [114]。

Kerberos 数据库传播

存储在主 KDC 上的数据库必须定期传播到从 KDC。可以将数据库传播配置为以增量方式进行。增量过程仅将更新的信息（而不是整个数据库）传播到从 KDC。有关数据库传播的信息，请参见“管理 Kerberos 数据库” [118]。

如果不使用增量传播，要解决的首要问题之一是，多久更新一次从 KDC。必须根据完成更新所需的时间，权衡是否需要将最新信息传播到所有客户机。

在一个领域包含多个 KDC 的大型安装中，一个或多个从 KDC 可以传播数据，以便以并行方式完成该过程。此策略可减少更新所用的时间，但同时也会提高管理工作的复杂度。有关更多信息，请参见“为 Kerberos 设置并行传播” [126]。

KDC 配置选项

有几种方法可用于配置 KDC。最简单的方法是使用 `kdcmgr` 实用程序以自动或交互方式配置 KDC。自动方式要求使用命令行选项来定义配置参数。该方法尤其适用于脚本。交互方式会提示您提供所有所需信息。有关此命令的使用说明链接，请参见“配置 KDC 服务器” [63]。

也可以使用 LDAP 来管理 Kerberos 的数据库文件。有关说明，请参见如何将主 KDC 配置为使用 LDAP 目录服务器 [76]。如果站点需要协调 Kerberos 数据库及其现有目录服务器设置，LDAP 可简化此类站点的管理工作。

规划 Kerberos 客户机

可以使用以下方法安装 Kerberos 客户机：自动安装、从脚本安装或通过编辑配置文件手动配置。对于受保护的网络登录，PAM 框架提供了 `pam_ldap` 模块。请参见 `pam_ldap(5)` 手册页。此外，当客户机请求某个服务时，可由主 KDC 之外的其他服务器授予该服务。

规划 Kerberos 客户机自动安装

使用 Oracle Solaris 的自动化安装程序 (Automated Installer, AI) 功能可以快速而轻松地配置 Kerberos 客户机。AI 服务器管理员可创建 Kerberos 配置文件并将其分配给 AI 客户机。此外，AI 服务器还会提供客户机密钥。因此，在安装时 Kerberos 客户机是一个能够托管安全服务的完全置备的 Kerberos 系统。使用自动化安装程序可降低系统管理和维护成本。

可以使用 `kclient` 命令为任意 KDC 类型的客户机创建自动化安装。

- 对于不属于 Oracle Solaris KDC 的客户机，可以使用 AI。有关 KDC 供应商的列表，请参见 `kclient(1M)` 手册页。

  所有 KDC 类型都支持传输预生成的密钥表。Oracle Solaris KDC 和 MS AD 还支持自动注册。
- 可以运行 `kclient` 命令为 AI 创建 Kerberos 配置文件。有关更多信息，请参见 `kclient(1M)` 手册页和“Kerberos 客户机配置选项” [58]。有关通过 AI 配置 Kerberos 客户机的说明，请参见《安装 Oracle Solaris 11.2 系统》中的“如何使用 AI 配置 Kerberos 客户机”。

Kerberos 客户机配置选项

可以通过使用 `kclient` 配置实用程序或手动编辑文件来配置 Kerberos 客户机。该实用程序既可以在交互模式下运行，也可以在非交互模式下运行。在交互模式下，该实用程序将提示用户提供特定于 Kerberos 的参数值，以便能够在配置客户机时进行更改。在非交互模式下，用户需要提供包含参数值的文件。此外，在非交互模式下可以添加命令行选项。由于交互模式和非交互模式所需的步骤都比手动配置少，因此速度更快，也更不容易出错。

如果已启用以下设置，则无需对 Kerberos 客户机执行显式配置：

- DNS 配置为返回 KDC 的 SRV 记录。
- 领域名称与 DNS 域名匹配，或者 KDC 支持引用。
- Kerberos 客户机不要求使用与 KDC 服务器不同的密钥。

由于以下原因，您可能仍希望对 Kerberos 客户机执行显式配置：

- 零配置过程所执行的 DNS 查找次数比直接配置的客户机多，因此其效率比直接配置低。
- 如果不使用引用，零配置逻辑会根据主机的 DNS 域名来确定领域。这种配置会带来一定的安全风险，但风险比启用 `dns_lookup_realm` 要小得多。
- `pam_krb5` 模块依赖于 keytab file（密钥表文件）中的主机密钥项。虽然可以在 `krb5.conf` 文件中禁用这项要求，但出于安全原因，不建议这样做。有关更多信息，请参见“Kerberos 客户机登录安全性” [59]和 `krb5.conf(4)` 手册页。

有关客户机配置的完整说明，请参见“配置 Kerberos 客户机” [84]。

Kerberos 客户机登录安全性

登录后，客户机会使用 `pam_krb5` 模块验证颁发最新 TGT 的 KDC 是否与颁发客户机 host 主体（存储在 `/etc/krb5/krb5.keytab` 文件中）的 KDC 相同。在验证栈中配置 `pam_krb5` 模块后，该模块即可验证 KDC。对于有些配置，例如不存储客户机 host 主体的 DHCP 客户机，则需要禁用该项检查。要关闭该项检查，必须将 `krb5.conf` 文件中的 `verify_ap_req_nofail` 选项设置为 `false`。有关更多信息，请参见“禁用票证授予票证的验证” [95]。

Kerberos 中的可信委托服务

对于某些应用程序，客户机可能需要将授权委托给代其联系其他服务的某个服务器。客户机必须将凭证转发给中间服务器。客户机获取服务器服务票证的功能不会向客户机传递是否可以信任服务器接受委托凭证的信息。`kadmin` 命令的 `ok_to_auth_as_delegate` 选项为 KDC 将本地领域策略传递给客户机提供了一种方法，该策略与是否应信任中间服务器接受此凭证有关。

如果设置了 `ok_to_auth_as_delegate` 选项，客户机 KDC 回复中的加密部分可以包含凭证票证标志的副本。客户机可以使用该设置来确定是否将凭证委托（通过授予代理或转发 TGT）给该服务器。设置该选项时，应考虑运行服务的服务器的安全性和位置，以及服务是否要求使用委托凭证。

规划 Kerberos 使用 UNIX 名称和凭证的方式

对于需要 GSS 凭证名称到 UNIX 用户 ID (user ID, UID) 的映射的 GSS 应用程序（例如 NFS），Kerberos 服务提供了缺省映射。使用 Kerberos 服务时，GSS 凭证名称相当于 Kerberos 主体名称。此外，可以使用 PAM 框架自动迁移在缺省 Kerberos 领域中没有有效用户帐户的 UNIX 用户。

将 GSS 凭证映射到 UNIX 凭证

缺省映射算法使用 Kerberos 主体的主名称来查找 UID。查找将在缺省领域中进行，或者在 `/etc/krb5/krb5.conf` 文件的 `auth_to_local_realm` 参数所允许的任何领域中进行。例如，可以使用口令表将用户主体名称 `jdoe@EXAMPLE.COM` 映射到名为 `jdoe` 的 UNIX 用户的 UID。不会映射用户主体名称 `jdoe/admin@EXAMPLE.COM`，因为该主体名称包括 `admin` 实例部分。

如果用户凭证的缺省映射满足要求，则无需填充 GSS 凭证表。如果缺省映射不满足要求（例如，如果要映射包含实例部分的主体名称），则需要使用其他方法。有关更多信息，请参见以下内容：

- 如何创建和修改凭证表 [106]
- 如何提供各领域之间的凭证映射 [106]
- “观察从 GSS 凭证到 UNIX 凭证的映射” [175]

`gsscred` 表

如果缺省映射不满足要求，则 NFS 服务器尝试标识 Kerberos 用户时，将使用 `gsscred` 表。NFS 服务使用 UNIX UID 来标识用户。这些 ID 不属于用户主体或凭证。`gsscred` 表提供从 GSS 凭证到 UNIX UID 的其他映射（通过口令文件）。填充 KDC 数据库后，必须创建和管理该表。

接收到客户机请求时，NFS 服务便会尝试将凭证名称映射到 UNIX UID。如果映射失败，会检查 `gsscred` 表。

自动将用户迁移到 Kerberos 领域

可以使用 PAM 框架自动迁移在缺省 Kerberos 领域中没有有效用户帐户的 UNIX 用户。具体而言，可将 `pam_krb5_migrate.so` 模块添加到 PAM 服务的验证栈。这样服务就配置为按如下方式工作：只要没有 Kerberos 主体的用户使用口令成功登录到系统，便会自动为该用户创建 Kerberos 主体。因此，新的主体口令与 UNIX 口令相同。有关使用 `pam_krb5_migrate.so` 模块的说明，请参见如何在 Kerberos 领域中配置用户自动迁移 [96]。

♦ ♦ ♦ 第 4 章

4 配置 Kerberos 服务

本章介绍 KDC 服务器、网络应用服务器、NFS 服务器和 Kerberos 客户机的配置过程。其中许多过程都要求 root 访问权限，因此应由系统管理员或高级用户来执行。本章还将介绍跨领域配置过程以及与 KDC 服务器相关的其他主题。

本章包含以下主题：

配置 Kerberos 服务

由于配置期间的部分过程依赖于其他过程，因此必须按特定顺序执行这些过程。这些过程通常用于建立使用 Kerberos 服务所需的服务。其他过程与顺序无关，可以在适当的情况下执行。以下任务列表给出了 Kerberos 安装的建议顺序。

注 - 这些章节中的示例使用缺省加密类型，不是 Oracle Solaris 通过 FIPS 140 验证的类型。要以 FIPS 140 模式运行，必须将数据库、服务器和客户机通信的加密类型限定为 des3-cbc-sha1 加密类型。在创建 KDC 之前，请编辑如何配置 Kerberos 以 FIPS 140 模式运行 [64]中所述的文件。

表 4-1 配置 Kerberos 服务任务列表

任务	说明	有关说明
1. 规划 Kerberos 安装。	开始软件配置过程之前，请先解决配置问题。提前规划可节省以后所用的时间和其他资源。	第 3 章 规划 Kerberos 服务
2. 配置 KDC 服务器。	为领域配置并构建主 KDC 服务器和从 KDC 服务器，以及 KDC 数据库。	“配置 KDC 服务器” [63]
2a.（可选）配置 Kerberos 以 FIPS 140 模式运行。	设置为只使用 FIPS 140 验证算法。	如何配置 Kerberos 以 FIPS 140 模式运行 [64]
2b.（可选）将 Kerberos 配置为基于 LDAP 运行。	将 KDC 配置为使用 LDAP 目录服务器。	“在 LDAP 目录服务器上管理 KDC” [82]
3. 安装网络时间协议 (Network Time Protocol, NTP) 软件。	创建为网络上所有系统提供时间的中央时钟。	“同步 KDC 与 Kerberos 客户机的时钟” [112]
4.（可选）配置可交换 KDC 服务器。	使交换主 KDC 服务器与从 KDC 服务器的任务更轻松。	如何配置可交换的从 KDC 服务器 [114]
4.（可选）增强 KDC 服务器的安全性。	防止 KDC 服务器中发生安全违规。	“限制对 KDC 服务器的访问” [130]

配置其他 Kerberos 服务

完成必需的步骤后，可以在适当的情况下执行以下过程。

表 4-2 配置其他 Kerberos 服务任务列表

任务	说明	有关说明
配置跨领域验证。	启用领域之间的通信。	“配置跨领域验证” [110]
配置 Kerberos 应用服务器。	使服务器支持使用 Kerberos 验证的服务，如 `ftp`。	“配置 Kerberos 网络应用服务器” [101]
配置延迟执行服务。	使 `cron` 主机在任意时间执行任务。	“为 Kerberos 服务访问配置延迟执行” [109]
配置 Kerberos NFS 服务器。	使服务器共享要求 Kerberos 验证的文件系统。	“配置 Kerberos NFS 服务器” [104]
配置 Kerberos 客户机。	使客户机使用 Kerberos 服务。	“配置 Kerberos 客户机” [84]
同步时钟以帮助验证。	配置时钟同步协议。	“同步 KDC 与 Kerberos 客户机的时钟” [112]
管理 Kerberos 数据库。	维护 Kerberos 数据库。	“管理 Kerberos 数据库” [118]
管理 Kerberos 数据库的存储文件。	处理 Kerberos 数据库的密钥。	“管理 Kerberos 数据库的存储文件” [127]

配置 KDC 服务器

安装 Kerberos 软件之后，必须配置密钥分发中心 (Key Distribution Center, KDC) 服务器。配置主 KDC 服务器和至少一个从 KDC 服务器来提供颁发凭证的服务。这些凭证是 Kerberos 服务的基础，因此必须先配置 KDC 然后再尝试执行其他任务。

主 KDC 服务器与从 KDC 服务器之间最大的差别是，只有主 KDC 服务器可以处理数据库管理请求。例如，更改口令或添加新主体必须在主 KDC 服务器上完成。然后可以将这些更改传播到从 KDC 服务器。从 KDC 服务器和主 KDC 服务器都可以生成凭证。从 KDC 服务器可在主 KDC 服务器无法响应时提供冗余性。

可以选择通过以下多种方法配置并构建主 KDC 服务器、数据库和其他服务器：

- 自动 – 建议用于脚本
- 交互式 – 可满足大多数安装的要求
- 手动 – 执行较为复杂的安装时需要
- 手动加 LDAP – 将 LDAP 与 KDC 配合使用时需要

表 4-3 配置 KDC 服务器任务列表

任务	说明	有关说明
安装 KDC 软件包。	创建 KDC 所必需的软件包。	如何安装 KDC 软件包 [63]
（可选）配置 Kerberos 以 FIPS 140 模式运行。	设置为只使用 FIPS 140 验证算法。	如何配置 Kerberos 以 FIPS 140 模式运行 [64]
使用脚本配置主 KDC。	简化初始配置。	如何使用 `kdcmgr` 配置主 KDC [65] 例 4-1 “运行不带参数的 `kdcmgr` 命令”
使用脚本配置从 KDC 服务器。	简化初始配置。	如何使用 `kdcmgr` 配置从 KDC [67]
手动配置主 KDC 服务器。	在初始安装期间可控制 KDC 配置文件中的每个项。	如何手动配置主 KDC 服务器 [68]
手动配置从 KDC 服务器。	在初始安装期间可控制 KDC 配置文件中的每个项。	如何手动配置从 KDC 服务器 [72]
手动将主 KDC 配置为使用 LDAP。	将主 KDC 服务器配置为使用 LDAP。	如何将主 KDC 配置为使用 LDAP 目录服务器 [76]
替换 KDC 服务器上的主体密钥。	更新传统 KDC 服务器上的会话密钥，以使用更强的加密类型。	“替换主服务器上的票证授予服务密钥” [81]

▼ 如何安装 KDC 软件包

缺省情况下，Kerberos 客户机系统已安装在您的系统上。要安装密钥分发中心 (Key Distribution Center, KDC)，必须添加 KDC 软件包。

开始之前 您必须分配有 "Software Installation"（软件安装）权限配置文件才能将软件包添加到系统。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 安装 KDC 软件包。

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

有关更多信息，请参见 pkg(1) 手册页。

2. （可选） 列出 Kerberos 服务。

添加服务器软件包后，您的系统即配置了两个额外的 Kerberos 服务。与 Kerberos 客户机软件一样，这些服务在缺省情况下处于禁用状态。您应先配置 Kerberos，然后再启用这些服务。

```
$ svcs krb5
STATE          STIME    FMRI
disabled       Sep_10   svc:/network/security/krb5kdc:default
disabled       Sep_10   svc:/network/security/krb5_prop:default
$ svcs | grep kerb
STATE          STIME    FMRI
disabled       Sep_07   svc:/system/kerberos/install:default
```

▼ 如何配置 Kerberos 以 FIPS 140 模式运行

开始之前 要使 Kerberos 以 FIPS 140 模式运行，必须在系统上启用 FIPS 140 模式。请参见《在 Oracle Solaris 11.2 中管理加密和证书》中的“创建启用了 FIPS 140 的引导环境”。

1. 编辑 KDC 加密类型。

在 kdc.conf 文件的 [realms] 一节中，设置 KDC 数据库的主密钥类型：

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

2. 在同一文件中，显式禁止其他加密类型。

由于还可以通过运行命令设置加密，因此配置文件应禁止在命令中使用非 FIPS 140 算法参数。

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

3. 在 **krb5.conf** 文件的 **[libdefaults]** 一节中，编辑事务的加密类型。

这些参数限制 Kerberos 服务器、服务和客户机的加密类型。

```
# 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. 在同一文件中，显式禁止弱加密类型。

```
allow_weak_enctypes = false
```

故障排除　请参见“Kerberos 加密类型” [47]。

▼ 如何使用 kdcmgr 配置主 KDC

kdcmgr 脚本提供了用于安装主从 KDC 的命令行界面。对于主服务器，必须分别为 Kerberos 数据库和管理员创建一个口令。在从 KDC 上，必须提供这些口令才能完成安装。有关这些口令的信息，请参见 kdcmgr(1M) 手册页。

开始之前　您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **创建主 KDC。**

   在命令行上运行 kdcmgr 命令，并指定管理员和领域。

   系统将提示您提供 Kerberos 数据库口令（称为主密钥）和管理主体的口令。脚本将提示提供口令。

```
kdc1# kdcmgr -a kws/admin -r EXAMPLE.COM 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@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

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#
```

注 - 保存这些口令并将其存储在安全位置。

2. （可选） 显示主 KDC 的状态。

```
# kdcmgr status
```

3. 使用 NTP 或其他机制将此系统的时钟与该领域中的其他时钟同步。

要使验证成功通过，每个时钟都必须处于 krb5.conf 文件的 libdefaults 部分中定义的缺省时间之内。有关更多信息，请参见 krb5.conf(4) 手册页。有关网络时间协议 (Network Time Protocol, NTP) 的信息，请参见“同步 KDC 与 Kerberos 客户机的时钟” [112]。

注 - 主 KDC 不能是 NTP 服务器。如果未安装 NTP 服务器，请在安装 NTP 服务器后返回到主 KDC，并将主 KDC 配置为 NTP 服务器的一个客户机。

例 4-1 运行不带参数的 kdcmgr 命令

在本示例中，管理员在脚本提示时提供领域名称和 admin 主体。

```
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#
```

▼ 如何使用 kdcmgr 配置从 KDC

开始之前 已配置主 KDC 服务器。

您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 创建从 KDC。

   在命令行上运行 kdcmgr 命令，并指定管理员、领域和主 KDC。

   脚本将提示您提供在如何使用 kdcmgr 配置主 KDC [65]中创建的两个口令：其中一个是管理主体的口令，另一个是 KDC 数据库的口令。

```
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. （可选） 显示 KDC 的状态。

```
# kdcmgr status
```

3. 使用 NTP 或其他机制将此系统的时钟与该领域中的其他时钟同步。

如果未安装 NTP 服务器，可将此系统用作 NTP 服务器。

要使验证成功通过，每个时钟都必须处于 `krb5.conf` 文件的 `libdefaults` 部分中定义的缺省时间之内。有关更多信息，请参见 `krb5.conf(4)` 手册页。有关网络时间协议 (Network Time Protocol, NTP) 的信息，请参见“同步 KDC 与 Kerberos 客户机的时钟” [112]。

接下来的步骤 安装 NTP 服务器后返回到主 KDC，并将主 KDC 配置为 NTP 服务器的一个客户机。

▼ 如何手动配置主 KDC 服务器

在此过程中，将配置增量传播。此过程使用以下配置参数：

- 领域名称 = `EXAMPLE.COM`
- DNS 域名 = `example.com`
- 主 KDC = `kdc1.example.com`
- `admin` 主体 = `kws/admin`
- 联机帮助 URL = `http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html`

注 - 将 URL 调整为指向联机帮助的位置，如“`gkadmin` GUI” [134]中所述。

开始之前 主机已配置为使用 DNS。有关交换主 KDC 服务器的具体命名说明，请参见“交换主 KDC 服务器与从 KDC 服务器” [114]。

您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **安装 KDC 软件包。**
   按照如何安装 KDC 软件包 [63]中的说明操作。

2. **编辑 Kerberos 配置文件 (`krb5.conf`)。**
   有关此文件的说明，请参见 `krb5.conf(4)` 手册页。
   在本示例中，管理员更改了 `default_realm`、`kdc` 和 `admin_ server` 行以及所有 `domain_realm` 项，并编辑了 `help_url` 项。

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

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

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

[appdefaults]
gkadmin = {
help_url = http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html
}
```

注 - 如果必须与旧版 Kerberos 系统通信，您可能需要限制加密类型。有关限制加密类型所涉及的问题的说明，请参见“Kerberos 加密类型” [47]。

3. 在 KDC 配置文件 (`kdc.conf`) 中指定领域。

   有关此文件的说明，请参见 `kdc.conf`(4) 手册页。

   在本示例中，管理员不但更改了领域名称定义，而且更改了增量传播和日志记录缺省值。

```
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
sunw_dbprop_enable = true
sunw_dbprop_master_ulogsize = 1000
}
```

注 - 如果必须与旧版 Kerberos 系统通信，您可能需要限制加密类型。有关限制加密类型所涉及的问题的说明，请参见“Kerberos 加密类型” [47]。

4. 使用 `kdb5_util` 命令创建 KDC 数据库。

kdb5_util 命令可以创建 KDC 数据库。此外，使用 -s 选项时，该命令会在启动 kadmind 和 krb5kdc 守护进程之前，创建一个用于向自己验证 KDC 的存储文件。有关更多信息，请参见 kdb5_util(1M)、kadmind(1M) 和 krb5kdc(1M) 手册页。

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

提示 - 如果此步骤失败，请验证 KDC 主体是否用其 FQDN 标识。

```
# getent hosts IP-address-of-KDC
IP-address-of-KDC  kdc      /** This entry does not include FQDN **/
```

然后将该 FQDN 添加为 /etc/hosts 文件中的第一个 KDC 项，例如：

IP-address-of-KDC **kdc.***kdc-principal***.example.com kdc**

---

5. 编辑 Kerberos 访问控制列表文件 (**kadm5.acl**)。

   填充后，/etc/krb5/kadm5.acl 文件必须包含所有获许管理 KDC 的主体名称。

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

   通过前一项，EXAMPLE.COM 领域中的 kws/admin 主体可以修改 KDC 中的主体和策略。缺省主体项是一个型号 (*)，此字符可匹配所有 admin 主体。该项可能存在安全风险。修改文件以显式列出每个 admin 主体及其权限。有关更多信息，请参见 kadm5.acl(4) 手册页。

6. 向数据库添加管理主体。

   可以根据需要添加任意数目的 admin 主体。至少必须添加一个 admin 主体，这样才能完成 KDC 配置过程。此示例中添加了 kws/admin 主体。可以用适当的主体名称替代 "kws"。

   ```
   kadmin.local: addprinc kws/admin
   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.
   kadmin.local:
   ```

   有关更多信息，请参见 kadmin(1M) 手册页。

7. 启动 Kerberos 守护进程。

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

8. 启动 **kadmin** 并添加更多的主体。

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

a. 创建主 KDC **host** 主体。

基于 Kerberos 的应用程序（例如 kprop）使用 host 主体将更改传播到从 KDC。也可以通过使用网络应用程序（如 ssh），将该主体用于提供对 KDC 服务器的安全远程访问。请注意，当主体实例为主机名时，无论名称服务中的域名是大写还是小写，都必须以小写字母指定 FQDN。

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

b. （可选） 创建 **kclient** 主体。

kclient 实用程序在 Kerberos 客户机安装期间使用该主体。如果您没有计划使用此实用程序，则不需要添加该主体。kclient 实用程序的用户需要使用该口令。有关更多信息，请参见 kclient(1M) 手册页。

```
kadmin: addprinc clntconfig/admin
Enter password for principal clntconfig/admin@EXAMPLE.COM:     /** Type strong  password **/
Re-enter password for principal clntconfig/admin@EXAMPLE.COM: xxxxxxxx
Principal "clntconfig/admin@EXAMPLE.COM" created.
kadmin:
```

注 - 保存此口令并将其存储在安全位置。

c. 将特权添加到 **clntconfig/admin** 主体。

编辑 kadm5.acl 文件以向 clntconfig 主体授予执行 kclient 安装任务所需的足够特权。

```
# pfedit /etc/krb5/kadm5.acl
...
clntconfig/admin@EXAMPLE.COM  acdilm
```

d. 将主 KDC 的 **host** 主体添加到主 KDC 的密钥表文件。

通过将 host 主体添加到密钥表文件，应用服务器（如 sshd）可自动使用该主体。

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.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/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

e. 退出 **kadmin**。

```
kadmin: quit
```

9. 使用 NTP 或其他机制将此系统的时钟与该领域中的其他时钟同步。

   要使验证成功通过，每个时钟都必须处于 `krb5.conf` 文件的 `libdefaults` 部分中定义的缺省时间之内。有关更多信息，请参见 `krb5.conf`(4) 手册页。有关网络时间协议 (Network Time Protocol, NTP) 的信息，请参见“同步 KDC 与 Kerberos 客户机的时钟” [112]。

   注 - 主 KDC 不能是 NTP 服务器。如果未安装 NTP 服务器，请在安装 NTP 服务器后返回到主 KDC，并将主 KDC 配置为 NTP 服务器的一个客户机。

10. 配置从 KDC。

    要提供冗余性，请确保至少安装了一个从 KDC 服务器。按照如何使用 `kdcmgr` 配置从 KDC [67]或如何手动配置从 KDC 服务器 [72]中的说明操作。

▼ 如何手动配置从 KDC 服务器

在此过程中，将配置新的名为 `kdc2` 的从 KDC 服务器。此外还会配置增量传播。此过程使用以下配置参数：

- 领域名称 = `EXAMPLE.COM`
- DNS 域名 = `example.com`
- 主 KDC = `kdc1.example.com`
- 从 KDC = `kdc2.example.com`
- `admin` 主体 = `kws/admin`

开始之前 已配置主 KDC。要使该从服务器可交换，请按照如何交换主 KDC 服务器与从 KDC 服务器 [114]中的说明操作。

您必须承担 KDC 服务器上的 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 在主 KDC 服务器上，启动 **kadmin**。

   必须使用在配置主 KDC 服务器时创建的一个 `admin` 主体名称登录。

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

有关更多信息，请参见 kadmin(1M) 手册页。

a. 在主 KDC 服务器上，向数据库中添加从 host 主体（如果尚未完成）。

要使从 KDC 服务器正常工作，它必须有 host 主体。请注意，当主体实例为主机名时，无论名称服务中的域名是大写还是小写，都必须以小写字母指定 FQDN。

```
kadmin: addprinc -randkey host/kdc2.example.com
Principal "host/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

b. 在主 KDC 上，创建用于增量传播的主体。

kiprop 主体用于授权来自主 KDC 服务器的增量传播。

```
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

c. 退出 **kadmin**。

```
kadmin: quit
```

2. 在主 KDC 上，编辑 Kerberos 配置文件 (**krb5.conf**)。

需要为每个从 KDC 服务器添加一个项。有关此文件的说明，请参见 krb5.conf(4) 手册页。

```
kdc1 # pfedit /etc/krb5/krb5.conf
.
.
[realms]
EXAMPLE.COM = {
kdc = kdc1.example.com
kdc = kdc2.example.com
admin_server = kdc1.example.com
}
```

3. 在主 KDC 上，将 **kiprop** 项添加到 **kadm5.acl**。

通过此项，主 KDC 服务器可以接收对 kdc2 服务器的增量传播请求。

```
kdc1 # pfedit /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

4. 在主 KDC 上，重新启动 **kadmin** 服务以使用 **kadm5.acl** 文件中的新项。

```
kdc1 # svcadm restart network/security/kadmin
```

5. 在所有从 KDC 上，从主 KDC 服务器复制 KDC 管理文件。

   每个从 KDC 必须包含有关主 KDC 服务器的最新信息。可以使用 `sftp` 或类似的传输机制从主 KDC 获取以下文件的副本：

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

6. 在所有的从 KDC 服务器上，将主 KDC 服务器和每个从 KDC 服务器的项添加到数据库传播配置文件 **kpropd.acl** 中。

   需要在所有从 KDC 服务器上更新此信息。

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

7. 在所有的从 KDC 服务器上，确保未填充 Kerberos 访问控制列表文件 **kadm5.acl**。

   未修改的 `kadm5.acl` 文件可能如以下示例所示：

   ```
   kdc2 # pfedit /etc/krb5/kadm5.acl
   */admin@___default_realm___ *
   ```

   如果该文件中包含 `kiprop` 项，请将其删除。

8. 在新的从服务器上，在 **kdc.conf** 文件中定义其轮询间隔。

   将 `sunw_dbprop_master_ulogsize` 项替换为定义从服务器轮询间隔的项。以下项将轮询时间设置为两分钟：

   ```
   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
   sunw_dbprop_enable = true
   sunw_dbprop_slave_poll = 2m
   }
   ```

9. 在新的从 KDC 服务器上，启动 **kadmin** 命令。

   使用在配置主 KDC 时创建的 `admin` 主体名称之一登录。

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin:
```

a. 使用 **kadmin** 将从 KDC 的 host 主体添加到从 KDC 的密钥表文件中。

该项可使 kprop 命令及其他基于 Kerberos 的应用程序正常工作。请注意，当主体实例为主机名时，无论名称服务中的域名是大写还是小写，都必须以小写字母指定 FQDN。有关更多信息，请参见 kprop(1M) 手册页。

```
kadmin: ktadd host/kdc2.example.com
Entry for principal host/kdc2.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/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

b. 将 **kiprop** 主体添加到从 KDC 服务器的密钥表文件中。

通过将 kiprop 主体添加到 krb5.keytab 文件，允许 kpropd 命令在启动增量传播时对其自身进行验证。

```
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.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 kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. 退出 **kadmin**。

```
kadmin: quit
```

10. 在新的从 KDC 服务器上，启动 Kerberos 传播守护进程。

```
kdc2 # svcadm enable network/security/krb5_prop
```

11. 在新的从服务器上，使用 **kdb5_util** 命令创建一个存储文件。

```
kdc2 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key: xxxxxxxx
```

有关更多信息，请参见 kdb5_util(1M) 手册页。

12. **使用 NTP 或其他机制将此系统的时钟与该领域中的其他时钟同步。**
    要使验证成功通过，每个时钟都必须处于 `krb5.conf` 文件的 `libdefaults` 部分中定义的缺省时间之内。有关更多信息，请参见 `krb5.conf(4)` 手册页。有关网络时间协议 (Network Time Protocol, NTP) 的信息，请参见“同步 KDC 与 Kerberos 客户机的时钟” [112]。

13. **在新的从服务器上，启动 KDC 守护进程。**

```
kdc2 # svcadm enable network/security/krb5kdc
```

接下来的步骤 安装 NTP 服务器后返回到主 KDC，并将主 KDC 配置为 NTP 服务器的一个客户机。

▼ 如何将主 KDC 配置为使用 LDAP 目录服务器

此过程使用以下配置参数：

- 领域名称 = `EXAMPLE.COM`
- DNS 域名 = `example.com`
- 主 KDC = `kdc1.example.com`
- 目录服务器 = `dsserver.example.com`
- `admin` 主体 = `kws/admin`
- LDAP 服务的 FMRI = `svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1`
- 联机帮助 URL = `http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html`

注 - 将 URL 调整为指向联机帮助的位置，如“`gkadmin` GUI” [134]中所述。

开始之前 主机已配置为使用 DNS。为使性能更佳，请在同一台服务器上安装 KDC 和 LDAP 目录服务。此外还应运行目录服务器。以下过程适用于使用 Oracle Directory Server 企业版的服务器。有关更多信息，请参见 Oracle Identity Management 文档。

您必须承担 KDC 服务器上的 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **配置主 KDC 使用 SSL 访问目录服务器。**
   以下步骤将 KDC 配置为使用目录服务器的自签名证书。

   a. **在目录服务器上，导出自签名证书。**

```
# /export/sun-ds6.1/ds6/bin/dsadm show-cert -F der /export/sun-ds6.1/directory2 \
defaultCert > /tmp/defaultCert.cert.der
```

b. 在主 KDC 上，导入目录服务器的证书。

```
# pktool setpin keystore=nss dir=/var/ldap
# chmod a+r /var/ldap/*.db
# pktool import keystore=nss objtype=cert trust="CT" \
infile=/tmp/defaultCert.cert.der \
label=defaultCert dir=/var/ldap
```

有关更多信息，请参见 pktool(1) 手册页。

c. 在主 KDC 上，测试 SSL 是否可以正常运行。
此示例假定 cn=directory manager 项具有管理特权。

```
master# /usr/bin/ldapsearch -Z -P /var/ldap -D "cn=directory manager" \
-h dsserver.example.com -b "" -s base objectclass='*'
Subject:
"CN=dsserver.example.com,CN=636,CN=Directory Server,O=Example Corporation
```

请注意，CN=dsserver.example.com 项必须包含全限定主机名，而不是简短版本。

2. 如有需要，填充 LDAP 目录。

3. 将 Kerberos 架构添加到 LDAP 的现有架构。

```
# ldapmodify -h dsserver.example.com -D "cn=directory manager" \
-f /usr/share/lib/ldif/kerberos.ldif
```

4. 在 LDAP 目录中创建 Kerberos 容器。
向 krb5.conf 文件中添加下列各项：

a. 定义数据库类型。
添加一个项来在 realms 部分定义 database_module。

```
database_module = LDAP
```

b. 定义数据库模块。

```
[dbmodules]
LDAP = {
ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
db_library = kldap
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 = ldaps://dsserver.example.com
}
```

c. 在 LDAP 目录中创建 KDC。

以下命令创建 krbcontainer 及其他几个对象。还会创建 /var/krb5/.k5.EXAMPLE.COM 主密钥以及该密钥的存储文件。有关该命令的选项的更多信息，请参见 kdb5_ldap_util(1M) 手册页。

```
# kdb5_ldap_util -D "cn=directory manager" create
   -P master-key -r EXAMPLE.COM -s
```

5. 存储 KDC 绑定标识名 (Distinguished Name, DN) 口令。

   当绑定到目录服务器时，KDC 会使用这些口令。KDC 会根据其使用的访问类型使用不同的角色。

```
# 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"
```

6. 添加 KDC 服务角色。

   a. 创建一个内容如下的 **kdc_roles.ldif** 文件：

```
dn: cn=kdc service,ou=profile,dc=example,dc=com
cn: kdc service
sn: kdc service
objectclass: top
objectclass: person
userpassword: xxxxxxxx

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

   b. 在 LDAP 目录中创建角色项

```
# ldapmodify -a -h dsserver.example.com -D "cn=directory manager" -f kdc_roles.ldif
```

7. 设置 **kadmin** 相关角色的 ACL。

```
# cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
# Set kadmin ACL for everything under krbcontainer.
dn: cn=krbcontainer,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=krbcontainer,dc=example,dc=com")(targetattr="krb*")(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
```

```
add: aci
aci: (target="ldap:///ou=people,dc=example,dc=com")(targetattr="krb*")(version 3.0;\
acl kadmin_ACL; allow (all)\
userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)
EOF
```

8. 编辑 Kerberos 配置文件 (`krb5.conf`)。

   需要指定领域和服务器。有关此文件的说明，请参见 `krb5.conf(4)` 手册页。

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

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

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

[appdefaults]
gkadmin = {
help_url = http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html
}
```

注 - 将 URL 调整为指向联机帮助的位置，如“`gkadmin` GUI” [134]中所述。

在本示例中，`default_realm`、`kdc`、`admin_server` 行和所有 `domain_realm` 项都已更改。此外，联机帮助 URL 也已更改。

注 - 如果必须与旧版 Kerberos 系统通信，您可能需要限制加密类型。有关限制加密类型所涉及的问题的说明，请参见“Kerberos 加密类型” [47]。

---

9. 编辑 KDC 配置文件 (`kdc.conf`)。

   需要指定领域。有关此文件的说明，请参见 `kdc.conf(4)` 手册页。

   在本示例中，管理员不但更改了领域名称定义，而且更改了增量传播和日志记录缺省值。

```
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
sunw_dbprop_enable = true
sunw_dbprop_master_ulogsize = 1000
}
```

注 - 如果必须与旧版 Kerberos 系统通信，您可能需要限制加密类型。有关限制加密类型所涉及的问题的说明，请参见“Kerberos 加密类型” [47]。

10. 编辑 Kerberos 访问控制列表文件 (**kadm5.acl**)。

    填充后，/etc/krb5/kadm5.acl 文件应包含所有获许管理 KDC 的主体名称。

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

    通过前一项，EXAMPLE.COM 领域中的 kws/admin 主体可以修改 KDC 中的主体和策略。缺省主体项是一个型号 (*)，此字符可匹配所有 admin 主体。该项可能存在安全风险。修改文件以显式列出每个 admin 主体及其权限。有关更多信息，请参见 kadm5.acl(4) 手册页。

11. 启动 **kadmin.local** 命令并创建 **admin** 主体。

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

    a. 向数据库添加管理主体。

       可以根据需要添加任意数目的 admin 主体。必须至少创建一个 admin 主体，才能完成 KDC 配置过程。在本示例中，创建的是 kws/admin 主体。可以用适当的主体名称替代 "kws"。

       ```
       kadmin.local: addprinc kws/admin
       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.
       kadmin.local:
       ```

    b. 退出 **kadmin.local**。

       ```
       kadmin.local: quit
       ```

12. （可选） 配置 LDAP 对 Kerberos 服务的依赖性。

如果 LDAP 和 KDC 服务器运行在同一台主机上，且 LDAP 服务配置为使用 SMF，则添加 LDAP 服务对 Kerberos 守护进程的依赖性。如果 LDAP 服务重新启动，此依赖性将重新启动 KDC 服务。

a. 添加对 **krb5kdc** 服务的依赖性。

```
# svccfg -s security/krb5kdc
svc:/network/security/krb5kdc> addpg dsins1 dependency
svc:/network/security/krb5kdc> setprop dsins1/entities = \
fmri: "svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1"
svc:/network/security/krb5kdc> setprop dsins1/grouping = astring: "require_all"
svc:/network/security/krb5kdc> setprop dsins1/restart_on = astring: "restart"
svc:/network/security/krb5kdc> setprop dsins1/type = astring: "service"
svc:/network/security/krb5kdc> exit
```

b. 添加对 **kadmin** 服务的依赖性。

```
# svccfg -s security/kadmin
svc:/network/security/kadmin> addpg dsins1 dependency
svc:/network/security/kadmin> setprop dsins1/entities =\
fmri: "svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1"
svc:/network/security/kadmin> setprop dsins1/grouping = astring: "require_all"
svc:/network/security/kadmin> setprop dsins1/restart_on = astring: "restart"
svc:/network/security/kadmin> setprop dsins1/type = astring: "service"
svc:/network/security/kadmin> exit
```

13. 执行如何手动配置主 KDC 服务器 [68]中的步骤 7 到步骤 9 以完成 LDAP 中的 Kerberos 配置。

14. 配置从 KDC。

要提供冗余性，请确保至少安装了一个从 KDC 服务器。有关说明，请参见如何手动配置从 KDC 服务器 [72]。

替换主服务器上的票证授予服务密钥

注 - 如果要将新的更强加密类型用于所有会话密钥，请替换这些密钥。

当票证授予服务 (Ticket Granting Service, TGS) 主体仅具有 DES 密钥时，该密钥会将票证授予票证 (Ticket Granting Ticket, TGT) 会话密钥的加密类型限制为 DES。将 KDC 更新到支持更强加密类型的发行版后，必须替换 TGS 主体的 DES 密钥，以便该主体能够为所有会话密钥生成更强的加密。

可以远程替换密钥，也可以在主服务器上替换。您必须是分配了 `changepw` 特权的 `admin` 主体。

- 要从任何 Kerberos 系统替换 TGS 服务主体密钥，请使用 `kadmin` 命令。

```
kdc1 % /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM
Enter TGS key: xxxxxxxx
Enter new TGS key:      /** Type strong password **/
Re-enter TGS key to verify: xxxxxxxx
```

cpw 是 change_password 命令的别名。-randkey 选项将提示您提供新口令。

- 如果以 root 用户身份登录到主 KDC，您可以使用 kadmin.local 命令。该命令将提示您提供新的数据库口令。

```
kdc1 # kadmin.local -q 'cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM'
```

注 - 保存此口令并将其存储在安全位置。

在 LDAP 目录服务器上管理 KDC

大多数使用 LDAP 目录服务器的 KDC 管理任务与使用 DB2 服务器的任务相同。部分新任务特定于使用 LDAP。

表 4-4 配置 KDC 服务器使用 LDAP 任务列表

任务	说明	有关说明
配置主 KDC。	通过使用手动过程并将 KDC 配置为使用 LDAP，为领域配置并构建主 KDC 服务器和数据库。	如何将主 KDC 配置为使用 LDAP 目录服务器 [76]
混合使用 Kerberos 主体属性与非 Kerberos 对象类类型。	允许将信息存储在 Kerberos 记录中，以与其他 LDAP 数据库共享。	如何在非 Kerberos 对象类类型中混合 Kerberos 主体属性 [82]
销毁领域。	删除与领域关联的所有数据。	如何在 LDAP 目录服务器上销毁领域 [83]

▼ 如何在非 Kerberos 对象类类型中混合 Kerberos 主体属性

在该过程中，krbprincipalaux、krbTicketPolicyAux 和 krbPrincipalName 属性都与 people 对象类关联。

此过程使用以下配置参数：

- 目录服务器 = dsserver.example.com
- 用户主体 = mre@EXAMPLE.COM

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 准备 people 对象类中的每一项。

   在目录服务器上，对每个项重复此步骤。

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

2. 将一个子树属性添加到领域容器。

   本示例允许在 ou=people,dc=example,dc=com 容器和缺省 EXAMPLE.COM 容器中搜索主体项。

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

3. （可选） 如果 KDC 记录存储在 DB2 中，请迁移 DB2 项。

   a. 转储 DB2 项。

      ```
      # kdb5_util dump > dumpfile
      ```

   b. 将数据库装入 LDAP 服务器。

      ```
      # kdb5_util load -update dumpfile
      ```

4. （可选） 将主体属性添加到 KDC。

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

▼ 如何在 LDAP 目录服务器上销毁领域

此过程可用于已经将其他 LDAP 目录服务器配置为处理领域的情况。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

- 销毁领域。

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

配置 Kerberos 客户机

Kerberos 客户机包括网络上所有不是 KDC 服务器并且需要使用 Kerberos 服务的主机。本节介绍有关安装 Kerberos 客户机的过程，以及关于使用 `root` 验证来挂载 NFS 文件系统的信息。

客户机配置选项与服务器选项类似，但前者增加了自动化安装程序 (Automated Installer, AI)：

- AI – 建议用于快速而轻松地安装多个 Kerberos 客户机
- 自动 – 建议用于脚本
- 交互式 – 可满足大多数安装的要求
- 手动 – 执行较为复杂的安装时需要

以下任务列表描述了本节所包含的任务。

表 4-5 配置 Kerberos 客户机任务列表

任务	说明	有关说明
使用自动化安装程序 (Automated Installer, AI) 安装客户机。	如果希望在系统安装期间配置 Kerberos 客户机，则此任务适用。	《安装 Oracle Solaris 11.2 系统》中的“如何使用 AI 配置 Kerberos 客户机”
为相似的 Kerberos 客户机创建安装配置文件。	创建可重用的客户机安装配置文件。	如何创建 Kerberos 客户机安装配置文件 [84]
使用脚本安装客户机。	如果每个客户机的安全参数均相同，则此任务适用。	如何自动配置 Kerberos 客户机 [85]
通过回答提示安装客户机。	如果只需要更改少数几个安装参数，则此任务适用。	如何交互配置 Kerberos 客户机 [86]
手动安装客户机。	如果每个客户机安装都需要唯一的安装参数，则此任务适用。	如何手动配置 Kerberos 客户机 [90]
将 Kerberos 客户机加入到 Active Directory 服务器。	自动安装 Active Directory 服务器的 Kerberos 客户机。	如何将 Kerberos 客户机加入到 Active Directory 服务器 [89]
禁用颁发客户机票证授予票证 (Ticket Granting Ticket, TGT) 的 KDC 的验证。	当 Kerberos 客户机未将 host 主体存储在本地密钥表文件时，此任务可简化 KDC 验证。	“禁用票证授予票证的验证” [95]
使客户机能够以 `root` 用户身份访问 NFS 文件系统	使客户机能够使用 `root` 访问权限挂载 NFS 文件系统。此外，还使客户机能够访问 NFS 文件系统运行 cron 作业。	如何以 `root` 用户身份访问受 Kerberos 保护的 NFS 文件系统 [95]

▼ 如何创建 Kerberos 客户机安装配置文件

此过程创建可在安装 Kerberos 客户机时使用的 kclient 配置文件。使用该配置文件可以降低键入错误的可能性。此外，与交互式过程相比，使用该配置文件可以减少用户干预。

注 - 要创建最初作为完全配置的 Kerberos 客户机引导的系统，请参见《安装 Oracle Solaris 11.2 系统》中的“配置安全性”。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **创建 kclient 安装配置文件。**

   以下是 kclient 配置文件样例：

```
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. **保护并存储该文件以供其他客户机使用。**

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

▼ 如何自动配置 Kerberos 客户机

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **创建 kclient 配置文件。**

   使用在如何创建 Kerberos 客户机安装配置文件 [84]中创建的安全配置文件。

2. **运行带配置文件参数的 kclient 命令。**

   要完成此过程，必须提供 clntconfig 主体的口令。在“配置 KDC 服务器” [63]步骤中配置主 KDC 时，已创建该口令。有关更多信息，请参见 kclient(1M) 手册页。

```
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#
```

例 4-2 使用安装配置文件配置 Kerberos 客户机

以下示例使用 `kcprofile` 客户机配置文件以及两个命令行覆盖项来配置客户机。

```
# /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#
```

▼ 如何交互配置 Kerberos 客户机

此过程使用 `kclient` 安装实用程序而非安装配置文件。如果客户机将加入 Active Directory 服务器，则转到如何将 Kerberos 客户机加入到 Active Directory 服务器 [89]。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **运行不带参数的 kclient 命令。**

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

   脚本将提示您提供以下信息：

   - Kerberos 领域名称
   - 主 KDC 服务器主机名
   - KDC 从主机名
   - 映射到本地领域的域
   - 用于 Kerberos 验证的 PAM 服务名称和选项

   有关更多信息，请参见 kclient(1M) 手册页。

2. **如果 KDC 服务器未运行 Oracle Solaris 发行版，请回答 y 并定义运行 KDC 的服务器类型。**

   有关可用服务器的列表，请参见 kclient(1M) 手册页中的 -T 选项。

3. **如果应将 DNS 用于 Kerberos 查找，请回答 y 并指示要使用的 DNS 查找选项。**

   有效的选项有：dns_lookup_kdc、dns_lookup_realm 和 dns_fallback。有关这些值的更多信息，请参见 krb5.conf(4) 手册页。

4. **定义 Kerberos 领域的名称和主 KDC 主机名。**

   此信息将添加到 /etc/krb5/krb5.conf 配置文件中。

5. **如果从 KDC 位于该领域，请回答 y 并提供从 KDC 主机名。**

   此信息用于在客户机配置文件中创建其他 KDC 项。

6. **如果需要服务密钥或主机密钥，请回答 y。**

   通常情况下不需要服务密钥或主机密钥，除非客户机系统正在托管基于 Kerberos 的服务。

7. **如果客户机是某个群集的成员，请回答 y 并提供该群集的逻辑名称。**

   逻辑主机名可用于创建服务密钥，而后者是在群集中托管 Kerberos 服务所必需的。

8. **确定映射到当前领域的所有域和主机。**

   此映射允许其他域存在于客户机的缺省领域。

9. **指定客户机是否会使用基于 Kerberos 的 NFS。**

   当客户机使用 Kerberos 托管 NFS 服务时，需要创建 NFS 服务密钥。

10. **指示是否需要创建新的 PAM 策略。**

    要设置使用 Kerberos 进行验证的 PAM 服务，需要提供服务名称以及指示 Kerberos 验证的使用方式的标志。有效的标志选项包括：

    - `first` – 首先使用 Kerberos 验证，仅当 Kerberos 验证失败时才使用 UNIX
    - `only` – 仅使用 Kerberos 验证
    - `optional` – 有选择地使用 Kerberos 验证

    有关为 Kerberos 提供的 PAM 服务的信息，请查看 `/etc/security/pam_policy` 中的文件。

11. **指定是否应复制主 `/etc/krb5/krb5.conf` 文件。**

    借助此选项，可以指定在 `kclient` 的参数不足时使用的特定配置信息。

例 4-3 `kclient` 脚本的运行样例

```
...
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]: n
No action performed.
Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC host name for the above realm: kdc1.example.com

Note, this system and the KDC's time must be within 5 minutes of each other for
Kerberos to function. Both systems 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}: xscreensaver:first
Configuring /etc/pam.conf.
```

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

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

▼ 如何将 Kerberos 客户机加入到 Active Directory 服务器

此过程使用 `kclient` 命令而非安装配置文件。

开始之前 您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **（可选） 启用针对客户机的 DNS 资源记录创建。**

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

2. **运行 `kclient` 命令。**

以下输出显示了运行 `kclient` 命令将客户机加入 AD 域 (`EXAMPLE.COM`) 后生成的输出样例。

`-T` 选项将选择一种 KDC 服务器类型，本示例中选择的服务器类型为 Microsoft Active Directory (AD)。缺省情况下，必须提供 AD 服务器管理员主体的口令。

```
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.
#
```

有关更多信息，请参见 `kclient(1M)` 手册页。

▼ 如何手动配置 Kerberos 客户机

此过程使用以下配置参数：

- 领域名称 = `EXAMPLE.COM`
- DNS 域名 = `example.com`
- 主 KDC = `kdc1.example.com`
- 从 KDC = `kdc2.example.com`
- NFS 服务器 = `denver.example.com`
- 客户机 = `client.example.com`
- `admin` 主体 = `kws/admin`
- 用户主体 = `mre`
- 联机帮助 URL = `http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html`

开始之前 您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 编辑 Kerberos 配置文件 (**`krb5.conf`**)。

   更改 Kerberos 配置文件中的领域名称和服务器名称。还可以指定 `gkadmin` 的帮助文件的路径。

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

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

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

[appdefaults]
gkadmin = {
help_url = http://www.example.com/doclib/OSMKA/aadmin-23.html
```

注 - 如果必须与旧版 Kerberos 系统通信，您可能需要限制加密类型。有关限制加密类型所涉及的问题的说明，请参见“Kerberos 加密类型” [47]。

2. （可选） 更改用于定位 KDC 的过程。

   缺省情况下，Kerberos 领域到 KDC 的映射按如下顺序确定：

   - `krb5.conf` 中 `realms` 部分中的定义
   - 在 DNS 中查找 SRV 记录

   将 `dns_lookup_kdc` 或 `dns_fallback` 添加到 `krb5.conf` 文件的 `libdefaults` 部分，可以更改此行为。有关更多信息，请参见 `krb5.conf(4)`。请注意，始终会首先尝试引用。

3. （可选） 更改用于确定主机领域的过程。

   缺省情况下，主机到领域的映射按如下顺序确定：

   - 如果 KDC 支持引用，则 KDC 会通知客户机该主机属于哪个领域。
   - `krb5.conf` 文件中的 `domain_realm` 的定义。
   - 主机的 DNS 域名。
   - 缺省领域。

   将 `dns_lookup_kdc` 或 `dns_fallback` 添加到 `krb5.conf` 文件的 `libdefaults` 部分，可以更改此行为。有关更多信息，请参见 `krb5.conf(4)` 手册页。请注意，始终会首先尝试引用。

4. 使用 NTP 或其他时钟同步机制将客户机时钟与主 KDC 服务器时钟同步。

   要使验证成功通过，每个时钟都必须与 KDC 服务器上的时间同步，最大时差不得超过 `krb5.conf` 文件中 `clockskew` 关系定义的值。有关更多信息，请参见 `krb5.conf(4)` 手册页。有关网络时间协议 (Network Time Protocol, NTP) 的信息，请参见“同步 KDC 与 Kerberos 客户机的时钟” [112]。

5. 创建 Kerberos 主体。

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

   有关更多信息，请参见 `kadmin(1M)` 手册页。

   a. （可选） 如果不存在用户主体，请创建一个。

      仅当未向与该主机关联的用户分配主体时，才需要创建用户主体。

```
kadmin: addprinc mre
Enter password for principal mre@EXAMPLE.COM:     /** Type strong password **/
Re-enter password for principal mre@EXAMPLE.COM: xxxxxxxx
kadmin:
```

b. **（可选） 创建 root 主体，并将其添加到服务器的密钥表文件。**

注 - 如果客户机不需要对使用 NFS 挂载的远程文件系统有 root 访问权限，则可以跳过此步骤。

如果需要非交互式 root 访问权限（例如以 root 用户身份运行 cron 作业），则执行此步骤。

root 主体应具有两个组成部分。第二个组成部分应该是 Kerberos 客户机系统的主机名，以避免创建领域范围的 root 主体。当主体实例为主机名时，无论名称服务中的域名是大写还是小写，都必须以小写字母指定 FQDN。

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

c. **创建 root 主体，并将其添加到服务器的密钥表文件。**

远程访问服务使用 host 主体来提供验证。如果密钥表文件中尚无凭证，此主体允许 root 获取凭证。

```
kadmin: addprinc -randkey host/denver.example.com
Principal "host/denver.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/denver.example.com
Entry for principal host/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 host/denver.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **（可选） 将服务器的 NFS 服务主体添加到服务器的密钥表文件中。**

仅当客户机需要使用 Kerberos 验证访问 NFS 文件系统时才需要执行此步骤。

```
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:
```

e. 退出 **kadmin**。

```
kadmin: quit
```

6. （可选） 对 NFS 启用 Kerberos。

a. 在 **/etc/nfssec.conf** 文件中启用 Kerberos 安全模式。
在 /etc/nfssec.conf 文件中，删除将 Kerberos 安全模式注释掉的 "#"。

```
# 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
```

b. 启用 DNS。
如未启用 svc:/network/dns/client:default 服务，请启用它。有关更多信息，请参见 resolv.conf(4) 手册页。

```
# svcadm enable network/dns/client:default
```

c. 重新启动 **gss** 服务。

```
# svcadm restart network/rpc/gss
```

7. （可选） 要使客户机自动更新 TGT 或警告用户 Kerberos 票证失效，请在 **/etc/krb5/warn.conf** 文件中创建一个相应项。
有关更多信息，请参见 warn.conf(4) 手册页和“自动更新所有票证授予票证” [100]。

例 4-4 将 Oracle Solaris 客户机配置为使用多主 KDC

Microsoft Active Directory (AD) Kerberos 服务提供了一个在多个主服务器上运行的 KDC。要使 Oracle Solaris 客户机更新信息，/etc/krb5/krb5.conf 文件中的 admin_server 或 kpasswd_server 声明必须列出所有服务器。本示例显示了如何使客户机更新有关 kdc1 与 kdc2 共享的 KDC 的信息。

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

```
}
```

例 4-5 将 Kerberos 客户机配置为使用非 Oracle Solaris KDC

可以通过以下方法将 Kerberos 客户机配置为使用非 Oracle Solaris KDC：将一个行添加到 `/etc/krb5/krb5.conf` 文件的 `realms` 部分。该行会更改客户机与 Kerberos 口令更改服务器通信时使用的协议。以下摘录显示了该行的格式。

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

例 4-6 主机和域名到 Kerberos 领域的映射的 DNS TXT 记录

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
1989020501   ;serial
10800        ;refresh
3600         ;retry
3600000      ;expire
86400 )      ;minimum

IN      NS      kdc1.example.com.
kdc1                    IN      A       192.146.86.20
kdc2                    IN      A       192.146.86.21

_kerberos.example.com.             IN      TXT     "EXAMPLE.COM"
_kerberos.kdc1.example.com.        IN      TXT     "EXAMPLE.COM"
_kerberos.kdc2.example.com.        IN      TXT     "EXAMPLE.COM"
```

例 4-7 Kerberos 服务器位置的 DNS SRV 记录

此示例分别为 KDC、`admin` 服务器和 `kpasswd` 服务器的位置定义记录。

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
1989020501   ;serial
10800        ;refresh
3600         ;retry
3600000      ;expire
86400 )      ;minimum

IN      NS      kdc1.example.com.
kdc1                               IN      A       192.146.86.20
kdc2                               IN      A       192.146.86.21

_kerberos._udp.EXAMPLE.COM         IN      SRV 0 0 88  kdc2.example.com
_kerberos._tcp.EXAMPLE.COM         IN      SRV 0 0 88  kdc2.example.com
_kerberos._udp.EXAMPLE.COM         IN      SRV 1 0 88  kdc1.example.com
_kerberos._tcp.EXAMPLE.COM         IN      SRV 1 0 88  kdc1.example.com
```

```
_kerberos-adm._tcp.EXAMPLE.COM     IN      SRV 0 0 464 kdc1.example.com
_kpasswd._udp.EXAMPLE.COM          IN      SRV 0 0 464 kdc1.example.com
```

禁用票证授予票证的验证

缺省情况下，Kerberos 会检查存储在本地 /etc/krb5/krb5.keytab 文件中的 host 主体的 KDC 与颁发票证授予票证 (ticket-granting ticket, TGT) 的 KDC 是否相同。此检查功能 (verify_ap_req_nofail) 可防止 DNS 欺骗攻击。

但是，对于 host 主体不可用的客户机配置，必须禁用此检查功能。以下配置要求禁用此检查功能：

- 客户机 IP 地址以动态方式分配，例如 DHCP 客户机。
- 客户机未配置为托管任何服务，所以没有创建 host 主体。
- 未在客户机上储存主机密钥。

要禁用 TGT 验证，请将 krb5.conf 文件中的 verify_ap_req_nofail 选项设置为 false。可以在 krb5.conf 文件的 [libdefaults] 或 [realms] 部分中输入 verify_ap_req_nofail 选项。[libdefaults] 部分中的设置用于所有领域：

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

如果该选项位于 [realms] 部分，则该设置仅应用到定义的领域。有关此选项的更多信息，请参见 krb5.conf(4) 手册页。

▼ 如何以 root 用户身份访问受 Kerberos 保护的 NFS 文件系统

通过此过程，客户机可以使用 root ID 特权访问要求 Kerberos 验证的 NFS 文件系统。特别是，使用如下选项共享 NFS 文件系统时：-o sec=krb5,root=client1.example.com。

1. 运行 **kadmin** 命令。

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

2. 为 NFS 客户机创建 **root** 主体。

此主体用于对要求 Kerberos 验证的 NFS 挂载的文件系统提供 root 等效访问权限。root 主体应具有两个组成部分。第二个组成部分应该是 Kerberos 客户机系统的主机名，以

避免创建领域范围的 root 主体。请注意，当主体实例为主机名时，无论名称服务中的域名是大写还是小写，都必须以小写字母指定 FQDN。

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

3. **将 root 主体添加到服务器的密钥表文件中。**

   如果客户机需要对使用 NFS 服务挂载的文件系统有 root 访问权限，则必须执行此步骤。如果需要非交互式 root 访问权限（例如以 root 用户身份运行 cron 作业），也必须执行该步骤。

```
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:
```

4. **退出 kadmin。**

```
kadmin: quit
```

▼ 如何在 Kerberos 领域中配置用户自动迁移

没有 Kerberos 主体的用户可以使用 PAM 自动迁移到现有 Kerberos 领域。可以在迁移服务器和主服务器上定制基于系统的 PAM 配置文件，以处理 UNIX 凭证的识别和 Kerberos 领域中的重新验证。

有关 PAM 的信息，请参见第 1 章 使用可插拔验证模块和 pam.conf(4) 手册页。

在此过程中，login 服务名称配置为使用自动迁移。本示例使用以下配置参数：

- 领域名称 = EXAMPLE.COM
- 主 KDC = kdc1.example.com
- 托管迁移服务的计算机 = server1.example.com
- 迁移服务主体 = host/server1.example.com

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **确保 server1 存在 host 服务主体。**

   server1 的 keytab 文件中的 host 服务主体用于向主 KDC 服务器验证该服务器。

```
server1 # klist -k
```

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

有关该命令的选项的更多信息，请参见 klist(1) 手册页。

2. 如果未列出 **server1**，请将其配置为 **EXAMPLE.COM** 领域的一个 Kerberos 客户机。
   有关步骤，请参见“配置 Kerberos 客户机” [84]中的示例。

3. 修改 **server1** 的 PAM 策略。
   有关更多信息，请参见“分配基于用户的 PAM 策略” [19]。

   a. 确定 **server1** 正在使用的 Kerberos 策略。

   ```
   % 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. 复制该 PAM 策略文件，然后修改新的策略文件以将 **pam_krb5_migrate.so.1** 模块附加到每个验证栈。

   ```
   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 requisite    pam_authtok_get.so.1
   ...
   login auth required  pam_unix_auth.so.1
   login    auth optional  pam_krb5_migrate.so.1
   #
   # rlogin service (explicit because of pam_rhost_auth)
   #
   rlogin auth sufficient  pam_rhosts_auth.so.1
   ...
   rlogin auth required    pam_unix_auth.so.1
   rlogin   auth optional        pam_krb5_migrate.so.1
   #
   # Kerberized rlogin service
   #
   krlogin auth required  pam_unix_cred.so.1
   krlogin auth required  pam_krb5.so.1
   krlogin auth optional  pam_krb5_migrate.so.1
   #
   # rsh service (explicit because of pam_rhost_auth)
   #
   rsh auth sufficient  pam_rhosts_auth.so.1
   rsh auth required    pam_unix_cred.so.1
   rsh auth optional    pam_krb5_migrate.so.1
   ```

```
#
# Kerberized rsh service
#
krsh auth required  pam_unix_cred.so.1
krsh auth required  pam_krb5.so.1
krsh auth optional  pam_krb5_migrate.so.1
#
# Kerberized telnet service
#
ktelnet auth required  pam_unix_cred.so.1
ktelnet auth required  pam_krb5.so.1
ktelnet auth optional  pam_krb5_migrate.so.1
#
# PPP service (explicit because of pam_dial_auth)
#
ppp auth requisite    pam_authtok_get.so.1
...
ppp auth required  pam_unix_auth.so.1
ppp auth optional  pam_krb5_migrate.so.1
#
# GDM Autologin (explicit because of pam_allow).  These need to be
# here as there is no mechanism for packages to amend pam.conf as
# they are installed.
#
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
# Used when service name is not explicitly mentioned for authentication
#
OTHER auth requisite    pam_authtok_get.so.1
...
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
#
# cron service (explicit because of non-usage of pam_roles.so.1)
#
cron account required pam_unix_account.so.1
#
# cups service (explicit because of non-usage of pam_roles.so.1)
#
cups account required pam_unix_account.so.1
#
# GDM Autologin (explicit because of pam_allow) This needs to be here
# as there is no mechanism for packages to amend pam.conf as they are
# installed.
#modified
gdm-autologin account  sufficient  pam_allow.so.1
#
```

```
.
.
.
```

c. （可选） 强制立即更改口令。

对于新创建的 Kerberos 帐户，请通过以下方法将口令失效时间设置为当前时间：将 `expire_pw` 选项添加到 `pam_krb5_migrate` 项。有关更多信息，请参见 `pam_krb5_migrate(5)` 手册页。

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

d. 在此配置文件中，修改 **OTHER** 帐户栈以在 Kerberos 口令失效时阻止访问。

```
# Definition for Account management
# Used when service name is not explicitly mentioned for account management
# Re-ordered pam_krb5 causes password expiration in Kerberos 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. 更改 **policy.conf** 文件中的 **PAM_POLICY** 项以使用修改后的配置文件。

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

有关更多信息，请阅读 `policy.conf` 文件。

4. 在主 KDC 上，更新 **kadm5.acl** 访问控制文件。

以下项将为所有用户（`root` 用户除外）授予对 `host/server1.example.com` 服务主体的迁移和查询特权。使用 `U` 特权列出不得迁移的用户。这些项必须位于 "permit all" 或 `ui` 项之前。有关更多信息，请参见 `kadm5.acl(4)` 手册页。

```
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. 在主 KDC 上，启用 **kadmind** 守护进程以使用 **k5migrate** PAM 服务。

如果 k5migrate 服务文件不在 /etc/pam.d 目录中，请将该服务文件添加到此目录。有关更多信息，请参见pam.d(4) 手册页。

这项修改可实现对需要迁移的帐户执行 UNIX 用户口令验证。

```
kdc1 # pfedit /etc/pam.d/k5migrate
...
# Permits validation of migrated UNIX accounts
auth    required        pam_unix_auth.so.1
account required        pam_unix_account.so.1
```

注 - k5migrate 是 PAM 服务的名称。文件必须命名为 k5migrate。

6. 请先测试配置，然后再将其投入使用。

   - 以一般用户身份测试每个修改后的 PAM 服务。
   - 以 root 用户身份测试每个修改后的 PAM 服务。
   - 强制立即更改口令，然后测试修改后的 PAM 服务。

自动更新所有票证授予票证

为便于管理，可以配置票证更新和有关票证授予票证 (Ticket Granting Ticket, TGT) 失效的警告消息。管理员可以设置针对所有用户的警告，而用户可以定制自己的警告。有关更多信息，请参见 warn.conf(4) 和 ktkt_warnd(1M) 手册页。

注 - 缺省情况下，ktkt_warn 服务处于禁用状态。要在现有 Kerberos 客户机上启用该服务，请运行 svcadm enable ktkt_warn 命令。

例 4-8 为所有用户配置 TGT 失效消息

本示例显示了为 TGT 配置更新和消息系统的多种方法。

```
# pfedit /etc/krb5/warn.conf
##
## renew the TGT 30 minutes before expiration and send message to users terminal
##
mre@EXAMPLE.COM renew:log terminal 30m
##
## send a warning message to a specific email address 20 minutes before TGT expiration
##
mre@EXAMPLE.COM mail 20m mre@example2.com
##
# renew the TGT 20 minutes before expiration and send an email message on failure
```

```
##
bricker@EXAMPLE.COM renew:log-failure mail 20m -
##
## catch-all: any principal not matched above will get an email warning
* mail 20m -
```

配置消息后，在新客户机上运行 kclient 命令。

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

在现有客户机上启用该服务。

```
# svcadm enable network/security/ktkt_warn
```

例 4-9 为用户配置 TGT 失效消息

每个用户都可以配置单独的 warnd 配置文件，该配置文件名为 /var/user/*$USER*/krb-warn.conf。存在此文件是为了防止读取管理员文件。

```
% pfedit /var/user/mre/krb-warn.conf
mre@EXAMPLE.COM renew:log mail 25m &
```

TGT 会于失效的 25 分钟之前更新，系统会记录更新操作，并且会在当时向 Kerberos 用户 mre 发送邮件。

配置 Kerberos 网络应用服务器

网络应用服务器是使用以下一个或多个网络应用程序提供访问的主机：ftp、rcp、rlogin、rsh、ssh 和 telnet。要在服务器上启用这些应用程序的 Kerberos 版本，只需执行几个步骤。

▼ 如何配置 Kerberos 网络应用服务器

此过程使用以下配置参数：

- 应用服务器 = boston
- admin 主体 = kws/admin
- DNS 域名 = example.com
- 领域名称 = EXAMPLE.COM

开始之前 已配置主 KDC。时钟已同步，如“同步 KDC 与 Kerberos 客户机的时钟” [112]中所述。要完全测试此过程，需要多个客户机。

您必须承担应用程序服务器上的 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **确定新服务器是否存在 host 主体。**

   以下命令报告是否存在 host 主体：

   ```
   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
   ```

   如果命令确实返回一个主体，则表示存在。如果命令未返回主体，请通过以下步骤创建新主体。

2. **使用在配置主 KDC 时创建的 admin 主体名称之一登录到服务器。**

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

3. **创建服务器的 host 主体。**

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

   host 主体有以下用途：

   - 使用远程命令（如 rsh 和 ssh）时验证通信。
   - 供 pam_krb5 防止 KDC 欺骗攻击，以确认用户的 Kerberos 凭证是从可信 KDC 获取的。
   - 允许 root 用户在不要求存在 root 主体的情况下，自动获取 Kerberos 凭证。在执行手动 NFS 挂载时，若共享需要使用 Kerberos 凭证，此功能很有用。

   如果要使用 Kerberos 服务验证使用远程应用程序的通信，则需要该主体。如果服务器有多个与之关联的主机名，则应使用主机名的 FQDN 格式为每个主机名创建一个主体。

4. **将服务器的 host 主体添加到服务器的密钥表文件中。**

   如果没有运行 kadmin 命令，请使用以下类似语法重新启动该命令：/usr/sbin/kadmin -p *kws*/admin

   如果服务器有多个与之关联的主机名，请在密钥表中为每个主机名添加一个主体。

   ```
   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:
```

5. 退出 **kadmin**。

```
kadmin: quit
```

▼ 运行 FTP 时如何将通用安全服务与 Kerberos 配合使用

Kerberos 网络应用程序可以使用通用安全服务 (generic security service, GSS) 实现验证、完整性和保密性。以下步骤显示如何为 ProFTPD 启用 GSS。

开始之前 您必须承担 FTP 服务器上的 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 为 FTP 服务器添加主体并更新 FTP 服务器的密钥表文件。

   如果您之前执行过更改，可能无需执行下列步骤。

   a. 启动 **kadmin** 命令。

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

   b. 为 FTP 服务器添加 **ftp** 服务主体。

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

   c. 将 **ftp** 服务主体添加到新的密钥表文件中。

   新的密钥表文件将使该信息可供 ftp 服务使用，同时不公开该服务器的密钥表文件中的所有信息。

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

   有关更多信息，请参见 kadmin(1M) 手册页中的 ktadd 命令。

2. 更改新的密钥表文件的所有权。

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

3. 为 FTP 服务器启用 GSS。

   对 /etc/proftpd.conf 文件执行下列更改。

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

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

4. 重新启动 FTP 服务器。

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

配置 Kerberos NFS 服务器

NFS 服务使用 UNIX 用户 ID (user ID, UID) 标识用户，不能直接使用 GSS 凭证。要将凭证转换为 UID，可能需要创建将用户凭证映射到 UNIX UID 的凭证表。有关缺省凭证映射的信息，请参见“将 GSS 凭证映射到 UNIX 凭证” [60]。本节中的过程重点介绍配置 Kerberos NFS 服务器、管理凭证表以及对 NFS 挂载的文件系统启动 Kerberos 安全模式所需执行的任务。以下任务列表描述了本节所包含的任务。

表 4-6 配置 Kerberos NFS 服务器任务列表

任务	说明	有关说明
配置 Kerberos NFS 服务器。	使服务器共享要求 Kerberos 验证的文件系统。	如何配置 Kerberos NFS 服务器 [104]
创建并修改凭证表。	当缺省映射不满足要求时，创建用于将 GSS 凭证映射到 UNIX UID 的凭证表，然后添加一个项。	如何创建和修改凭证表 [106]
将来自其他领域的用户凭证映射到 UNIX UID。	更新凭证表中的信息。	例 4-10 “将其他域中的主体添加到 Kerberos 凭证表”
创建两个类似领域之间的凭证映射。	当多个领域共享同一个口令文件时，将来自其中一个领域的 UID 映射到另一个领域。	如何提供各领域之间的凭证映射 [106]
使用 Kerberos 验证共享文件系统。	使用安全模式共享文件系统，以便要求 Kerberos 验证。	如何设置使用多种 Kerberos 安全模式的安全 NFS 环境 [107]

▼ 如何配置 Kerberos NFS 服务器

此过程使用以下配置参数：

- 领域名称 = `EXAMPLE.COM`
- DNS 域名 = `example.com`
- NFS 服务器 = `denver.example.com`
- `admin` 主体 = `kws/admin`

开始之前 您必须承担 NFS 服务器上的 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

已配置主 KDC。时钟已同步，如“同步 KDC 与 Kerberos 客户机的时钟” [112]中所述。要完全测试此过程，需要多个客户机。

1. 将 NFS 服务器配置为 Kerberos 客户机。
   请按照“配置 Kerberos 客户机” [84]中的说明操作。

2. 添加 NFS 服务主体。
   使用 `kadmin` 命令。

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

   a. 创建 NFS 服务主体。
      请注意，当主体实例为主机名时，无论名称服务中的域名是大写还是小写，都必须以小写字母指定 FQDN。
      对系统上可能用于访问 NFS 数据的每个唯一接口重复此步骤。如果主机有多个接口都具有唯一名称，则每个唯一名称必须具有自己的 NFS 服务主体。

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

   b. 将服务器的 NFS 服务主体添加到服务器的密钥表文件中。
      对步骤 2.a 中创建的每个唯一服务主体重复此步骤。

      ```
      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:
      ```

   c. 退出 **kadmin**。

      ```
      kadmin: quit
      ```

3. 如果需要，创建特殊 GSS 凭证映射。
   通常，Kerberos 服务生成 GSS 凭证与 UNIX UID 之间的适当映射。缺省映射在“将 GSS 凭证映射到 UNIX 凭证” [60]中有介绍。如果缺省映射不满足要求，请参见如何创建和修改凭证表 [106]了解更多信息。

4. 使用 Kerberos 安全模式共享 NFS 文件系统。
   有关更多信息，请参见如何设置使用多种 Kerberos 安全模式的安全 NFS 环境 [107]。

▼ 如何创建和修改凭证表

NFS 服务器使用 gsscred 凭证表将 Kerberos 凭证映射到 UNIX UID。缺省情况下，主体名称的基本部分与 UNIX 登录名匹配。如果缺省映射不满足要求，可创建此表。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **确保 /etc/gss/gsscred.conf 中指示的安全机制为 files。**

```
# cat /etc/gss/gsscred.conf
...
#
files
#
#
# Syslog (auth.debug) a message for GSS cred to Unix cred mapping
#SYSLOG_UID_MAPPING=yes
```

2. **使用 gsscred 命令创建凭证表。**

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

gsscred 命令从 svc:/system/name-service/switch:default 服务的 passwd 项列出的所有源中收集信息。如果不想将本地口令项包含到凭证表，可临时删除 files 项。有关更多信息，请参见 gsscred(1M) 手册页。

3. **（可选） 将项添加到凭证表。**
例如，在 NFS 服务器上以 root 角色添加一个项，以将 sandy/admin 主体映射到 3736 UID。-a 选项可将该项添加到凭证表。

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

例 4-10 将其他域中的主体添加到 Kerberos 凭证表

本示例使用全限定域名 (Fully Qualified Domain Name, FQDN) 来指定其他域中的主体。

```
# gsscred -m kerberos_v5 -n sandy/admin@EXAMPLE.COM -u 3736 -a
```

▼ 如何提供各领域之间的凭证映射

此过程提供使用同一个口令文件的领域之间的正确凭证映射。在本示例中，领域 CORP.EXAMPLE.COM 和 SALES.EXAMPLE.COM 使用同一个口令文件。*username*@CORP.EXAMPLE.COM 和 *username*@SALES.EXAMPLE.COM 的凭证映射到同一个 UID。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

- **在客户机系统上，将 default_realm 和 auth_to_local_realm 两个项添加到 krb5.conf 文件。**

```
# pfedit /etc/krb5/krb5.conf
[libdefaults]
default_realm = CORP.EXAMPLE.COM
.
[realms]
CORP.EXAMPLE.COM = {
.
auth_to_local_realm = SALES.EXAMPLE.COM
.
}
```

故障排除 有关排除凭证映射问题的帮助，请参见“观察从 GSS 凭证到 UNIX 凭证的映射” [175]。

▼ 如何设置使用多种 Kerberos 安全模式的安全 NFS 环境

通过此过程，NFS 服务器可以使用多种安全模式提供安全的 NFS 访问。客户机与 NFS 服务器协商安全模式时，客户机将使用服务器所提供的第一种模式。客户机随后再请求该服务器共享的文件系统时，都将使用此模式。

开始之前 您必须承担 NFS 服务器上的 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **检验在密钥表文件中是否存在 NFS 服务主体。**

   klist 命令报告是否存在密钥表文件并显示主体。如果结果显示不存在密钥表文件或者不存在 NFS 服务主体，则您需要确认是否已完成如何配置 Kerberos NFS 服务器 [104]中的所有步骤。

```
# 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
```

   有关更多信息，请参见 klist(1) 手册页。

2. **在 `/etc/nfssec.conf` 文件中启用 Kerberos 安全模式。**

   在 `/etc/nfssec.conf` 文件中，删除将 Kerberos 安全模式注释掉的 "#"。

```
# 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. **使用适当的安全模式共享文件系统。**

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

mode	指定共享文件系统时要使用的安全模式。使用多种安全模式时，会将列表中的第一种模式用作缺省模式。
file-system	定义要共享的文件系统的路径。

   尝试从指定的文件系统访问文件的所有客户机都需要进行 Kerberos 验证。要访问文件，应验证 NFS 客户机上的用户主体。

4. **（可选） 使用非缺省安全模式挂载文件系统。**

   如果可接受缺省安全模式，则不要执行此过程。

   - **如果使用的是自动挂载程序，请编辑 `auto_master` 数据库以进入非缺省安全模式。**

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

   - **手动发出 `mount` 命令以使用非缺省模式访问文件系统。**

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

例 4-11 使用一种 Kerberos 安全模式共享文件系统

在本示例中，使用 `krb5` 安全模式进行的验证必须成功，才能通过 NFS 服务访问任何文件。

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

例 4-12 使用多种 Kerberos 安全模式共享文件系统

在此示例中，选择了所有三种 Kerberos 安全模式。客户机与 NFS 服务器协商决定使用的模式。如果命令中的第一种模式失败，即将尝试下一种。有关更多信息，请参见 nfssec(5) 手册页。

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

为 Kerberos 服务访问配置延迟执行

在缺省 Kerberos 环境中，凭证将在有限的时间后失效。对于可在任意时间执行的进程（例如 cron 和 at），时间受限制将带来问题。本过程介绍如何配置 Kerberos 环境支持需要通过 Kerberos 验证服务的延迟执行进程。Oracle Solaris 提供了 PAM 模块，并使用服务密钥和 kclient 配置选项来实现带 Kerberos 验证的延迟执行，同时使其比其他解决方案更加安全。

注 - 如果 cron 服务器受到危害，攻击者可能会伪装成用户来获取配置在 cron 服务器上的目标服务的访问权限。因此，请将在本过程配置的 cron 主机视为敏感性较高的系统，因为该主机为用户提供中间服务。

▼ 如何为 Kerberos 服务访问配置 cron 主机

此过程使用以下配置参数：

- cron 主机 = host1.example.com
- NFS 服务器 = host2.example.com
- LDAP 服务器 = host3.example.com

1. **配置 cron 服务以支持 Kerberos。**

   - **如果未针对 Kerberos 配置 cron 主机，请在系统上运行 kclient 命令。**

     有关更多信息，请参见 kclient(1M) 手册页。

     例如，以下命令将配置位于 EXAMPLE.COM 领域中的客户机。该命令使用 include 机制将 pam_gss_s4u 文件包含到 /etc/pam.d/cron 服务文件。

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

   - **如果已针对 Kerberos 配置 cron 主机，则必须手动在该主机上修改 cron 服务的 PAM 配置。**

     确保 cron 服务的 PAM 配置包含 pam_gss_s4u 文件。

     ```
     # 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. 使 **cron** 充当委托代表。
   例如：

   ```
   # 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. 使 **cron** 主机代表创建了 **cron** 作业的用户为自身请求票证。

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

4. 在 LDAP 中，配置 **cron** 以指定其用作委托代表的服务。
   例如，要使 `cron` 主机访问用户位于 `host2`（基于 Kerberos 的 NFS 服务器）上的起始目录，请将 NFS 主机添加到 `cron` 服务器的 LDAP 定义中的 `krbAllowedToDelegateTo` 参数。

   a. 创建委托指定。

      ```
      # 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. 将该指定添加到 LDAP。

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

配置跨领域验证

有几种方法可以将各个领域链接在一起，从而允许在一个领域中验证另一个领域中的用户。跨领域验证通过建立一个由两个领域共享的密钥来实现。领域的关系可以为层次化或方向化。有关更多信息，请参见“Kerberos 领域分层结构” [54]。

▼ 如何建立层次化跨领域验证

本过程中的示例在 `CORP.EAST.EXAMPLE.COM` 与 `EAST.EXAMPLE.COM` 之间建立了双向跨领域验证。必须在这两个领域的主 KDC 上完成本过程。

开始之前 已为每个领域配置主 KDC。要完全测试验证过程，需要多个客户机。

您必须同时承担两台 KDC 服务器上的 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **为两个领域创建票证授予票证服务主体。**
   必须使用在配置主 KDC 服务器时创建的一个 admin 主体名称登录。

```
# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: addprinc krbtgt/CORP.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
Enter password for principal krbtgt/CORP.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:     /** Type strong
 password **/
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM
Enter password for principal krbtgt/EAST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM:     /** Type strong
 password **/
kadmin: quit
```

   注 - 保存这些口令并将其存储在安全位置。

2. **向 Kerberos 配置文件中添加项来定义每个领域的域名。**

```
# pfedit /etc/krb5/krb5.conf
[libdefaults]
.
.
[domain_realm]
.corp.east.example.com = CORP.EAST.EXAMPLE.COM
.east.example.com = EAST.EXAMPLE.COM
```

   在本示例中，定义了 CORP.EAST.EXAMPLE.COM 和 EAST.EXAMPLE.COM 两个领域的域名。由于会从上向下搜索文件，因此在文件中子域必须位于域名之前。

3. **将 Kerberos 配置文件复制到此领域中的所有客户机。**
   要使跨领域验证正常工作，所有系统（包括从 KDC 和其他服务器）都必须使用 /etc/krb5/krb5.conf 的主 KDC 版本。

4. **在第二个领域中重复此过程。**

   注 - 在两个 KDC 中为每个服务主体指定的口令必须完全相同。因此，服务主体 krbtgt/CORP.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM 的口令在两个领域中必须相同。

▼ 如何建立直接跨领域验证

本过程中的示例使用 CORP.EAST.EXAMPLE.COM 和 SALES.WEST.EXAMPLE.COM 两个领域。将按两个方向建立跨领域验证。必须在主 KDC 服务器上在这两个领域中完成此过程。

开始之前 已为每个领域配置主 KDC。要完全测试验证过程，需要多个客户机。

您必须同时承担两台 KDC 服务器上的 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **为两个领域创建票证授予票证服务主体。**
   必须使用在配置主 KDC 服务器时创建的一个 admin 主体名称登录。

```
# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: addprinc krbtgt/CORP.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
krbtgt/CORP.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM:     /** Type strong password **/
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM
Enter password for principal
krbtgt/SALES.WEST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM:     /** Type strong password **/
kadmin: quit
```

2. **向 Kerberos 配置文件中添加项来定义指向远程领域的直接路径。**
   本示例显示了 CORP.EAST.EXAMPLE.COM 领域中的客户机。要在 SALES.WEST.EXAMPLE.COM 领域中添加相应定义，请交换领域名称。

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

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

3. **将 Kerberos 配置文件复制到当前领域中的所有客户机。**
   要使跨领域验证正常工作，所有系统（包括从 KDC 和其他服务器）都必须使用 Kerberos 配置文件 (/etc/krb5/krb5.conf) 的新版本。

4. **对第二个领域重复此过程。**

同步 KDC 与 Kerberos 客户机的时钟

所有参与 Kerberos 验证系统的主机都必须在指定的最长时间（称为时钟相位差）内同步其内部时钟。针对这一要求，需要进行另一种 Kerberos 安全检查。如果任意两个参与主机之间的时间偏差超过了时钟相位差，则客户机请求会被拒绝。

时钟相位差还决定应用服务器必须跟踪 Kerberos 协议消息的时间长度，以便识别并拒绝重放的请求。因此，时钟相位差的值越大，应用服务器必须收集的信息就越多。

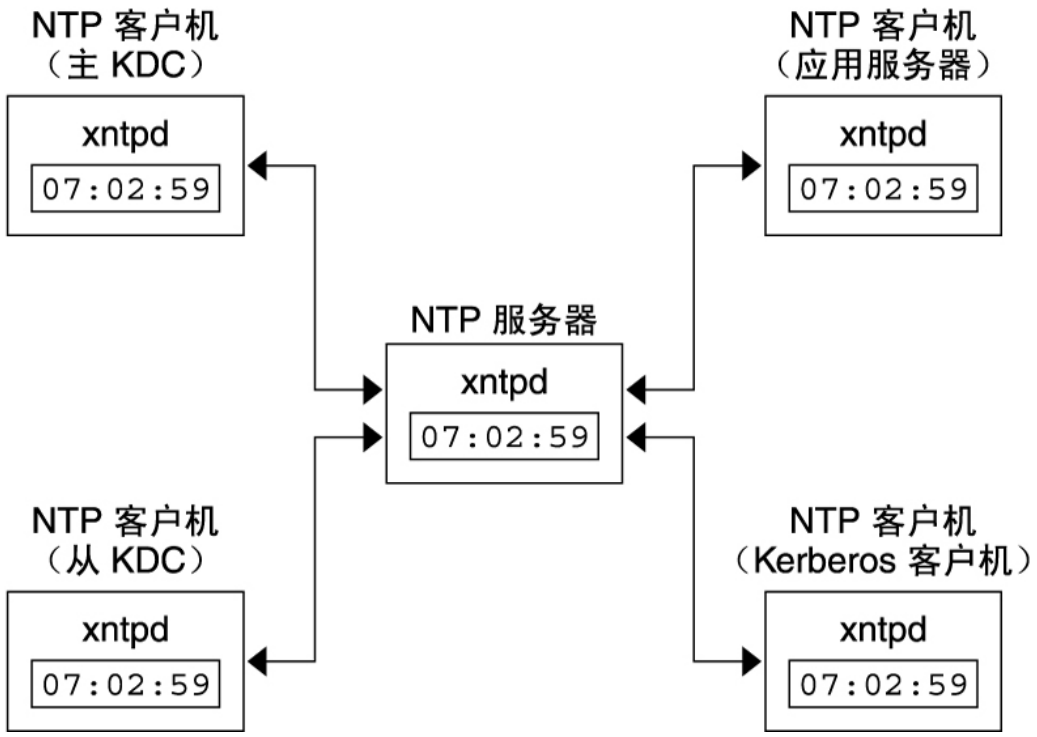
时钟相位差的最大缺省值为 300 秒（5 分钟）。可以在 `krb5.conf` 文件的 `libdefaults` 部分中更改此缺省值。

注 - 出于安全原因，不要将时钟相位差增大到超过 300 秒。

在 KDC 与 Kerberos 客户机之间保持时钟同步非常重要，因此应使用网络时间协议 (Network Time Protocol, NTP) 软件同步这些时钟。Oracle Solaris 软件中包括了由美国特拉华大学开发的 NTP 公共域软件。 可从 NTP Documentation（NTP 文档）获取文档。

通过 NTP，您可以在网络环境中管理精确时间或网络时钟同步，或者同时管理这两者。NTP 是一种服务器-客户机协议。其中一个系统（NTP 服务器）为主时钟。所有其他系统为 NTP 客户机，这些客户机将其时钟与主时钟同步。为了同步时钟，NTP 使用 `xntpd` 守护进程，该守护进程设置并维护 UNIX 系统时间，使其与 Internet 标准时间服务器的时间保持一致。下图显示了该服务器-客户机 NTP 实现的示例。

图 4-1 使用 NTP 同步时钟



确保 KDC 与 Kerberos 客户机保持时钟同步需要实现以下步骤：

1. 在网络上设置一个 NTP 服务器。此服务器可以是主 KDC 服务器以外的任何系统。
2. 在网络上配置 KDC 和 Kerberos 客户机时，将它们设置为 NTP 服务器的 NTP 客户机。返回到主 KDC 以将其配置为 NTP 客户机。
3. 在所有系统上启用 NTP 服务。

交换主 KDC 服务器与从 KDC 服务器

采用本节中的过程可以更轻松地交换主 KDC 与从 KDC。仅当主 KDC 服务器由于某种原因出现故障，或者需要重新安装主 KDC 服务器（例如，由于安装了新硬件）时，才应将主 KDC 服务器与从 KDC 服务器进行交换。

▼ 如何配置可交换的从 KDC 服务器

在使用增量传播的领域中，在想要使其成为主 KDC 的从 KDC 服务器上执行此过程。

开始之前 您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **在安装 KDC 过程中，使用主 KDC 服务器和可交换从 KDC 服务器的别名。**
   定义 KDC 的主机名时，请确保 DNS 中包含每个系统的别名。此外，在 `/etc/krb5/krb5.conf` 文件中定义主机时也应使用别名。

2. **安装从 KDC。**
   在进行任何交换之前，该服务器应充当领域中的任何其他从 KDC。有关更多信息，请参见如何手动配置从 KDC 服务器 [72]。

3. **安装后，移动主 KDC 命令。**
   不得从该从 KDC 运行主 KDC 命令。

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

▼ 如何交换主 KDC 服务器与从 KDC 服务器

在此过程中，要交换出来的主 KDC 服务器名为 `kdc1`。将成为新的主 KDC 服务器的从 KDC 服务器名为 `kdc4`。该过程假定使用了增量传播。

开始之前 完成如何配置可交换的从 KDC 服务器 [114]中所述的过程。

您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 在新的主 KDC 服务器上，启动 **kadmin**。

```
kdc4 # /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin:
```

2. 为 **kadmind** 服务创建新的主体。

以下示例中将第一个 addprinc 命令显示为两行，但实际上应在同一行中键入该命令。

```
kadmin: addprinc -randkey -allow_tgs_req +password_changing_service -clearpolicy \
changepw/kdc4.example.com
Principal "changepw/kdc4.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey -allow_tgs_req -clearpolicy kadmin/kdc4.example.com
Principal "kadmin/kdc4.example.com@EXAMPLE.COM" created.
kadmin:
```

3. 退出 **kadmin**。

```
kadmin: quit
```

4. 在新的主 KDC 服务器上，强制执行同步。

以下步骤将在从服务器上强制执行完全 KDC 更新。

a. 禁用 **krb5kdc** 服务并删除其日志文件。

```
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
```

b. 验证更新是否已完成。

```
kdc4 # /usr/sbin/kproplog -h
```

c. 重新启动 KDC 服务。

```
kdc4 # svcadm enable -r network/security/krb5kdc
```

d. 重新初始化新的主 KDC 服务器的更新日志。

```
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
```

5. 在旧的主 KDC 上，中止 **kadmin** 和 **krb5kdc** 两个服务。

中止 kadmin 服务可防止对 KDC 数据库进行任何更改。

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

6. 在旧的主 KDC 服务器上，指定请求传播的轮询时间。
   注释掉 `/etc/krb5/kdc.conf` 中的 `sunw_dbprop_master_ulogsize` 项，并添加定义从服务器的轮询间隔的项。该项将轮询时间设置为两分钟。

```
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
sunw_dbprop_enable = true
#  sunw_dbprop_master_ulogsize = 1000
sunw_dbprop_slave_poll = 2m
}
```

7. 在旧的主 KDC 服务器上，移动主 KDC 服务器命令和 **kadm5.acl** 文件。
   不得从旧的主 KDC 运行主 KDC 命令。

```
kdc1 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc1 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc1 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
kdc1 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
```

8. 在 DNS 服务器上，更改主 KDC 服务器的别名。
   要更改服务器，请编辑 `example.com` 区域文件并更改 `masterkdc` 的项。

```
masterkdc IN CNAME kdc4
```

9. 在 DNS 服务器上，重新装入新的别名信息。

```
# svcadm refresh network/dns/server
```

10. 在新的主 KDC 服务器上，移动主 KDC 服务器命令和从 **kpropd.acl** 文件。
    已在如何配置可交换的从 KDC 服务器 [114]的步骤 3 中移动了主 KDC 命令。

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

11. 在新的主 KDC 上，创建 Kerberos 访问控制列表文件 (**kadm5.acl**)。

填充后，/etc/krb5/kadm5.acl 文件应包含所有获许管理 KDC 的主体名称。该文件还应列出可以请求增量传播的所有从服务器。有关更多信息，请参见 kadm5.acl(4) 手册页。

```
kdc4 # pfedit /etc/krb5/kadm5.acl
kws/admin@EXAMPLE.COM   *
kiprop/kdc1.example.com@EXAMPLE.COM p
```

12. 在新的主 KDC 上，在 **kdc.conf** 文件中指定更新日志大小。

    注释掉 sunw_dbprop_slave_poll 项，并添加定义 sunw_dbprop_master_ulogsize 的项。该项将日志大小设置为 1000 个项。

```
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
sunw_dbprop_enable = true
#               sunw_dbprop_slave_poll = 2m
sunw_dbprop_master_ulogsize = 1000
}
```

13. 在新的主 KDC 上，启用 **kadmin** 和 **krb5kdc** 两个服务。

```
kdc4 # svcadm enable -r network/security/krb5kdc
kdc4 # svcadm enable -r network/security/kadmin
```

14. 在旧的主 KDC 服务器上，添加 **kiprop** 服务主体。

    通过将 kiprop 主体添加到 krb5.keytab 文件，允许 kpropd 守护进程对自身进行增量传播服务验证。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Authenticating as principal kws/admin@EXAMPLE.COM with password.
Enter password: xxxxxxxx
kadmin: ktadd kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.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 kiprop/kdc1.example.com with kvno 3,
encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

15. 在旧的主 KDC 上，在传播配置文件中为 **krb5.conf** 文件中的每个 KDC 添加一个项。

```
kdc1 # pfedit /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
host/kdc4.example.com@EXAMPLE.COM
```

16. 在旧的主 KDC 上，启用 **kpropd** 和 **krb5kdc** 两个服务。

```
kdc1 # svcadm enable -r network/security/krb5_prop
kdc1 # svcadm enable -r network/security/krb5kdc
```

管理 Kerberos 数据库

Kerberos 数据库是 Kerberos 的主干，必须正确维护。本节介绍有关管理 Kerberos 数据库的一些过程，例如备份和恢复数据库、设置增量传播或并行传播以及管理存储文件。如何手动配置主 KDC 服务器 [68]中介绍了该数据库的初始设置步骤。

备份和传播 Kerberos 数据库

将 Kerberos 数据库从主 KDC 服务器传播到从 KDC 服务器是最重要的配置任务之一。如果传播发生频率不够高，主 KDC 服务器与从 KDC 服务器将无法保持同步。因此，如果主 KDC 服务器关闭，从 KDC 服务器将不能获取最新的数据库信息。此外，如果出于负载平衡目的将从 KDC 服务器配置为主 KDC 服务器，则将该从 KDC 服务器用作主 KDC 服务器的客户机将不能获取最新的信息。

所以，必须确保传播发生频率足够高，或者基于更改 Kerberos 数据库的频率配置服务器进行增量传播。应首选增量传播，而不是其他传播方法。数据库手动传播需要更多管理开销，而完全传播效率低下。

注意 - 如果在计划的定期传播之前向 Kerberos 数据库添加了重要更新，应手动传播该数据库，以避免数据丢失。

kpropd.acl 文件

从 KDC 服务器上的 `kpropd.acl` 文件提供 host 主体名称的列表（每个名称占一行），用于指定该 KDC 可以通过传播从其接收更新数据库的系统。如果使用主 KDC 服务器传播到所有从 KDC 服务器，则每个从 KDC 服务器上的 `kpropd.acl` 文件仅需包含主 KDC 服务器的 host 主体名称。

但是，本指南中所述的 Kerberos 安装和后续配置步骤将引导您在主 KDC 和从 KDC 上使用同一个 kpropd.acl 文件。该文件包含所有 KDC host 主体名称。通过此配置，在传播端 KDC 临时不可用时，可以从任何 KDC 进行传播。在所有 KDC 上使用完全相同的副本可方便维护。

kprop_script 命令

kprop_script 命令使用 kprop 命令将 Kerberos 数据库传播到其他 KDC。如果在从 KDC 服务器上运行 kprop_script 命令，则会将该从 KDC 服务器的 Kerberos 数据库副本传播到其他 KDC。kprop_script 接受主机名列表作为参数，该列表以空格分隔，表示要传播的 KDC。

运行 kprop_script 时，将在 /var/krb5/slave_datatrans 文件中创建 Kerberos 数据库的备份，并将该文件复制到指定的 KDC。在完成传播之前，Kerberos 数据库处于锁定状态。

备份 Kerberos 数据库

配置主 KDC 服务器时，可以在 cron 作业中设置 kprop_script 命令以自动将 Kerberos 数据库备份到 /var/krb5/slave_datatrans 转储文件，并将该数据库传播到从 KDC 服务器。不过，与其他任何文件一样，Kerberos 数据库可能会损坏。从 KDC 服务器上发生数据损坏不是问题，因为下一次数据库自动传播会安装一个全新的副本。但是，如果在主 KDC 服务器上发生数据损坏，则在下一次传播期间会将损坏的数据库传播到所有从 KDC 服务器。损坏的备份还会覆盖主 KDC 上先前未损坏的备份文件。

要防止出现这种情况，请设置一个 cron 作业定期将 slave_datatrans 转储文件复制到其他位置，或使用 kdb5_util 的 dump 命令创建另一个单独的备份副本。这样，即使数据库损坏，也可以使用 kdb5_util 的 load 命令在主 KDC 服务器上恢复最新的备份。

另一个重要注意事项是：数据库转储文件包含主体密钥，因此需要防止未经授权的用户访问该文件。缺省情况下，只有 root 身份才具有读写数据库转储文件的权限。要防止未经授权的访问，请使用 kprop 命令传播数据库转储文件，因为该命令会对传输的数据进行加密。此外，kprop 命令仅将数据传播到从 KDC 服务器，这可以最大程度地降低将数据库转储文件意外发送到未经授权的主机的几率。

例 4-13 手动备份 Kerberos 数据库

可以使用 kdb5_util 命令的 dump 命令备份数据库。在 root 拥有的目录中运行此命令。

```
# /usr/sbin/kdb5_util dump
```

在以下示例中，Kerberos 数据库将备份到名为 dumpfile 的文件。由于指定了 -verbose 选项，备份时会显示每个主体。由于未指定任何主体，因此将备份整个数据库。

```
# kdb5_util dump -verbose /var/user/kadmin/dumpfile
```

```
kadmin/kdc1.corp.example.com@CORP.EXAMPLE.COM
krbtgt/CORP.EXAMPLE.COM@CORP.EXAMPLE.COM
kadmin/history@CORP.EXAMPLE.COM
pak/admin@CORP.EXAMPLE.COM
pak@CORP.EXAMPLE.COM
changepw/kdc1.corp.example.com@CORP.EXAMPLE.COM
```

在以下示例中，转储仅包含 pak 和 pak/admin 两个主体。

```
# kdb5_util dump -verbose pakfile pak/admin@CORP.EXAMPLE.COM pak@CORP.EXAMPLE.COM
pak/admin@CORP.EXAMPLE.COM
pak@CORP.EXAMPLE.COM
```

有关更多信息，请参见 kdb5_util(1M) 手册页。

▼ 如何恢复 Kerberos 数据库的备份

开始之前 在 KDC 主服务器上，您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 在主 KDC 服务器上，停止 KDC 守护进程。

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

2. 使用 **kdb_util** 命令 的 **load** 命令恢复 Kerberos 数据库。
   例如，装入在例 4-13 “手动备份 Kerberos 数据库” 中创建的 dumpfile 备份。

```
# /usr/sbin/kdb5_util load /var/user/kadmin/dumpfile
```

3. 启动 KDC 守护进程。

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

例 4-14 恢复 Kerberos 数据库

在以下示例中，将从 dumpfile 文件将名为 database1 的数据库恢复到当前目录。由于未指定 -update 选项，因此将创建新数据库。

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

▼ 如何在服务升级后转换 Kerberos 数据库

如果 KDC 数据库是在运行旧发行版的服务器上创建的，转换该数据库可让您利用增强的数据库格式。

开始之前 仅当数据库使用的是旧格式时才使用此过程。

在 KDC 主服务器上，您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 在主 KDC 服务器上，停止 KDC 守护进程。

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

2. 创建一个目录用来存储数据库的临时副本。

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

3. 转储 KDC 数据库。

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

4. 保存当前数据库文件的副本。

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

5. 装入数据库。

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

6. 启动 KDC 守护进程。

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

▼ 如何重新配置主 KDC 服务器以使用增量传播

此过程中的步骤可用于重新配置现有的主 KDC 服务器，以使用增量传播。此过程使用以下配置参数：

- 领域名称 = EXAMPLE.COM
- DNS 域名 = example.com
- 主 KDC = kdc1.example.com
- 从 KDC = kdc2.example.com
- admin 主体 = kws/admin

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 向 **kdc.conf** 中添加项。

   需要启用增量传播，并选择主 KDC 服务器在日志中记录的更新的数目。有关更多信息，请参见 kdc.conf(4) 手册页。

```
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
sunw_dbprop_enable = true
sunw_dbprop_master_ulogsize = 1000
}
```

2. 创建 **kiprop** 主体。

   kiprop 主体用于验证主 KDC 服务器以及授权来自主 KDC 服务器的更新。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

3. 在主 KDC 服务器上，向 **kadm5.acl** 中添加 kiprop 项。

   通过此项，主 KDC 服务器可以接收来自 kdc2 服务器的增量传播请求。

```
kdc1 # pfedit /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

4. 注释掉 **root** crontab 文件中的 **kprop** 行。

   此步骤禁止从 KDC 服务器传播其 KDC 数据库副本。

```
kdc1 # crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
```

```
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com
```

5. 重新启动 **kadmind**。

```
kdc1 # svcadm restart network/security/kadmin
```

6. 重新配置所有使用增量传播的从 KDC 服务器。

   有关完整说明，请参见如何重新配置从 KDC 服务器以使用增量传播 [123]。

▼ 如何重新配置从 KDC 服务器以使用增量传播

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 向 **kdc.conf** 中添加项。

   第一个新项将启用增量传播。第二个新项将轮询时间设为两分钟。

```
kdc2 # 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
sunw_dbprop_enable = true
sunw_dbprop_slave_poll = 2m
}
```

2. 将 **kiprop** 主体添加到 **krb5.keytab** 文件中。

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.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 kiprop/kdc2.example.com with kvno 3,
encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
```

```
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

3. 重新启动 **kpropd**。

```
kdc2 # svcadm restart network/security/krb5_prop
```

▼ 如何验证 KDC 服务器是否已同步

如果配置了增量传播，则此过程可确保已更新有关从 KDC 服务器的信息。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 在主 KDC 服务器上，运行 **kproplog** 命令。

```
kdc1 # /usr/sbin/kproplog -h
```

2. 在从 KDC 服务器上，运行 **kproplog** 命令。

```
kdc2 # /usr/sbin/kproplog -h
```

3. 检查最后一个序列号和最后一个时间戳的值是否匹配。

例 4-15 验证 KDC 服务器是否同步

以下是在主 KDC 服务器上运行 kproplog 命令的结果样例。

```
kdc1 # /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
Log version #: 1
Log state: Stable
Entry block size: 2048
Number of entries: 2500
First serial #: 137966
Last serial #: 140465
First time stamp: Wed Dec 4 00:59:27 2013
Last time stamp: Wed Dec 4 01:06:13 2013
```

以下是在从 KDC 服务器上运行 kproplog 命令的结果样例。

```
kdc2 # /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
Log version #: 1
```

```
Log state: Stable
Entry block size: 2048
Number of entries: 0
First serial #: None
Last serial #: 140465
First time stamp: None
Last time stamp: Wed Dec 4 01:06:13 2013
```

请注意，最后一个序列号和最后一个时间戳的值相同，表示从 KDC 服务器与主 KDC 服务器同步。

在从 KDC 服务器的输出中，可以注意到从 KDC 服务器的更新日志中不存在任何更新项。这是因为与主 KDC 服务器不同，从 KDC 服务器不保留更新。此外，由于第一个序列号或第一个时间戳不是相关信息，因此从 KDC 服务器不包含这些信息。

手动将 Kerberos 数据库传播到从 KDC 服务器

通常，`cron` 作业可将 Kerberos 数据库传播到从 KDC。如果需要在定期执行的 `cron` 作业之外的其他时间将从 KDC 与主 KDC 同步，可选择使用以下两个命令：`/usr/lib/krb5/kprop_script` 和 `kprop`。有关更多信息，请查看脚本和 kprop(1M) 手册页。

注意 - 如果在从 KDC 上启用了增量传播，请勿使用这些命令。

▼ 如何手动将 Kerberos 数据库传播到从 KDC 服务器

1. 确保未在从 KDC 上启用增量传播。

```
slave# grep sunw_dbprop_enable /etc/krb5/kdc.conf
sunw_dbprop_enable = true
```

2. 如果值为 **true**，则禁用增量传播并重新启动 **krb5_prop** 服务。

```
slave# cp /etc/krb5/kdc.conf /etc/krb5/kdc.conf.sav
slave# pfedit /etc/krb5/kdc.conf
...
sunw_dbprop_enable = false
...

slave# svcadm restart krb5_prop
```

3. 在主 KDC 上，使用以下命令之一将主 KDC 数据库传播到从 KDC。

   - **kprop_script** 命令将先备份数据库，然后同步从 KDC。

     ```
     master# /usr/lib/krb5/kprop_script slave-KDC
     ```

- **kprop** 命令会传播当前数据库备份，而不先创建 Kerberos 数据库的新备份。

  ```
  master# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
  ```

4. （可选） 手动传播完成后，恢复原始的 **krb5.conf** 文件。

```
slave# mv /etc/krb5/kdc.conf.sav /etc/krb5/kdc.conf
```

为 Kerberos 设置并行传播

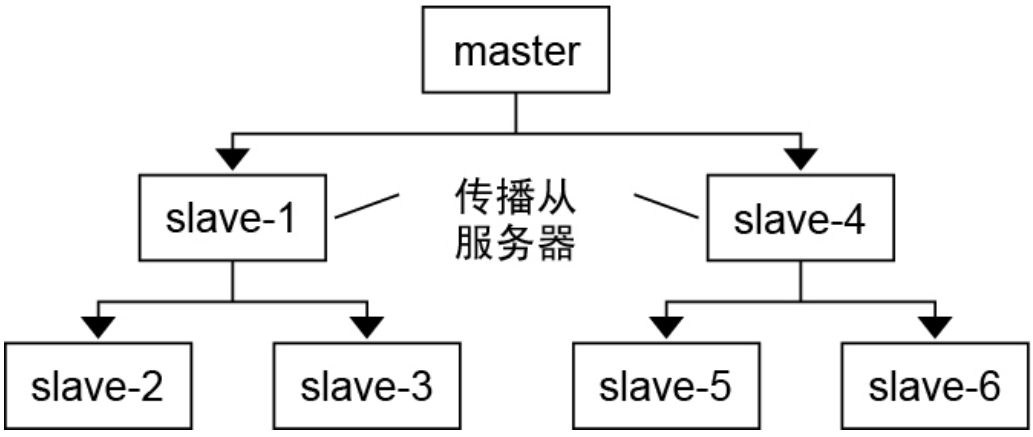
大多数情况下，会以独占的方式使用主 KDC 服务器将其 Kerberos 数据库传播到从 KDC 服务器。但是，如果站点上有很多从 KDC 服务器，可以考虑分担传播过程的负荷，称为并行传播。

注意 - 如果正在使用增量传播，请勿配置并行传播。

通过并行传播，特定的从 KDC 服务器可以与主 KDC 服务器分担传播工作负荷。通过这种分担，可以更快地完成传播并减轻主 KDC 服务器的工作负荷。

例如，假设站点上有一个主 KDC 和六个从 KDC（如图 4-2 “Kerberos 中的并行传播配置示例” 所示），其中，slave-1 到 slave-3 组成一个逻辑组，slave-4 到 slave-6 组成另一个逻辑组。要设置并行传播，可以使主 KDC 将数据库传播到 slave-1 和 slave-4。然后，这些从 KDC 服务器又可将该数据库传播到自己组中的其他从 KDC 服务器。

图 4-2 Kerberos 中的并行传播配置示例



设置并行传播的配置步骤

启用并行传播的概要配置步骤如下：

1. 在主 KDC 服务器上，更改其 `cron` 作业中的 `kprop_script` 项，以仅包括将执行后续传播的从 KDC 服务器（从 *KDC* 传播服务器）的参数。
2. 在每个从 KDC 传播服务器上，将 `kprop_script` 项添加到其 `cron` 作业中，其中必须包括要传播的从 KDC 服务器的参数。要成功实现并行传播，应设置 `cron` 作业，使其在将新 Kerberos 数据库传播到从 KDC 传播服务器本身之后再运行。

注 - 从 KDC 传播服务器进行传播所需的时间取决于多种因素，例如网络带宽和 Kerberos 数据库的大小。

---

3. 在每个从 KDC 上，通过以下方法设置要传播的相应权限：将传播 KDC 的 host 主体名称添加到 `kpropd.acl` 文件。

例 4-16 在 Kerberos 中设置并行传播

以图 4-2 “Kerberos 中的并行传播配置示例” 为例，主 KDC 服务器的 `kprop_script` 项类似于以下示例：

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

`slave-1` 的 `kprop_script` 项类似于以下示例：

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

请注意，从 KDC 服务器上的传播在主 KDC 服务器将数据库传播到它一小时后开始。

从 KDC 传播服务器上的 `kpropd.acl` 文件将包含以下项：

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

要从 `slave-1` 传播到的从 KDC 服务器上的 `kpropd.acl` 文件将包含以下项：

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

管理 Kerberos 数据库的存储文件

存储文件包含 Kerberos 数据库的主密钥，该密钥在创建 Kerberos 数据库时自动创建。如果存储文件损坏，可以使用 `kdb5_util` 实用程序的 `stash` 命令替换损坏的文件。仅在使用 `kdb5_util` 的 `destroy` 命令删除 Kerberos 数据库后，才需要删除存储文件。由于存储文件不会自动随数据库一起删除，所以您必须手动删除存储文件。

要删除存储文件，请使用 `rm` 命令：

```
# rm stash-file
```

其中，*stash-file* 是存储文件的路径。缺省情况下，存储文件位于 `/var/krb5/.k5.`*realm*。

注 - 如果需要重新创建存储文件，可以使用 `kdb5_util` 命令的 `-f` 选项。

▼ 如何为 Kerberos 数据库创建、使用和存储新的主密钥

开始之前 您必须承担 `root` 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **创建一个新的密钥。**

   此命令将添加一个新的随机生成的主密钥。`-s` 选项要求将新的主密钥存储在缺省的密钥表中。

```
# kdb5_util add_mkey -s

Creating new master key for master key principal 'K/M@EXAMPLE.COM'
You will be prompted for a new 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
```

2. **验证新的主密钥是否存在。**

```
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KNVO: 2, Enctype: AES-256 CTS mode with 96-bit SHA-1 HMAC, No activate time set
KNVO: 1, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC, Active on: Fri Dec 31 18:00:00 CST
 2011 *
```

   此输出中的星号标识当前处于活动状态的主密钥。

3. **设置新创建的主密钥变为活动状态的时间。**

```
# date
Fri Jul 11 17:57:00 CDT 2014
# kdb5_util use_mkey 2 'now+2days'
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KNVO: 2, Enctype: AES-256 CTS mode with 96-bit SHA-1 HMAC,
Active on: Sun Jul 13 17:57:15 CDT 2014
KNVO: 1, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC,
Active on: Fri Dec 31 18:00:00 CST 2011 *
```

在下面的示例中，日期设置为未来的两天，这样新的主密钥便有时间传播到所有 KDC。调整日期，使其适合您的环境。

4. （可选） 创建新的主体后，确保正在使用新的主密钥。

```
# kadmin.local -q 'getprinc tamiko' |egrep 'Principal|MKey'
Authenticating as principal root/admin@EXAMPLE.COM with password.
Principal: tamiko@EXAMPLE.COM
MKey: vno 2
```

在下面的示例中，MKey: vno 2 指示主体的密钥受新创建的主密钥 (2) 的保护。

5. 使用新的主密钥重新加密用户主体密钥。

如果将模式参数添加到命令的末尾，将会更新与此模式相匹配的主体。向此命令语法添加 -n 选项，以标识将要更新的主体。

```
# kdb5_util update_princ_encryption -f -v
Principals whose keys WOULD BE re-encrypted to master key vno 2:
updating: host/kdc1.example.com@EXAMPLE.COM
skipping:   tamiko@EXAMPLE.COM
updating: kadmin/changepw@EXAMPLE.COM
updating: kadmin/history@EXAMPLE.COM
updating: kdc/admin@EXAMPLE.COM
updating: host/kdc2.example.com@EXAMPLE.COM
6 principals processed: 5 updated, 1 already current
```

6. 清除旧的主密钥。

当某个主密钥不再用于保护任何主体密钥时，可将其从主密钥主体中清除。此命令不会清除那些仍被主体使用的密钥。向此命令添加 -n 选项，以确保将清除正确的主密钥。

```
# kdb5_util purge_mkeys -f -v
Purging the follwing master key(s) from K/M@EXAMPLE.COM:
KNVO: 1
1 key(s) purged.
```

7. 验证是否已清除旧的主密钥。

```
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KNVO: 2, Enctype: AES-256 CTS mode with 96-bit SHA-1 HMAC,
Active on: Sun Jul 13 17:57:15 CDT 2014 *
```

8. 更新存储文件。

```
# kdb5_util stash
Using existing stashed keys to update stash file.
```

9. 验证存储文件是否已更新。

```
# klist -kt /var/krb5/.k5.EXAMPLE.COM
```

```
Keytab name: FILE:.k5.EXAMPLE.COM
KVNO Timestamp             Principal
---- --------------- ------------------------------------------------------
2 05/11/2014 18:03 K/M@EXAMPLE.COM
```

增强 Kerberos 服务器的安全性

本节提供有关增强 Kerberos 应用服务器和 KDC 服务器的安全性的建议。

限制对 KDC 服务器的访问

主 KDC 服务器和从 KDC 服务器都包含存储在本地的 KDC 数据库的副本。限制对这些服务器的访问以保证数据库安全对于 Kerberos 安装的整体安全性非常重要。

- 限制对支持 KDC 的硬件的实地访问。

  确保 KDC 服务器及其监视器位于安全场所。禁止一般用户以任何方法接近此服务器。
- 将 KDC 数据库备份存储在本地磁盘或从 KDC 服务器上。

  仅在可以安全存储磁带时才应创建 KDC 的磁带备份。此做法同样适用于密钥表文件的备份。

  推荐的做法是将这些文件存储在不与其他系统共享的本地文件系统上。存储文件系统可以位于主 KDC 服务器或任何从 KDC 服务器上。

使用字典文件提高口令的安全性

Kerberos 服务可以使用字典文件来防止将字典中的词用作新凭证的口令。防止将字典术语用作口令可以让其他人更难猜到口令。缺省情况下使用 `/var/krb5/kadm5.dict` 文件，但它是空的。

您需要将一行添加到 KDC 配置文件 (`kdc.conf`)，以指示服务使用字典文件。在本示例中，管理员使用 `spell` 实用程序附带的字典，然后重新启动 Kerberos 服务。有关配置文件的完整说明，请参见 `kdc.conf(4)` 手册页。

```
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
sunw_dbprop_enable = true
sunw_dbprop_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
```

♦ ♦ ♦ 第 5 章

5 管理 Kerberos 主体和策略

本章介绍有关管理主体及与其关联的策略的过程。本章还将说明如何管理主机的密钥表文件。

本章包含以下主题：

有关主体和策略的背景信息，请参见第 2 章 关于 Kerberos 服务和第 3 章 规划 Kerberos 服务。

管理 Kerberos 主体和策略的方法

主 KDC 上的 Kerberos 数据库包含您所在领域的所有 Kerberos 主体、主体口令、策略和其他管理信息。创建和删除主体以及修改其属性时，可以使用 `kadmin` 或 `gkadmin` 命令。

`kadmin` 命令提供一个交互式命令行接口，用于维护 Kerberos 主体、策略和密钥表文件。也可以运行脚本自动创建主体。`kadmin` 命令具有本地版本和远程版本：

- `kadmin` – 使用 Kerberos 验证从网络中的任何位置安全操作
- `kadmin.local` – 必须直接在主 KDC 上运行

这两种版本的功能完全相同。必须使用本地版本对数据库进行充分配置，然后才能使用远程版本。

Oracle Solaris 发行版还提供一个交互式图形用户界面 (graphical user interface, GUI) `gkadmin`。

本节介绍 `kadmin.local` 命令的脚本功能，并比较命令行界面和 GUI 界面。

自动创建新的 Kerberos 主体

可以在脚本中使用 kadmin.local 命令，以自动创建新的 Kerberos 主体。如果要将许多新主体添加到数据库，自动化将很有用。

以下 shell 脚本行显示了如何自动创建新主体：

```
awk '{ print "ank +needchange -pw", $2, $1 }' < /tmp/princnames |
time /usr/sbin/kadmin.local> /dev/null
```

为了方便阅读，以上示例拆分为两行。

- 脚本读取名为 princnames 的文件。此文件包含主体名称及其口令，并将它们添加到 Kerberos 数据库。

  您必须创建 princnames 文件，并在每一行上包含一个主体名称及其口令，中间用一个或多个空格分隔。
- ank 命令添加一个新密钥。ank 是 add_principal 命令的别名。
- +needchange 选项用于配置主体，以便在属于主体的用户首次登录时提示其提供新口令。

  要求更改口令有助于确保 princnames 文件中的口令不会引入安全风险。

可以生成更详细的脚本。例如，脚本可使用名称服务中的信息来获取主体名称的用户名列表。所执行的操作和执行操作的方式取决于站点的需要以及脚本编制技术。

gkadmin GUI

Kerberos gkadmin GUI 提供了 CLI 的大部分功能，并提供联机帮助。下表介绍 CLI 与 GUI 之间的差别。

表 5-1 gkadmin GUI 的命令行等效项

gkadmin GUI 过程	等效的 kadmin 命令
查看主体列表。	list_principals 或 get_principals
查看主体属性。	get_principal
创建新主体。	add_principal
复制主体。	无命令行等效项
修改主体。	modify_principal 或 change_password
删除主体。	delete_principal
设置缺省值以创建新主体。	无命令行等效项
查看策略列表。	list_policies 或 get_policies
查看策略属性。	get_policy

gkadmin GUI 过程	等效的 **kadmin** 命令
创建新策略。	`add_policy`
复制策略。	无命令行等效项
修改策略。	`modify_policy`
删除策略。	`delete_policy`

Kerberos `gkadmin` GUI 提供与上下文相关的帮助。对于浏览器访问，请使用 Oracle URL (http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html)。可以将此文件复制到站点 URL。配置使用 `gkadmin` GUI 的主机时，请在 `krb5.conf` 文件中指定联机帮助 URL。

管理 Kerberos 主体

本节提供有关使用 `kadmin` 命令和 `gkadmin` GUI 管理主体的示例。有关更多信息，请参见 `kadmin(1M)` 和 `gkadmin(1M)` 手册页。

查看 Kerberos 主体及其属性

以下示例显示了如何列出主体及其属性。可以使用通配符来构造列表。有关可用通配符的信息，请查看 `kadmin(1M)` 手册页中的表达式定义。

例 5-1 查看 Kerberos 主体

本示例使用 `list_principals` 子命令列出所有与 `kadmin*` 匹配的主体。如果未指定参数，`list_principals` 将列出在 Kerberos 数据库中定义的所有主体。

```
# /usr/sbin/kadmin
kadmin: list_principals kadmin*
kadmin/changepw@EXAMPLE.COM
kadmin/kdc1.example.com@EXAMPLE.COM
kadmin/history@EXAMPLE.COM
```

例 5-2 查看 Kerberos 主体的属性

以下示例显示了 `jdb/admin` 主体的属性。

```
kadmin: get_principal jdb/admin
Principal: jdb/admin@EXAMPLE.COM
```

```
Expiration date: [never]
Last password change: [never]

Password expiration date: Fri Sep 13 11:50:10 PDT 2013
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
Last modified: Thu Aug 15 13:30:30 PST 2013 (host/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, AES-256 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, AES-128 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, Triple DES with HMAC/sha1, no salt
Key: vno 1, ArcFour with HMAC/md5, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

例 5-3 使用 gkadmin GUI 列出 Kerberos 主体并为其设置缺省值

在本示例中，管理员希望向新管理员显示主体及其属性的列表，因此使用 gkadmin GUI。管理员还为将来的主体设置新的缺省值。

```
# /usr/sbin/gkadmin
```

窗口将显示 "Principal Name"（主体名称）、"Password"（口令）、"Realm"（领域）和 "Master KDC"（主 KDC）等字段。

管理员浏览至所有主体名称的列表，然后向新管理员展示如何使用区分大小写的过滤器。

接着，管理员单击 "Edit"（编辑）菜单并选择 "Properties"（属性）。管理员单击 "Require Password Change"（要求口令更改）后，所做更改随即应用。

管理员浏览至主体的列表并从中选择一个主体，以查看当前主体的属性。第一个对话框显示基本属性。管理员单击 "Next"（下一页）按钮以显示所有属性。

创建新的 Kerberos 主体

如果新主体需要新策略，则必须先创建新策略，然后再创建主体。有关策略创建，请参见例 5-10 “创建新的 Kerberos 口令策略”。大多数 Kerberos 策略都指定了口令要求。

例 5-4 创建新的 Kerberos 主体

以下示例创建名为 pak 的新主体，并将该主体的策略设置为 testuser。其他必需值（例如加密类型）使用缺省值。

```
# /usr/sbin/kadmin
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": xxxxxxxx
Re-enter password for principal "pak@EXAMPLE.COM": xxxxxxxx
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

通常，只有少数用户有权管理 Kerberos 数据库。如果此新主体需要管理特权，请继续执行“修改主体的 Kerberos 管理特权” [138]中的相关操作。

修改 Kerberos 主体

以下示例显示了如何修改 Kerberos 主体的口令属性。

例 5-5 修改 Kerberos 主体的口令重试最大次数

```
# /usr/sbin/kadmin
kadmin: modify_principal jdb
kadmin: maxtries=5
kadmin: quit
```

例 5-6 修改 Kerberos 主体的口令

```
# /usr/sbin/kadmin
kadmin: change_password jdb
Enter password for principal "jdb": xxxxxxxx
Re-enter password for principal "jdb": xxxxxxxx
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

删除 Kerberos 主体

以下示例显示了如何删除主体。按照联机消息的提示继续操作。

```
# /usr/sbin/kadmin
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
kadmin: quit
```

要将此主体从所有 ACL 中删除，请参见“修改主体的 Kerberos 管理特权” [138]。

使用 gkadmin GUI 复制 Kerberos 主体

可以使用 gkadmin GUI 复制主体。此过程没有等效命令行。

1. 首先在 GUI 中键入 /usr/sbin/gkadmin 命令，然后单击 "Principals"（主体）选项卡并选择要复制的主体。
2. 单击 "Duplicate"（复制）按钮。此操作将复制所选主体的所有属性，主体名称和口令除外。
3. 指定新名称和口令，然后单击 "Save"（保存）以创建完全一样的副本。
4. 要修改该副本，请指定主体属性的值。

   有三个窗口包含属性信息。请从 "Help"（帮助）菜单中选择 "Context-Sensitive Help"（上下文相关帮助），获取有关每个窗口中各种属性的信息。

通常，只有少数用户有权管理 Kerberos 数据库。如果此新主体需要管理特权，请继续执行“修改主体的 Kerberos 管理特权” [138]中的相关操作。

修改主体的 Kerberos 管理特权

有权管理 Kerberos 数据库的少数用户在 Kerberos 访问控制列表 (Access Control List, ACL) 中指定。此列表作为 /etc/krb5/kadm5.acl 文件中的项进行维护。有关更多信息，请参见 kadm5.acl(4) 手册页。

要将项添加到 kadm5.acl 文件，请使用 pfedit 命令。

```
# pfedit /usr/krb5/kadm5.acl
```

kadm5.acl 文件中的项采用以下格式：

principal privileges [*principal-target*]

- *principal* – 指定要授予特权的主体。主体名称的任何部分都可以包含 "*" 通配符，这在为一组主体提供相同特权时很有用。例如，如果要指定包含 admin 实例的所有主体，可使用 */admin@*realm*。

  请注意，admin 实例常用于将单独的特权（如对 Kerberos 数据库的管理访问权限）授予单独的 Kerberos 主体。例如，用户 jdb 可能有一个供管理使用的主体 jdb/admin。通过创建两个主体，用户 jdb 只会在需要管理特权时获取 jdb/admin 票证。
- *privileges* – 指定主体可执行的操作。此字段由包含下面列出的一个或多个字符的字符串组成。如果字符为大写或未指定，则不允许执行该操作。如果字符为小写，则允许执行该操作。
  - [A]a – [不]允许添加主体或策略。
  - [C]c – [不]允许更改主体的口令。

- [D]d – [不]允许删除主体或策略。
- [I]i – [不]允许查询 Kerberos 数据库。
- [L]l – [不]允许列出主体或策略。
- [M]m – [不]允许修改主体或策略。
- x 或 * – [不]允许所有特权 (admcil)。

- *principal-target* – 在此字段中指定一个主体时，主体的特权仅应用到该主体。要为一组主体分配特权，请在 *principal-target* 中使用 "*" 通配符。

例 5-7 修改 Kerberos 主体的特权

kadm5.acl 文件中的以下条目授予 EXAMPLE.COM 领域中包含 admin 实例的任何主体对 Kerberos 数据库的所有特权：

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

kadm5.acl 文件中的以下项授予 jdb@EXAMPLE.COM 主体列出和查询包含 root 实例的任何主体的特权。

```
jdb@EXAMPLE.COM li */root@EXAMPLE.COM
```

管理 Kerberos 策略

本节提供有关使用 kadmin 命令和 gkadmin GUI 管理 Kerberos 策略的示例。大多数 Kerberos 策略都指定了口令要求。

管理策略的步骤与管理主体的步骤相似。有关更多信息，请参见 kadmin(1M) 和 gkadmin(1M) 手册页。

例 5-8 查看 Kerberos 策略的列表

本示例使用 list_policies 子命令列出所有与 *user* 匹配的策略。如果未指定参数，list_policies 将列出在 Kerberos 数据库中定义的所有策略。

```
# kadmin
kadmin: list_policies *user*
testuser
financeuser
kadmin: quit
```

例 5-9 查看 Kerberos 策略的属性

本示例使用 get_policy 子命令查看 financeuser 策略的属性。

```
# /usr/sbin/kadmin.local
kadmin.local: get_policy financeuser
Policy: financeuser
Maximum password life: 13050000
Minimum password life: 10886400
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
Reference count: 8
Maximum password failures before lockout: 5
Password failure count reset interval: 200
Password lockout duration: 300
kadmin: quit
```

"Reference count"（引用计数）是分配了此策略的主体数。

例 5-10 创建新的 Kerberos 口令策略

本示例使用 `add_policy` 子命令创建 `build11` 策略。此策略要求口令至少包含三类字符。

```
# kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```

例 5-11 处理 Kerberos 帐户锁定策略

在本示例中，`300` 秒内发生了三次验证失败，从而触发长达 `900` 秒的帐户锁定。

```
kadmin: add_policy -maxfailure 3 -failurecountinterval "300 seconds"\
-lockoutduration "900 seconds" default
```

要在 15 分钟内解除锁定，需要执行管理操作。

```
# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: modify_principal -unlock principal
```

例 5-12 修改 Kerberos 策略

本示例使用 `modify_policy` 子命令将 `build11` 策略的最小口令长度更改为八个字符。

```
# kadmin
kadmin: modify_policy -minlength 8 build11
kadmin: quit
```

例 5-13 删除 Kerberos 策略

本示例使用 `delete_policy` 子命令删除 `build11` 策略。

1. 管理员将该策略从使用该策略的所有主体中删除。

```
# kadmin
kadmin: modify_principal -policy build11 *admin*
```

2. 然后，管理员删除该策略。

```
kadmin: delete_policy build11
Are you sure you want to delete the policy "build11"? (yes/no): yes
kadmin: quit
```

如果该策略仍分配给某个主体，`delete_policy` 命令将失败。

例 5-14 使用 `gkadmin` GUI 复制 Kerberos 策略

在 `gkadmin` GUI 中，可以通过单击 "Duplicate"（复制）按钮复制所选策略。在 "Policy Name"（策略名称）字段中，为新策略命名。还可以修改已复制的策略属性。相关步骤与“使用 `gkadmin` GUI 复制 Kerberos 主体” [138]中所述的步骤相似。

管理密钥表文件

提供服务的每台主机都必须包含一个名为 keytab file（密钥表文件）的本地文件，keytab 是 "key table"（密钥表）的缩写。密钥表包含相应服务的主体，称为 service key（服务密钥）。服务使用服务密钥向 KDC 进行自我验证，并且只有 Kerberos 和服务本身知道服务密钥。例如，如果使用基于 Kerberos 的 NFS 服务器，该服务器必须具有包含 `nfs` 服务主体的服务密钥的密钥表文件。

要将服务密钥添加到密钥表文件，可在 `kadmin` 进程中使用 `ktadd` 命令，将相应的服务主体添加到主机的密钥表文件。要将服务主体添加到密钥表文件，该主体必须已存在于 Kerberos 数据库中。在提供基于 Kerberos 的服务的应用服务器上，密钥表文件的缺省位置为 `/etc/krb5/krb5.keytab`。

密钥表类似于用户的口令。正如用户必须保护其口令一样，应用服务器也必须保护其密钥表文件。应始终将密钥表文件存储在本地磁盘上，并且只允许 `root` 用户读取这些文件。此外，绝不要通过不安全的网络发送密钥表文件。

在某些特殊情况下，可能需要将 `root` 主体添加到主机的密钥表文件。如果希望 Kerberos 客户机用户挂载基于 Kerberos 的 NFS 文件系统（需要与 `root` 等效的访问权限），则必须将客户机的 `root` 主体添加到客户机的密钥表文件。否则，每当用户要使用 `root` 权限挂载基于 Kerberos 的 NFS 文件系统时，即使正在使用自动挂载程序，也必须以 `root` 身份使用 `kinit` 命令来获取客户机的 `root` 主体的凭证。

注意 - 以 `root` 用户身份挂载 NFS 服务器会引入安全风险。

也可以使用 ktutil 命令管理密钥表文件。使用此交互式命令，可在没有 Kerberos 管理特权的情况下管理本地主机的密钥表文件，因为此命令不会像 kadmin 那样与 Kerberos 数据库交互。将主体添加到密钥表文件后，可使用 ktutil 来查看密钥表文件中的密钥列表，或临时禁用对某个服务的验证。

注 - 在 kadmin 进程中使用 ktadd 命令更改密钥表文件中的主体时，将生成一个新的密钥并将其添加到密钥表文件。

将 Kerberos 服务主体添加到密钥表文件

确认某个主体存在于 Kerberos 数据库中后，可将该主体添加到密钥表文件。有关更多信息，请参见“查看 Kerberos 主体及其属性” [135]。

在需要将主体添加到其密钥表文件的主机上，在 kadmin 进程中使用 ktadd 命令。有关更多信息，请参见 kadmin(1M) 手册页。

```
# /usr/sbin/kadmin
kadmin: ktadd [-e enctype] [-k keytab] [-q] [principal | -glob principal-exp]
```

-e *enctype*	覆盖 krb5.conf 文件中定义的加密类型列表。
-k *keytab*	指定密钥表文件。缺省情况下，使用 /etc/krb5/krb5.keytab。
-q	显示简要信息。
principal	指定要添加至密钥表文件的主体。可以添加以下服务主体：host、root、nfs 和 ftp。
-glob *principal-exp*	指定主体表达式。与 *principal-exp* 匹配的所有主体都将添加至密钥表文件。主体表达式的规则与 kadmin 的 list_principals 命令的规则相同。有关可用的表达式，请查看 kadmin(1M) 手册页中的表达式定义。

例 5-15 将服务主体添加至密钥表文件

在本示例中，denver 的 host 主体被添加到 denver 的密钥表文件，以便 KDC 可以验证 denver 的网络服务。

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver.example.com
Entry for principal host/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 host/denver.example.com with kvno 3,
```

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

从密钥表文件中删除服务主体

可以从密钥表文件中删除主体。首先，在需要从其密钥表文件中删除主体的主机上查看主体的列表。请参见“显示密钥表文件中的主体” [143]。

然后在 kadmin 进程中运行 ktadd 命令。有关更多信息，请参见 kadmin(1M) 手册页。

```
# /usr/sbin/kadmin
kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
```

-k *keytab*　　指定密钥表文件。缺省情况下，使用 /etc/krb5/krb5.keytab。

-q　　显示简要信息。

principal　　指定要从密钥表文件删除的主体。

kvno　　删除密钥版本号与 *kvno* 匹配的指定主体的所有项。

all　　删除指定主体的所有项。

old　　删除指定主体（具有最高密钥版本号的主体除外）的所有项。

例 5-16　从密钥表文件中删除服务主体

在本示例中，从 denver 的密钥表文件中删除了 denver 的 host 主体。

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

显示密钥表文件中的主体

在包含密钥表文件的主机上，运行 ktutil 命令，读取密钥表，然后列出主体。主体也称为密钥列表。

注 - 尽管可以创建由其他用户拥有的密钥表文件，但使用密钥表文件的缺省位置需要 root 所有权。

例 5-17 显示密钥表文件中的密钥列表（主体）

以下示例显示了 denver 主机上的 /etc/krb5/krb5.keytab 文件中的密钥列表。

```
denver # /usr/bin/ktutil
ktutil: read_kt /etc/krb5/krb5.keytab
ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------
1    5 host/denver@EXAMPLE.COM
ktutil: quit
```

在主机上临时禁用 Kerberos 服务

在网络应用服务器上，有时可能需要临时禁用某个服务（如 ssh 或 ftp）的验证机制。例如，可能希望在执行维护过程时禁止用户登录到系统。

注 - 缺省情况下，大多数服务都要求验证。如果某个服务不要求验证，则禁用验证不会造成任何影响。该服务仍然可用。

▼ 如何在主机上临时禁用对 Kerberos 服务的验证

ktutil 命令允许不具有 kadmin 特权的用户禁用某个服务。此用户还可以恢复该服务。有关更多信息，请参见 ktutil(1) 手册页。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **将当前密钥表文件保存到临时文件。**
   在步骤 9 中使用此临时文件重新启用验证。

2. **在包含该密钥表文件的主机上，启动 ktutil 命令。**

注 - 尽管可以创建由其他用户拥有的密钥表文件，但使用密钥表文件的缺省位置需要 root 所有权。

```
# /usr/bin/ktutil
```

3. 将密钥表文件读入密钥列表缓冲区。

   ```
   ktutil: read_kt keytab
   ```

4. 显示密钥列表缓冲区。

   ```
   ktutil: list
   ```

   此时会显示当前的密钥列表缓冲区。请记下要禁用的服务的槽号。

5. 通过从密钥列表缓冲区中删除特定服务主体，临时禁用主机的服务。

   ```
   ktutil: delete_entry slot-number
   ```

   其中 *slot-number* 指定要从 `list` 输出中删除的服务主体的槽号。

6. 将密钥列表缓冲区写到新的密钥表文件。

   ```
   ktutil: write_kt new-keytab
   ```

7. 退出 **ktutil** 命令。

   ```
   ktutil: quit
   ```

8. 使用新的密钥表文件禁用主体的验证。

   ```
   # mv new-keytab keytab
   ```

9. （可选） 通过将临时密钥表文件复制回其原始位置，重新启用该服务。

   ```
   # cp original-keytab keytab
   ```

例 5-18 临时禁用 Kerberos host 服务

在本示例中，临时禁用了 `denver` 主机上的 `host` 服务。要在 `denver` 上重新启用 host 服务，管理员可将已保存的密钥表文件复制回其原始位置。

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.save
denver # /usr/bin/ktutil
ktutil:read_kt /etc/krb5/krb5.keytab
ktutil:list
slot KVNO Principal
---- ---- -------------------------------------
1    8 root/denver@EXAMPLE.COM
2    5 host/denver@EXAMPLE.COM
ktutil:delete_entry 2
ktutil:list
slot KVNO Principal
---- ---- -------------------------------------
1    8 root/denver@EXAMPLE.COM
ktutil:write_kt /etc/krb5/nodenverhost.krb5.keytab
ktutil: quit
```

```
denver # cp /etc/krb5/nodenverhost.krb5.keytab /etc/krb5/krb5.keytab
```

用户将已保存的文件复制回其原始位置之前，host 服务将不可用。

```
denver # cp /etc/krb5/krb5.keytab.save /etc/krb5/krb5.keytab
```

♦ ♦ ♦ 第 6 章

使用 Kerberos 应用程序

本章说明如何使用 Oracle Solaris 中提供的基于 Kerberos 的命令和服务。阅读本章中有关这些命令的信息之前，应先熟悉其原始版本。

由于本章适用于应用程序用户，因此其中包含有关票证的以下信息：获取、查看和销毁票证。本章还包含有关选择或更改 Kerberos 口令的信息。

本章包含以下主题：

有关 Kerberos 的概述，请参见第 2 章 关于 Kerberos 服务。

Kerberos 票证管理

本节说明如何获取、查看和销毁票证。有关票证的介绍，请参见“Kerberos 服务的工作原理” [38]。

在 Oracle Solaris 中，Kerberos 内置到 `login` 命令中。不过，为自动获取票证，必须为适用的登录服务配置 PAM 服务。有关更多信息，请参见 `pam_krb5(5)` 手册页。

可以设置 `ssh` 命令将用户票证的副本转发到其他主机，这样用户不必显式请求票证就能获取对这些主机的访问权限。由于安全原因，管理员会阻止执行此操作。有关更多信息，请参见 `ssh(1)` 手册页中有关代理转发的讨论。

有关票证生命周期的信息，请参见“票证生命周期” [159]。

创建 Kerberos 票证

如果 PAM 配置正确，登录时会自动创建票证，您无需执行任何特殊操作即可获得票证。但是，如果票证过期，则可能需要创建票证。另外，除了缺省主体外，您可能还需要使用另一个主体。例如，在使用 `ssh -l` 以其他用户身份登录到系统时。

要创建票证，请使用 kinit 命令。

```
% /usr/bin/kinit
```

kinit 命令将提示您输入口令。有关 kinit 命令的完整语法，请参见 kinit(1) 手册页。

例 6-1 创建 Kerberos 票证

本示例说明用户 kdoe 如何在自己的系统上创建票证。

```
% kinit
Password for kdoe@CORP.EXAMPLE.COM: xxxxxxxx
```

在以下示例中，用户 kdoe 使用 -l 选项创建了一个有效期为三小时的票证。

```
% kinit -l 3h kdoe@EXAMPLE.ORG
Password for kdoe@EXAMPLE.ORG: xxxxxxxx
```

查看 Kerberos 票证

并非所有票证都相同。例如，第一个票证可能是 forwardable，第二个票证可能是 postdated，而第三个票证可能同时是 forwardable 和 postdated。使用带有 -f 选项的 klist 命令，可以查看所拥有的票证以及这些票证的属性：

```
% /usr/bin/klist -f
```

以下符号表示与每个票证关联的属性，如 klist 输出所示：

A	Preauthenticated（已预验证）
D	Postdatable（可以后生效）
d	Postdated（以后生效）
F	Forwardable（可转发）
f	Forwarded（已转发）
I	Initial（初始）
i	Invalid（无效）
P	Proxiable（可代理）
p	Proxy（代理）
R	Renewable（可更新）

“票证类型” [158]介绍了票证可以具有的各种属性。

例 6-2 查看 Kerberos 票证

本示例说明用户 kdoe 拥有一个 initial 票证，该票证是 forwardable (F) 和 postdated (d) 票证，但尚未经过验证 (i)。

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: kdoe@EXAMPLE.COM

Valid starting                Expires                Service principal
09 Feb 14 15:09:51  09 Feb 14 21:09:51  nfs/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 15:12:51, Flags: Fdi
```

以下示例说明 kdoe 拥有两个票证，这些票证是从其他主机 forwarded (f) 到用户主机的票证。这些票证也是 forwardable (F) 票证。

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: kdoe@EXAMPLE.COM

Valid starting                Expires                Service principal
07 Feb 14 06:09:51  09 Feb 14 23:33:51  host/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 17:09:51, Flags: fF

Valid starting                Expires                Service principal
08 Feb 14 08:09:51  09 Feb 14 12:54:51  nfs/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 15:22:51, Flags: fF
```

以下示例说明如何使用 -e 选项显示会话密钥和票证的加密类型。如果名称服务可以执行转换操作，则可使用 -a 选项将主机地址映射为主机名。

```
% klist -fea
Ticket cache: /tmp/krb5cc_74287
Default principal: kdoe@EXAMPLE.COM

Valid starting                Expires                Service principal
07 Feb 14 06:09:51  09 Feb 14 23:33:51  krbtgt/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 17:09:51, Flags: FRIA
Etype(skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC
Addresses: client.example.com
```

销毁 Kerberos 票证

如果要销毁当前会话期间获取的所有 Kerberos 票证，请使用 kdestroy 命令。该命令将销毁凭证高速缓存，从而销毁所有凭证和票证。虽然通常不必销毁凭证高速缓存，但运行 kdestroy 可降低您未登录期间凭证高速缓存遭受破坏的可能性。

要销毁票证，请使用 kdestroy 命令。

```
% /usr/bin/kdestroy
```

kdestroy 命令将销毁所有票证。不能使用此命令来有选择性地销毁特定票证。

如果要离开系统，应使用 kdestroy，或使用屏幕保护程序来锁定屏幕。

Kerberos 口令管理

配置 Kerberos 服务后，您即会拥有两个口令：一个常规 Oracle Solaris 口令和一个 Kerberos 口令。可以将两个口令设置为相同或不同。

更改口令

如果 PAM 配置正确，则可以采用以下两种方法来更改 Kerberos 口令。

- 使用 passwd 命令。配置 Kerberos 服务后，passwd 命令还会自动提示您输入新的 Kerberos 口令。

  通过使用 passwd，可以同时设置 UNIX 口令和 Kerberos 口令。但是，使用 passwd 只能更改一个口令，另一个口令将保持不变。

注 - passwd 的行为取决于 PAM 模块的配置方式。在某些配置中，可能要求您同时更改这两个口令。对于一些站点，必须更改 UNIX 口令，而对于另一些站点，则要求更改 Kerberos 口令。

- 使用 kpasswd 命令。kpasswd 只能更改 Kerberos 口令。如果要更改 UNIX 口令，则必须使用 passwd。

  kpasswd 主要用于更改不是有效 UNIX 用户的 Kerberos 主体的口令。例如，jdoe/admin 是 Kerberos 主体，但不是实际的 UNIX 用户，因此必须使用 kpasswd 来更改口令。

更改口令后，该口令必须在整个网络中传播。传播所需的时间可能为几分钟到一小时或更多，具体取决于 Kerberos 网络的规模。如果需要在更改口令后立即获取新的 Kerberos 票证，请先尝试新口令。如果新口令无效，请使用旧口令重试。

Kerberos 策略用于定义口令的条件。可以为每个用户设置一个策略，也可以应用缺省策略。有关策略的信息，请参见“管理 Kerberos 策略” [139]。有关可为 Kerberos 口令设置的条件的示例，请参见例 5-9 “查看 Kerberos 策略的属性”。口令字符分类包括小写字母、大写字母、数字、标点符号以及其他所有字符。

Kerberos 中的远程登录

就像在 Oracle Solaris 中一样，Kerberos 提供了 `ssh` 命令用于远程登录。安装后，`ssh` 命令是唯一接受网络请求的网络服务。因此，其他针对 Kerberos 进行了修改的网络服务（例如 `rlogin` 和 `telnet`）将被禁用。有关更多信息，请参见“Kerberos 网络程序” [42]和“Kerberos 用户命令” [151]。

Kerberos 用户命令

Kerberos V5 产品是一种单点登录系统，这意味着用户在使用网络应用程序时只需键入一次口令。Kerberos V5 应用程序将对用户执行验证并有选择地进行加密，因为 Kerberos 已内置到用户所熟悉的每个现有网络应用程序套件中。Kerberos V5 应用程序是现有 UNIX 网络应用程序添加了 Kerberos 功能的版本。

注 - 一般来说，`ssh` 应用程序应该能够满足您的远程登录需求。请参见 `ssh(1)` 手册页。有关启用过时的网络应用程序的信息，请参见“Kerberos 网络程序” [42]。

对于现有网络应用程序中的 Kerberos 功能，请参见以下手册页，尤其是其中的“选项”节：

- `ftp(1)`
- `rcp(1)`
- `rlogin(1)`
- `rsh(1)`
- `telnet(1)`

使用基于 Kerberos 的应用程序连接到远程系统时，该应用程序、KDC 和远程系统会执行一组快速协商。完成这些协商后，应用程序会代表您向远程系统证明身份，然后远程主机会向您授予访问权限。

有关已弃用的远程登录命令的示例，请参见《*Oracle Solaris 11.1* 管理：安全服务》中的“使用 Kerberos 应用程序（任务）”。

♦ ♦ ♦ 第 7 章

7 Kerberos 服务参考信息

本章列出了许多属于 Kerberos 产品的文件、命令和守护进程。

本章介绍以下参考信息：

Kerberos 文件

Kerberos 服务使用以下文件。

`~/.gkadmin`	用于在 Kerberos 管理 GUI 中创建新主体的缺省值
`~/.k5login`	用于向 Kerberos 帐户授予访问权限的主体的列表
`/etc/krb5/kadm5.acl`	Kerberos 访问控制列表文件，包括 KDC 管理员的主体名称及其 Kerberos 管理特权
`/etc/krb5/kdc.conf`	KDC 配置文件
`/etc/krb5/kpropd.acl`	Kerberos 数据库传播配置文件
`/etc/krb5/krb5.conf`	Kerberos 领域配置文件
`/etc/krb5/krb5.keytab`	网络应用服务器的密钥表文件
`/etc/krb5/warn.conf`	Kerberos 票证失效警告和自动更新配置文件

`/etc/pam.conf`	PAM 配置文件
`/tmp/krb5cc_`*UID*	缺省凭证高速缓存，其中 *UID* 是用户的十进制 UID
`/tmp/ovsec_adm.`*xxxxxx*	口令更改操作生命周期内的临时凭证高速缓存，其中 *xxxxxx* 是一个随机字符串
`/var/krb5/.k5.`*REALM*	KDC 存储文件，其中包含 KDC 主密钥的副本
`/var/krb5/kadmin.log`	`kadmind` 的日志文件
`/var/krb5/kdc.log`	KDC 的日志文件
`/var/krb5/principal`	Kerberos 主体数据库
`/var/krb5/principal.kadm5`	Kerberos 管理数据库，其中包含策略信息
`/var/krb5/principal.kadm5.lock`	Kerberos 管理数据库锁文件
`/var/krb5/principal.ok`	Kerberos 数据库成功初始化时创建的 Kerberos 主体数据库初始化文件
`/var/krb5/principal.ulog`	Kerberos 更新日志，其中包含增量传播更新
`/var/krb5/slave_datatrans`	`kprop_script` 脚本用于传播的 KDC 的备份文件
`/var/krb5/slave_datatrans_`*slave*	对指定的 slave（从服务器）执行完全更新时创建的临时转储文件
`/var/user/`*$USER*`/krb-warn.conf`	基于用户的票证失效警告和自动更新配置文件

Kerberos 命令

Kerberos 产品包含以下命令。这些命令按其手册页列出。

`ftp(1)`	文件传输协议应用程序

kdestroy(1)	销毁 Kerberos 票证
kinit(1)	获取并缓存 Kerberos 票证授予票证
klist(1)	显示当前的 Kerberos 票证
kpasswd(1)	更改 Kerberos 口令
ktutil(1)	管理 Kerberos 密钥表文件
kvno(1)	列出 Kerberos 主体的密钥版本号
rcp(1)	远程文件复制应用程序
rlogin(1)	远程登录应用程序
rsh(1)	远程 shell 应用程序
scp(1)	安全远程文件复制应用程序
sftp(1)	安全文件传输应用程序
ssh(1)	用于远程登录的安全 shell
telnet(1)	基于 Kerberos 的 `telnet` 应用程序
kprop(1M)	Kerberos 数据库传播程序
gkadmin(1M)	Kerberos 数据库管理 GUI 程序，用于管理主体和策略
gsscred(1M)	管理 `gsscred` 表项
kadmin(1M)	远程 Kerberos 数据库管理程序（需要进行 Kerberos 验证），用于管理主体、策略和密钥表文件
kadmin.local(1M)	在主 KDC 上运行的本地 Kerberos 数据库管理程序，用于管理主体、策略和密钥表文件
kclient(1M)	Kerberos 客户机安装脚本，有无安装配置文件皆可使用
kdb5_ldap_util(1M)	为 Kerberos 数据库创建 LDAP 容器
kdb5_util(1M)	创建 Kerberos 数据库和存储文件
kdcmgr(1M)	配置 Kerberos 主 KDC 和从 KDC
kproplog(1M)	列出更新日志中更新项的摘要

Kerberos 守护进程

Kerberos 产品使用以下守护进程。

`/usr/lib/inet/proftpd`	安全文件传输协议守护进程
`/usr/lib/ssh/sshd`	安全 shell 守护进程
`/usr/lib/krb5/kadmind`	Kerberos 数据库管理守护进程
`/usr/lib/krb5/kpropd`	Kerberos 数据库传播守护进程
`/usr/lib/krb5/krb5kdc`	Kerberos 票证处理守护进程
`/usr/lib/krb5/ktkt_warnd`	Kerberos 票证失效警告和自动更新守护进程
`/usr/sbin/in.rlogind`	远程登录守护进程
`/usr/sbin/in.rshd`	远程 Shell 守护进程
`/usr/sbin/in.telnetd`	`telnet` 守护进程

Kerberos 术语

整个 Kerberos 文档中使用了以下 Kerberos 术语。要掌握 Kerberos 概念，必须先了解这些术语。

特定于 Kerberos 的术语

要管理 KDC，您需要了解本节中介绍的术语。

密钥分发中心 *(Key Distribution Center, KDC)* 是负责颁发凭证的 Kerberos 组件。这些凭证是使用 KDC 数据库中存储的信息创建的。每个领域至少需要两个 KDC：一个主

KDC，以及至少一个从 KDC。所有 KDC 都可生成凭证，但仅有主 KDC 才能处理对 KDC 数据库所做的任何更改。

存储文件包含 KDC 的主密钥。当重新引导服务器以在启动 `kadmind` 和 `krb5kdc` 命令之前自动验证 KDC 时，将使用此密钥。因为该文件包含主密钥，因此，该文件及其所有备份都应安全保存。该文件以 `root` 的只读权限创建。为确保文件安全，请勿更改其权限。如果该文件遭破坏，可以使用密钥访问或修改 KDC 数据库。

特定于验证的术语

要了解验证过程，您需要知道本节中介绍的术语。程序员和系统管理员应该熟悉这些术语。

客户机是在用户工作站上运行的软件。在客户机上运行的 Kerberos 软件将在此过程中发出许多请求。因此，区分该软件的操作和用户的操作非常重要。

术语服务器和服务通常可以互换使用。具体而言，术语服务器用于定义运行 Kerberos 软件的物理系统。术语服务对应于服务器上支持的特定功能（如 `ftp` 或 `nfs`）。文档通常会将服务器描述为服务的一部分，但此定义混淆了这些术语的含义。因此，术语 *server* 是指物理系统。术语 *service* 是指软件。

Kerberos 产品使用两种类型的密钥。其中一种类型是口令派生的密钥，这种密钥分配给每个用户主体，且仅为用户和 KDC 所知。另一种类型是与口令无关的随机密钥，因此不适合用户主体使用。随机密钥通常用于在密钥表中具有相应项并由 KDC 生成会话密钥的服务主体。服务主体可以使用随机密钥，因为服务可以访问密钥表中允许其以非交互方式运行的密钥。会话密钥由 KDC 生成，并在客户机和服务之间共享，可用于在两者之间提供安全事务。

票证是一种信息包，用于将用户身份安全地传递到服务器或服务。一个票证仅对一台客户机以及某台特定服务器上的一项特殊服务有效。票证包含：

- 服务的主体名称
- 用户的主体名称
- 用户主机的 IP 地址
- 时间戳
- 定义票证生命周期的值
- 会话密钥的副本

所有此类数据都使用服务器的服务密钥进行加密。KDC 颁发的票证嵌入在凭证中。颁发后的票证可重复使用，直到过期。

凭证是一个信息包，其中包含票证和匹配的会话密钥。凭证使用请求主体的密钥进行加密。通常，KDC 会生成凭证以响应客户机的票证请求。

验证者是服务器用来验证客户机用户主体的信息。验证者包含用户的主体名称、时间戳和其他数据。与票证不同，验证者只能使用一次，通常在请求访问服务时使用。验证者

使用客户机和服务器共享的会话密钥进行加密。通常，客户机会创建验证者，并将其与服务器或服务的票证一同发送，以便向服务器或服务进行验证。

票证类型

票证具有可管理其使用方式的属性。这些属性在创建票证时指定，但之后亦可修改。例如，可将票证从 `forwardable` 改为 `forwarded`。可以使用 `klist` 命令查看票证属性。请参见“查看 Kerberos 票证” [148]。

以下一个或多个术语对票证进行了描述：

Forwardable（可转发）/ forwarded（已转发）	可转发票证可以从一台主机发送到另一台主机，从而使客户机无需对自身进行重新验证。例如，如果用户 `david` 登录到用户 `jennifer` 的计算机时获取了一张可转发票证，则 `david` 不必获取新的票证（从而对自身进行重新验证）即可登录到自己的计算机。有关可转发票证的示例，请参见例 6-1 “创建 Kerberos 票证”。
Initial（初始）	初始票证是一种直接颁发的票证，而并非基于票证授予票证的票证。某些服务（如用于更改口令的应用程序）可能会要求将票证标记为 "initial"（初始），以便确保客户机可证明其知道密钥。初始票证表明客户机最近已进行了自我验证，而并非依赖于较旧的票证授予票证。
Invalid（无效）	无效票证是一个尚不可用的以后生效票证。应用服务器会拒绝无效票证，直到其通过验证为止。要进行验证，必须在票证开始时间已过之后由客户机将设置了 `VALIDATE` 标志的票证放置在票证授予服务请求的 KDC 中。
Postdatable（可以后生效）/ postdated（以后生效）	以后生效的票证是一种在其创建之后的某个指定时间之前不会生效的票证。例如，此类票证对于计划在深夜运行的批处理作业非常有用，因为如果该票证被盗，则在运行批处理作业之前无法使用该票证。颁发以后生效的票证时，该票证尚未生效，且在其开始时间已过且客户机请求 KDC 进行验证之前一直保持此状态。通常，以后生效的票证在票证授予票证的截止时间之前会一直有效。但是，如果将以后生效的票证标记为可更新，则通常会将其生命周期设置为等于票证授予票证的整个生命周期的持续时间。
Proxiable（可代理）/proxy（代理）	有时，主体可能需要允许服务代表其执行操作。创建该票证时，必须指定代理的主体名称。Oracle Solaris 不支持可代理票证或代理票证。 代理票证与可转发票证类似，但前者仅对单个服务有效。而可转发票证会向服务授予对客户机身份的完全使用权限。因此，可以将可转发票证视为一种超级代理。

Renewable（可更新）	由于具有较长生命周期的票证存在安全风险，因此可将票证指定为可更新。可更新票证具有两个截止时间：票证当前实例的截止时间，以及任意票证的最长生命周期（一周）。如果客户机要继续使用票证，则可在第一个截止时间之前更新票证。例如，假设某个票证的有效期为一小时，而所有票证的最长生命周期为 10 小时。如果持有该票证的客户机要将该票证再保留几个小时，则此客户机必须在有效的小时数内更新票证。如果票证到达最长票证生命周期（10 个小时），则该票证将自动过期且无法更新。

有关如何查看票证属性的信息，请参见“查看 Kerberos 票证” [148]。

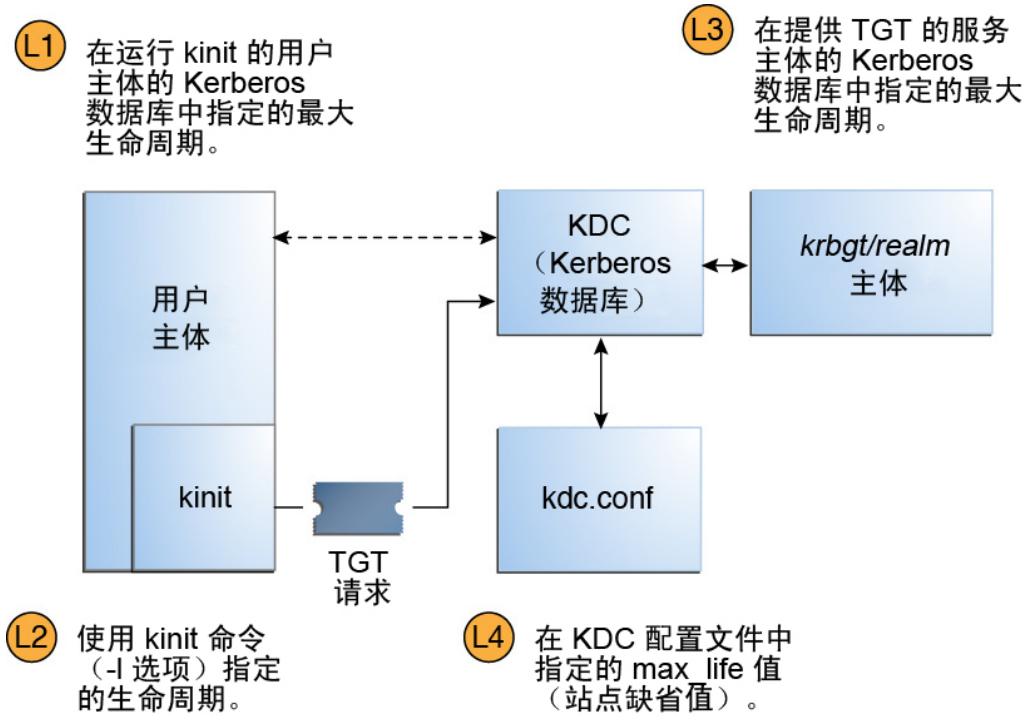
票证生命周期

当主体获取票证（包括票证授予票证 (Ticket Granting Ticket, TGT)）时，都会将票证的生命周期设置为以下生命周期值中的最小值：

- 通过 `kinit` 的 `-l` 选项指定的生命周期值，前提是使用 `kinit` 获取票证。缺省情况下，`kinit` 使用最长生命周期值。
- `kdc.conf` 文件中指定的最长生命周期值 (`max_life`)。
- Kerberos 数据库中为提供票证的服务主体指定的最长生命周期值。如果使用 `kinit`，则服务主体为 `krbtgt/`*realm*。
- Kerberos 数据库中为请求票证的用户主体指定的最长生命周期值。

下图显示了 TGT 生命周期的确定方法以及生命周期值的四个来源。当任何主体获取票证时，将以相似方式确定该票证的生命周期。有两处区别：`kinit` 不提供生命周期值，而提供票证的服务主体（而不是 `krbtgt/`*realm* 主体）会提供最长生命周期值。

图 7-1 如何确定 TGT 的生命周期



可更新票证的生命周期根据四个可更新生命周期值的最小值确定，如下所示：

- 当使用 `kinit` 获取或更新票证时，通过 `kinit` 的 `-r` 选项指定的可更新生命周期值。
- `kdc.conf` 文件中指定的最长可更新生命周期值 (`max_renewable_life`)。
- Kerberos 数据库中为提供票证的服务主体指定的最长可更新生命周期值。如果使用 `kinit`，则服务主体为 `krbtgt/`*realm*。
- Kerberos 数据库中为请求票证的用户主体指定的最长可更新生命周期值。

Kerberos 主体名称

每张票证都以主体名称标识。主体名称可以标识用户或服务。以下示例显示了典型的主体名称：

`changepw/kdc1.example.com@EXAMPLE.COM`	更改口令时，允许访问 KDC 的主 KDC 服务器的主体。
`clntconfig/admin@EXAMPLE.COM`	`kclient` 安装实用程序使用的主体。
`ftp/boston.example.com@EXAMPLE.COM`	`ftp` 服务使用的主体。此主体可用于替代 `host` 主体。
`host/boston.example.com@EXAMPLE.COM`	基于 Kerberos 的应用程序（如 `klist` 和 `kprop`）和服务（如 `ftp` 和 `telnet`）使用的主体。此主体称为 `host` 或服务主体，用于验证 NFS 挂载。客户机也会使用该主体验证 TGT 是否是从正确的 KDC 颁发给客户机的。
`K/M@EXAMPLE.COM`	主密钥名称主体。一个主密钥名称主体可与每个主 KDC 关联。
`kadmin/history@EXAMPLE.COM`	一种主体，其中包含用于保存其他主体的口令历史记录的密钥。每个主 KDC 都具有一个这样的主体。
`kadmin/kdc1.example.com@EXAMPLE.COM`	允许使用 `kadmind` 访问 KDC 的主 KDC 服务器的主体。
`kadmin/changepw.example.com@EXAMPLE.COM`	一种主体，用于接收来自未运行 Solaris 发行版的客户机的口令更改请求。
`krbtgt/EXAMPLE.COM@EXAMPLE.COM`	生成票证授予票证时使用的主体。
`krbtgt/EAST.EXAMPLE.COM@WEST.EXAMPLE.COM`	此主体是跨领域的票证授予票证的示例。
`nfs/boston.example.com@EXAMPLE.COM`	NFS 服务使用的主体。此主体可用于替代 `host` 主体。
`root/boston.example.com@EXAMPLE.COM`	与客户机上的 `root` 帐户关联的主体。该主体称为 `root` 主体，向已挂载 NFS 的文件系统提供 `root` 访问权限。
username`@EXAMPLE.COM`	用户的主体。
username`/admin@EXAMPLE.COM`	`admin` 主体，可用于管理 KDC 数据库。

♦ ♦ ♦ 第 8 章

Kerberos 错误消息和故障排除

本章将分析使用 Kerberos 服务时可能收到的错误消息，另外还针对各种问题提供一些故障排除提示。

本章包含以下主题：

Kerberos 错误消息

本节介绍有关 Kerberos 错误消息的信息，包括每个错误出现的原因以及解决此错误的方法。

gkadmin GUI 错误消息

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

原因: 登录时使用的 `admin` 主体在 Kerberos ACL 文件 (`kadm5.acl`) 中没有列出特权 (`l`)。因此，无法查看主体列表或策略列表。

解决方法: 必须在 "Name"（名称）字段中键入主体名称和策略名称才能对其进行处理，或者需要使用具有相应特权的主体登录。

```
JNI: Java * failed
```

原因: `gkadmin` GUI 使用的 Java 本地接口存在严重问题。

解决方法: 退出 `gkadmin` 然后重新启动。如果问题仍然存在，请报告错误。

常见的 Kerberos 错误消息 (A-M)

本节按字母顺序 (A-M) 列出了 Kerberos 命令、Kerberos 守护进程、PAM 框架、GSS 接口、NFS 服务和 Kerberos 库的常见错误消息。

`Bad lifetime value`

原因: 提供的生命周期值无效或格式不正确。

解决方法: 确保提供的值与 kinit(1) 手册页中的“时间格式”一节相符。

`Bad start time value`

原因: 提供的开始时间值无效或格式不正确。

解决方法: 确保提供的值与 kinit(1) 手册页中的“时间格式”一节相符。

`Cannot contact any KDC for requested realm`

原因: 请求的领域中没有 KDC 响应。

解决方法: 确保至少可以访问一个 KDC（主从皆可），或 `krb5kdc` 守护进程正在 KDC 上运行。有关已配置的 KDC (`kdc =` *kdc-name*) 的列表，请查看 `/etc/krb5/krb5.conf` 文件。

`Cannot determine realm for host: host is '`*hostname*`'`

原因: Kerberos 无法确定主机的领域名称。

解决方法: 确保存在缺省领域名称，或在 Kerberos 配置文件 (`krb5.conf`) 中设置了域名映射。

`Cannot find a kadmin KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

`Cannot find a kpassword KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

`Cannot find a master KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

`Cannot find any KDC entries in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

原因: `krb5.conf` 文件或 DNS 服务器记录配置不正确。

解决方法: 确保 Kerberos 配置文件 (`/etc/krb5/krb5.conf`) 或 KDC 的 DNS 服务器记录已配置正确。

`Cannot find address for '`*hostname*`': '`*error-string*`'`

原因: 在 DNS 记录中找不到给定主机名的地址。

解决方法: 修复 DNS 中的主机记录，或更正 DNS 查找进程中的错误。

`cannot initialize realm` *realm-name*

原因: KDC 可能没有存储文件。

解决方法: 确保 KDC 具有存储文件。否则，请使用 `kdb5_util` 命令创建一个存储文件，然后尝试重新启动 `krb5kdc` 命令。

`Cannot resolve KDC for requested realm`

原因: Kerberos 无法确定领域的任何 KDC。

解决方法: 确保 Kerberos 配置文件 (`krb5.conf`) 在 `realm` 部分指定了 KDC。

`Cannot resolve network address for KDCs '`*hostname*`' discovered via DNS Service Location records for realm '`*realm-name*`'`

`Cannot resolve network address for KDCs '`*hostname*`' specified in krb5.conf(4) for realm '`*realm-name*`'`

原因: `krb5.conf` 文件或 DNS 服务器记录配置不正确。

解决方法: 确保 Kerberos 配置文件 (`/etc/krb5/krb5.conf`) 或 KDC 的 DNS 服务器记录已正确配置。

`Can't open/find Kerberos configuration file`

原因: Kerberos 配置文件 (`krb5.conf`) 不可用。

解决方法: 确保 `krb5.conf` 文件位于正确的位置，并具有合适的权限。该文件应可由 `root` 写入，并可由其他任何用户读取。

`Client '`*principal*`' pre-authentication failed`

原因: 对主体的验证失败。

解决方法: 确保用户使用的是正确的口令。

`Client or server has a null key`

原因: 主体拥有空密钥。

解决方法: 使用 `kadmin` 的 `cpw` 命令修改主体，使其拥有非空密钥。

`Communication failure with server while initializing kadmin interface`

原因: 为主 KDC 指定的主机未运行 `kadmind` 守护进程。

解决方法: 确保为主 KDC 指定了正确的主机名。如果指定了正确的主机名，请确保 `kadmind` 正在指定的主 KDC 上运行。

`Credentials cache file permissions incorrect`

原因: 您对凭证高速缓存 (`/tmp/krb5cc_`*uid*) 没有相应的读写权限。

解决方法: 请确保您拥有对凭证高速缓存的读写权限。

`Credentials cache I/O operation failed` *XXX*

原因: Kerberos 在向系统的凭证高速缓存 (`/tmp/krb5cc_`*uid*) 进行写入时出现问题。

解决方法: 请使用 `df` 命令确认尚未删除凭证高速缓存，并且设备中还有剩余空间。

`Decrypt integrity check failed`

原因: 您的票证可能无效。

解决方法: 验证以下两种情况：

- 确保您的凭证有效。使用 `kdestroy` 销毁票证，然后使用 `kinit` 创建新的票证。
- 确保目标主机的密钥表文件的服务密钥版本正确。使用 `kadmin` 查看 Kerberos 数据库中服务主体（例如 `host/`*FQDN-hostname*）的密钥版本号。另外，在目标主机上使用 `klist -k` 命令，以确保该主机具有相同的密钥版本号。

`Decrypt integrity check failed for client 'principal' and server 'hostname'`

原因: 您的票证可能无效。

解决方法: 确保您的凭证有效。使用 `kdestroy` 命令销毁票证，然后使用 `kinit` 命令创建新票证。

`failed to obtain credentials cache`

原因: 在 kadmin 初始化期间，kadmin 尝试获取 admin 主体的凭证时失败。

解决方法: 确保在执行 kadmin 命令时使用正确的主体和口令。

```
Field is too long for this implementation
```

原因: 基于 Kerberos 的应用程序所发送的消息太长。如果传输协议是 UDP，则可能会生成此错误。UDP 的缺省最大消息长度是 65535 字节。此外，对通过 Kerberos 服务发送的协议消息中的单个字段也有限制。

解决方法: 确认您没有在 KDC 服务器的 /etc/krb5/kdc.conf 文件中将传输协议限制为 UDP。

```
GSS-API (or Kerberos) error
```

原因: 此消息是通用的 GSS-API 或 Kerberos 错误消息，可能由几种不同的问题所导致。

解决方法: 检查 /var/krb5/kdc.log 文件，找出发生此错误时记录的更具体的错误消息。

```
Improper format of Kerberos configuration file
```

原因: Kerberos 配置文件包含无效项。

解决方法: 确保 krb5.conf 文件中的所有关系后面都跟有 "=" 符号和值。另外，请确认每个子段中的括号都是成对出现的。

```
Invalid credential was supplied
Service key not available
```

原因: 凭证高速缓存中的服务票证可能不正确。

解决方法: 尝试使用该服务之前，请销毁当前凭证高速缓存并重新运行 kinit 命令。

```
Invalid flag for file lock mode
```

原因: 出现 Kerberos 内部错误。

解决方法: 请报告错误。

```
Invalid message type specified for encoding
```

原因: Kerberos 无法识别基于 Kerberos 的应用程序发送的消息类型。

解决方法: 如果使用的基于 Kerberos 的应用程序是由您的站点或供应商开发的，请确保此应用程序正确使用 Kerberos。

`kadmin: Bad encryption type while changing host/`*FQDN*`'s key`

原因: 在较新的发行版中，基本版本中包括更多的缺省加密类型。客户机请求的加密类型可能不受运行该软件旧版本的 KDC 支持。

解决方法: 在客户机上的 `krb5.conf` 中设置 `permitted_enctypes`，使其不包括 `aes256` 加密类型。将需要对每台新的客户机执行此步骤。

`KDC can't fulfill requested option`

原因: KDC 不允许请求的选项。一种可能是正在请求以后生效或可转发的选项，而 KDC 不允许这些选项。另一种可能是请求了 TGT 更新，但没有可更新的 TGT。

解决方法: 确定是请求了 KDC 不允许的选项，还是票证类型不可用。

`KDC reply did not match expectation: KDC not found. Probably got an unexpected realm referral`

原因: KDC 回复中未包含所需的主体名称，或者响应中的其他值不正确。

解决方法: 确保您与之通信的 KDC 符合 RFC4120，发送的请求是 Kerberos V5 请求，并且该 KDC 可用。

`kdestroy: Could not obtain principal name from cache`

原因: 凭证高速缓存缺失或已损坏。

解决方法: 检查提供的高速缓存位置是否正确。如有必要，使用 `kinit` 删除 TGT 并获取新的 TGT。

`kdestroy: No credentials cache file found while destroying cache`

原因: 凭证高速缓存 (`/tmp/krb5c_`*uid*) 缺失或已损坏。

解决方法: 检查提供的高速缓存位置是否正确。如有必要，使用 `kinit` 删除 TGT 并获取新的 TGT。

`kdestroy: TGT expire warning NOT deleted`

原因: 凭证高速缓存缺失或已损坏。

解决方法: 检查提供的高速缓存位置是否正确。如有必要，使用 `kinit` 删除 TGT 并获取新的 TGT。

`Kerberos authentication failed`

原因: Kerberos 口令错误，或该口令可能与 UNIX 口令不同步。

解决方法: 如果口令不同步，则必须指定 Kerberos 口令以完成验证。用户可能会忘记其原始口令。

`Key version` *number* `is not available for principal` *principal*

原因: 密钥的密钥版本与应用服务器中的密钥版本不匹配。

解决方法: 使用 `klist -k` 命令检查应用服务器上的密钥版本。

`Key version number for principal in key table is incorrect`

原因: 密钥表文件中主体的密钥版本与 Kerberos 数据库中的版本不同。可能已更改了服务密钥，也可能正在使用旧服务票证。

解决方法: 如果更改了服务密钥（例如，通过使用 `kadmin`），则需要提取新密钥，并将其存储在正在运行该服务的主机的密钥表文件中。

或者，您也可能正在使用具有较旧密钥的旧服务票证。在这种情况下，可能需要运行 `kdestroy` 命令，然后再次运行 `kinit` 命令。

`kinit: gethostname failed`

原因: 本地网络配置中的错误导致 `kinit` 失败。

解决方法: 确保主机配置正确。

`login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1`

原因: Kerberos PAM 模块缺失，或不是有效的可执行二进制文件。

解决方法: 确保 Kerberos PAM 模块在 `/usr/lib/security` 目录下，并且是有效的可执行二进制文件。另请确保 `login` 的 PAM 配置文件包含指向 `pam_krb5.so.1` 的正确路径。

`Looping detected getting initial creds: '`*client-principal*`' requesting ticket '`*service-principal*`'. Max loops is` *value*`. Make sure a KDC is available.`

原因: Kerberos 多次尝试获取初始票证，但均失败。

解决方法: 确保至少有一个 KDC 正在响应验证请求。

`Master key does not match database`

原因: 装入的数据库转储不是基于包含主密钥的数据库创建的。主密钥位于 `/var/krb5/.k5.`*REALM* 中。

解决方法: 确保装入的数据库转储中的主密钥与 `/var/krb5/.k5.`*REALM* 中的主密钥匹配。

`Matching credential not found`

原因: 未找到与请求匹配的凭证。凭证高速缓存中没有您的请求需要的凭证。

解决方法: 使用 `kdestroy` 销毁票证，然后使用 `kinit` 创建新的票证。

`Message out of order`

原因: 使用顺序保密机制发送的消息送达时顺序错误。某些消息可能已在传输过程中丢失。

解决方法: 必须重新初始化 Kerberos 会话。

`Message stream modified`

原因: 计算出的校验和与消息校验和不匹配。消息在传输过程中可能已被修改，这表明存在安全漏洞。

解决方法: 确保消息在网络中正确发送。由于此消息还可能表明消息在发送过程中被篡改，因此请销毁票证，然后重新初始化正在使用的 Kerberos 服务。

常见的 Kerberos 错误消息 (N-Z)

本节按字母顺序 (N-Z) 列出了 Kerberos 命令、Kerberos 守护进程、PAM 框架、GSS 接口、NFS 服务和 Kerberos 库的常见错误消息。

`No credentials cache file found`

原因: Kerberos 找不到凭证高速缓存 (`/tmp/krb5cc_`*uid*)。

解决方法: 确保凭证文件存在且可读。否则，请再次尝试运行 `kinit` 命令。

`No credentials were supplied, or the credentials were unavailable or inaccessible`
`No credential cache found`

原因: 用户的凭证高速缓存不正确或不存在。

解决方法: 用户必须在尝试启动服务之前运行 `kinit`。

`No credentials were supplied, or the credentials were unavailable or inaccessible`

`No principal in keytab ('`*filename*`') matches desired name` *principal*

原因: 尝试验证服务器时发生错误。

解决方法: 确保 host 主体或服务主体位于服务器的密钥表文件中。

`Operation requires "`*privilege*`" privilege`

原因: 未在 `kadm5.acl` 文件中为所使用的 `admin` 主体分配相应特权。

解决方法: 使用具有相应特权的主体。或者，也可以为所使用的主体配置相应特权。通常，名称中包含 `/admin` 的主体都具有相应的特权。

`PAM-KRB5 (auth): krb5_verify_init_creds failed: Key table entry not found`

原因: 远程应用程序尝试在本地 `/etc/krb5/krb5.keytab` 文件中读取主机的服务主体，但不存在任何主体。

解决方法: 将主机的服务主体添加到主机的密钥表文件中。

`Permission denied in replay cache code`

原因: 无法打开系统的重放高速缓存。首次运行服务器时所使用的用户 ID 可能与当前的用户 ID 不同。

解决方法: 确保重放高速缓存具有相应的权限。重放高速缓存存储在运行基于 Kerberos 的服务器应用程序的主机上。对于非 `root` 用户，重放高速缓存文件名为 `/var/krb5/rcache/rc_`*service_name_uid*。对于 root 用户，重放高速缓存文件名为 `/var/krb5/rcache/root/rc_`*service_name*。

`Protocol version mismatch`

原因: 很可能向 KDC 发送了 Kerberos V4 请求。Kerberos 服务仅支持 Kerberos V5 协议。

解决方法: 确保应用程序使用的是 Kerberos V5 协议。

`Request is a replay`

原因: 请求已经发送到该服务器，并进行了处理。票证可能已被盗用，其他用户正在尝试重用这些票证。

解决方法: 等待几分钟，然后重新发送请求。

`Requested principal and ticket don't match: Requested principal is '`*service-principal*`' and TGT principal is '`*TGT-principal*`'`

原因: 您正连接的服务主体和您拥有的服务票证不匹配。

解决方法: 确保 DNS 正常运行。如果使用的是其他供应商的软件，请确保该软件使用的主体名称正确。

`Server refused to negotiate authentication, which is required for encryption. Good bye.`

原因: 远程应用程序无法接受或已配置为不接受来自客户机的 Kerberos 验证。

解决方法: 提供可以协商验证的远程应用程序，或配置该应用程序以使用相应标志来启用验证。

`Server rejected authentication (during sendauth exchange)`

原因: 您正在尝试与其通信的服务器拒绝了验证。此错误通常出现在 Kerberos 数据库传播过程中。一些常见的原因可能是 `kpropd.acl` 文件、DNS 或密钥表文件存在问题。

解决方法: 如果在运行 `kprop` 以外的应用程序时收到此错误，请检查服务器的密钥表文件是否正确。

`Target name principal '`*principal*`' does not match` *service-principal*

原因: 所使用的服务主体与应用服务器所使用的服务主体不匹配。

解决方法: 在应用服务器中，确保服务主体包含在密钥表文件中。对于客户机，确保使用的是正确的服务主体。

`The ticket isn't for us`

`Ticket/authenticator do not match.`

原因: 请求中的主体名称可能与服务主体的名称不匹配。原因可能是所发送票证使用的是主体的 FQDN 名称，而服务需要非 FQDN 名称，或者所发送票证使用的是主体的非 FQDN 名称，而服务需要 FQDN 名称。

解决方法: 如果在运行 `kprop` 以外的应用程序时收到此错误，请检查服务器的密钥表文件是否正确。

`Truncated input file detected`

原因: 操作中使用的数据库转储文件不是完整的转储文件。

解决方法: 重新创建转储文件，或使用其他数据库转储文件。

Kerberos 故障排除

本节介绍了有关 Kerberos 软件的故障排除信息。

与密钥版本号相关的问题

有时，KDC 使用的密钥版本号 (Key Version Number, KVNO) 与 `/etc/krb5/krb5.keytab` 中存储的系统托管服务的服务主体密钥不匹配。如果 KDC 中创建了一组新密钥，但没有在密钥表文件中更新这些新密钥，KVNO 可能会不同步。诊断问题后，刷新 `krb5.keytab` 文件。

1. 列出 keytab 条目。

   每个主体的 KVNO 是每个条目中的第一项。

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

2. 使用 `host` 密钥获取初始凭证。

   ```
   # kinit -k
   ```

3. 确定 KDC 所使用的 KVNO。

   ```
   # kvno nfs/denver.example.com
   nfs/denver.example.com@EXAMPLE.COM: kvno = 3
   ```

   请注意，此处列出的 KVNO 是 3 而不是 2。

与 `krb5.conf` 的文件格式相关的问题

如果 `krb5.conf` 文件的格式不正确，则以下错误消息可能会显示在终端窗口中或记录在日志文件中：

```
Improper format of Kerberos configuration file while initializing krb5 library
```

如果格式不正确，则关联的服务将容易受到攻击。您必须首先解决该问题，然后再允许使用 Kerberos 功能。

传播 Kerberos 数据库时出现问题

如果 Kerberos 数据库传播失败，请在从 KDC 与主 KDC 之间尝试使用 `/usr/bin/rlogin -x`（此为双向操作）。

如果 KDC 在缺省情况下处于安全模式，则会禁用 `rlogin` 命令，因此不能使用它来解决该问题。要在 KDC 上启用 `rlogin`，必须启用 `eklogin` 服务。

```
# svcadm enable svc:/network/login:eklogin
```

解决此问题后，禁用 `eklogin` 服务。

```
# svcadm disable svc:/network/login:eklogin
```

如果远程服务无法正常工作，可能是因为 KDC 上的密钥表文件存在问题。如果远程服务能够正常工作，则问题不在于密钥表文件或名称服务，因为 `rlogin` 与传播软件使用同一个 `host/`*host-name* 主体。在这种情况下，请确保 `kpropd.acl` 文件正确。

挂载基于 Kerberos 的 NFS 文件系统时出现问题

- 如果挂载基于 Kerberos 的 NFS 文件系统失败，请确保 NFS 服务器上存在 `/var/rcache/root` 文件。如果文件系统不是由 `root` 所有，请删除该文件系统并再次尝试挂载。
- 如果访问基于 Kerberos 的 NFS 文件系统时出现问题，请确保您的系统和 NFS 服务器上启用了 `gssd` 服务。
- 如果尝试访问基于 Kerberos 的 NFS 文件系统时出现 `invalid argument` 或 `bad directory` 错误消息，可能是因为尝试挂载 NFS 文件系统时使用的不是全限定 DNS 名称。正在挂载的主机与服务器的密钥表文件中的服务主体主机名部分不相同。

  如果服务器有多个以太网接口，并且已将 DNS 设置为使用“每个接口一个名称”的方案，而不是“每台主机多条地址记录”的方案，则也可能会出现此问题。对于 Kerberos 服务，应为每台主机设置多条地址记录，如下所示[1]：

```
my.host.name.   A       1.2.3.4
A       1.2.4.4
A       1.2.5.4

my-en0.host.name.       A       1.2.3.4
my-en1.host.name.       A       1.2.4.4
my-en2.host.name.       A       1.2.5.4

4.3.2.1         PTR     my.host.name.
4.4.2.1         PTR     my.host.name.
4.5.2.1         PTR     my.host.name.
```

[1]Ken Hornstein，"Kerberos FAQ"，[http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html#kerbdns]，访问时间 2010 年 3 月 10 日。

在本示例中，此设置允许引用服务器的密钥表文件中的不同接口和一个服务主体（而非三个服务主体）一次。

以 root 用户身份进行验证时出现问题

如果尝试成为系统上的 root 用户时验证失败，并且已将 root 主体添加到主机的密钥表文件中，则应检查两个方面。首先，请确保密钥表文件中的 root 主体具有一个全限定主机名作为其实例。如果具有该名称，请检查 /etc/resolv.conf 文件，以确保系统已正确设置为 DNS 客户机。

观察从 GSS 凭证到 UNIX 凭证的映射

为了可以监视凭证映射，请首先在 /etc/gss/gsscred.conf 文件中取消对以下行的注释。

```
SYSLOG_UID_MAPPING=yes
```

下一步，让 gssd 服务读取 /etc/gss/gsscred.conf 文件。

```
# pkill -HUP gssd
```

现在，您可以在 gssd 请求凭证映射时对其进行监视了。如果 auth 系统工具将 syslog.conf 文件配置为 debug 严重级别，则可通过 syslog 守护进程记录这些映射。

注 - 如果已启用 rsyslog 服务实例，则可通过 rsyslog 守护进程记录映射。

对 Kerberos 服务使用 DTrace

Kerberos 机制支持多种 DTrace 探测器来解码各种协议消息。有关列表，请参见附录 A, 用于 Kerberos 的 DTrace 探测器。相对于其他协议检查器，DTrace 探测器具有明显优势，因为 DTrace 可让特权用户轻松查看未加密的 Kerberos 数据和应用程序数据。

以下示例指明了可以使用 Kerberos DTrace 探测器查看的信息。

例 8-1 使用 DTrace 跟踪 Kerberos 消息

以下脚本使用 DTrace 探测器来显示有关系统收发的 Kerberos 消息的详细信息。请注意，所显示的字段基于分配给 krb_message-recv 和 krb_message-send 探测器的结构。有关更多信息，请参见“Kerberos DTrace 探测器的定义” [191]。

```
kerberos$target:::krb_message-recv
{
    printf("<- krb message recved: %s\n", args[0]->krb_message_type);
```

```
    printf("<- krb message remote addr: %s\n", args[1]->kconn_remote);
    printf("<- krb message ports: local %d remote %d\n",
            args[1]->kconn_localport, args[1]->kconn_remoteport);
    printf("<- krb message protocol: %s transport: %s\n",
            args[1]->kconn_protocol, args[1]->kconn_type);
}

kerberos$target:::krb_message-send
{
    printf("-> krb message sent: %s\n", args[0]->krb_message_type);
    printf("-> krb message remote addr: %s\n", args[1]->kconn_remote);
    printf("-> krb message ports: local %d remote %d\n",
            args[1]->kconn_localport, args[1]->kconn_remoteport);
    printf("-> krb message protocol: %s transport: %s\n",
            args[1]->kconn_protocol, args[1]->kconn_type);
    printf("\n");
}

kerberos$target:::krb_error-read
{
    printf("<- krb error code: %s\n", args[1]->kerror_error_code);
    printf("<- krb error client: %s server: %s\n", args[1]->kerror_client,
            args[1]->kerror_server);
    printf("<- krb error e-text: %s\n", args[1]->kerror_e_text);
    printf("\n");
}
```

可以从命令行中调用前一个脚本，也可以通过 `krb5kdc` 守护进程调用。以下示例显示了如何从命令行中调用名为 `krb-dtrace.d` 的脚本。该命令将促使 Kerberos 收发消息。请注意，必须指定 `LD_NOLAZYLOAD=1` 来强制装入包含 Kerberos DTrace 探测器的 Kerberos `mech_krb5.so` 库。

```
# LD_NOLAZYLOAD=1 dtrace -s ./krb\-dtrace.d -c kinit
dtrace: script './krb-dtrace' matched 4 probes
kinit: Client 'root@DEV.ORACLE.COM' not found in Kerberos database while g
etting initial credentials
dtrace: pid 3750 has exited
 CPU     ID                    FUNCTION:NAME
   2  74782 k5_trace_message_send:krb_message-send -> krb message sent: KR
B_AS_REQ(10)
-> krb message remote addr: 10.229.168.163
-> krb message ports: local 62029 remote 88
-> krb message protocol: ipv4 transport: udp


   2  74781 k5_trace_message_recv:krb_message-recv <- krb message recved:
KRB_ERROR(30)
<- krb message remote addr: 10.229.168.163
<- krb message ports: local 62029 remote 88
<- krb message protocol: ipv4 transport: udp

   2  74776     krb5_rd_error:krb_error-read <- krb error code: KDC_ERR_C_
PRINCIPAL_UNKNOWN(6)
<- krb error client: root@DEV.ORACLE.COM server: krbtgt/DEV.ORACLE.COM@DEV
```

```
.ORACLE.COM
<- krb error e-text: CLIENT_NOT_FOUND
```

要通过 krb5kdc 守护进程使用该脚本，必须启用 svc:/network/security/krb5kdc:default 服务并使其联机。请注意，以下命令不使用 LD_NOLAZYLOAD=1，因为 mech_krb5.so 库将装入 krb5kdc 守护进程。

```
# dtrace -s ./krb\-dtrace.d -p $(pgrep -x krb5kdc)
```

例 8-2 使用 DTrace 查看 Kerberos 预验证类型

以下示例显示客户机选择的预验证类型。第一步是创建 DTrace 脚本，如下所示：

```
cat krbtrace.d
kerberos$target:::krb_message-recv
{
printf("<- krb message recved: %s\n", args[0]->krb_message_type);
printf("<- krb message remote addr: %s\n", args[1]->kconn_remote);
printf("<- krb message ports: local %d remote %d\n",
args[1]->kconn_localport, args[1]->kconn_remoteport);
printf("<- krb message protocol: %s transport: %s\n",
args[1]->kconn_protocol, args[1]->kconn_type);
}

kerberos$target:::krb_message-send
{
printf("-> krb message sent: %s\n", args[0]->krb_message_type);
printf("-> krb message remote addr: %s\n", args[1]->kconn_remote);
printf("-> krb message ports: local %d remote %d\n",
args[1]->kconn_localport, args[1]->kconn_remoteport);
printf("-> krb message protocol: %s transport: %s\n",
args[1]->kconn_protocol, args[1]->kconn_type);
printf("\n");
}

kerberos$target:::krb_kdc_req-make
{
printf("-> krb kdc_req make msg type: %s\n", args[0]->krb_message_type);
printf("-> krb kdc_req make pre-auths: %s\n", args[1]->kdcreq_padata_types);
printf("-> krb kdc_req make auth data: %s\n", args[1]->kdcreq_authorization_data);
printf("-> krb kdc_req make client: %s server: %s\n", args[1]->kdcreq_client,
args[1]->kdcreq_server );
}

kerberos$target:::krb_kdc_req-read
{
/* printf("<- krb kdc_req msg type: %s\n", args[0]->krb_message_type); */
printf("<- krb kdc_req client: %s server: %s\n", args[1]->kdcreq_client,
args[1]->kdcreq_server );
printf("\n");
}

kerberos$target:::krb_kdc_rep-read
```

```
{
/* printf("<- krb kdc_rep msg type: %s\n", args[0]->krb_message_type); */
printf("<- krb kdc_rep client: %s server: %s\n", args[1]->kdcrep_client,
args[1]->kdcrep_enc_server );
printf("\n");
}

kerberos$target:::krb_ap_req-make
{
printf("-> krb ap_req make server: %s client: %s\n", args[2]->kticket_server,
args[2]->kticket_enc_client );
}

kerberos$target:::krb_error-read
{
printf("<- krb error code: %s\n", args[1]->kerror_error_code);
printf("<- krb error client: %s server: %s\n", args[1]->kerror_client,
args[1]->kerror_server);
printf("<- krb error e-text: %s\n", args[1]->kerror_e_text);
printf("\n");
}
```

接下来，以 Kerberos 系统上的某个特权用户的身份执行 `krbtrace.d` 脚本，方法为键入以下命令：

```
# LD_BIND_NOW=1 dtrace -qs krbtrace.d -c "kinit -k"
.
.
-> krb kdc_req make pre-auths: FX_COOKIE(133) ENC_TIMESTAMP(2) REQ_ENC_PA_REP(149)
```

在输出中显示了预验证类型。有关各种预验证类型的更多信息，请参见 RFC 4120。

例 8-3 使用 DTrace 转储 Kerberos 错误消息

```
# dtrace -n 'krb_error-make {
printf("\n{");
printf("\n\tctime = %Y", (uint64_t)(args[1]->kerror_ctime * 1000000000));
printf("\n\tcusec = %d", args[1]->kerror_cusec);
printf("\n\tstime = %Y", (uint64_t)(args[1]->kerror_stime * 1000000000));
printf("\n\tsusec = %d", args[1]->kerror_susec);
printf("\n\terror_code = %s", args[1]->kerror_error_code);
printf("\n\tclient = %s", args[1]->kerror_client);
printf("\n\tserver = %s", args[1]->kerror_server);
printf("\n\te_text = %s", args[1]->kerror_e_text);
printf("\n\te_data = %s", "");
printf("\n}");
}'
dtrace: description 'krb_error-make ' matched 1 probe
CPU     ID                    FUNCTION:NAME
0  78307     krb5_mk_error:krb_error-make
{
ctime = 2012 May 10 12:10:20
cusec = 0
```

```
stime = 2012 May 10 12:10:20
susec = 319090
error_code = KDC_ERR_C_PRINCIPAL_UNKNOWN(6)
client = testuser@EXAMPLE.COM
server = krbtgt/EXAMPLE.COM@EXAMPLE.COM
e_text = CLIENT_NOT_FOUND
e_data =
}
```

例 8-4 使用 DTrace 查看 SSH 服务器的服务票证

```
# LD_PRELOAD_32=/usr/lib/gss/mech_krb5.so.1 dtrace -q -n '
kerberos$target:::krb_kdc_req-make {
printf("kdcreq_server: %s",args[1]->kdcreq_server);
}' -c "ssh local@four.example.com" -o dtrace.out
Last login: Wed Sep 10 10:10:20 2014
Oracle Solaris 11 X86    July 2014
$ ^D
# cat dtrace.out
kdcreq_server: host/four.example.com@EXAMPLE.COM
```

例 8-5 使用 DTrace 查看请求初始 TGT 时不可用的 KDC 的地址和端口

```
# LD_BIND_NOW=1 dtrace -q -n '
kerberos$target:::krb_message-send {
printf("%s:%d\n",args[1]->kconn_remote, args[1]->kconn_remoteport)
}' -c "kinit local4"

10.10.10.14:88
10.10.10.14:750
10.10.10.14:88
10.10.10.14:750
10.10.10.14:88
10.10.10.14:750
kinit(v5): Cannot contact any KDC for realm 'EXAMPLE.COM'
while getting initial credentials
```

例 8-6 使用 DTrace 查看来自 Kerberos 主体的请求

```
# LD_BIND_NOW=1 dtrace -qs /opt/kdebug/mykdtrace.d \
-c 'kadmin -p kdc/admin -w test123 -q listprincs'
Authenticating as principal kdc/admin with password.
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop1.example.com@TEST.NET
krb message sent: KRB_AS_REQ(10)
krb message recved: KRB_ERROR(30)
Err code: KDC_ERR_PREAUTH_REQUIRED(25)
Err msg client: kdc/admin@TEST.NET server: kadmin/interop1.example.com@TEST.NET
Err e-text: NEEDED_PREAUTH
krb kdc_req msg type: KRB_AS_REQ(10)
```

```
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop1.example.com@TEST.NET
krb message sent: KRB_AS_REQ(10)
krb message recved: KRB_AS_REP(11)
kadmin: Database error! Required KADM5 principal missing while
initializing kadmin interface
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop2.example.com@TEST.NET
krb message sent: KRB_AS_REQ(10)
krb message recved: KRB_ERROR(30)
Err code: KDC_ERR_S_PRINCIPAL_UNKNOWN(7)
Err msg client: kdc/admin@TEST.NET server: kadmin/interop2.example.com@TEST.NET
Err e-text: SERVER_NOT_FOUND
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop2.example.com@TEST.NET
```

以下脚本用于生成上述输出。

```
kerberos$target:::krb_message-recv
{
printf("krb message recved: %s\n", args[0]->krb_message_type);
}

kerberos$target:::krb_message-send
{
printf("krb message sent: %s\n", args[0]->krb_message_type);
}

kerberos$target:::krb_kdc_req-make
{
printf("krb kdc_req msg type: %s\n", args[0]->krb_message_type);
printf("krb kdc_req make client: %s server: %s\n", args[1]->kdcreq_client,
args[1]->kdcreq_server );
}

kerberos$target:::krb_ap_req-make
{
printf("krb ap_req make server: %s client: %s\n", args[2]->kticket_server,
args[2]->kticket_enc_client );
}

kerberos$target:::krb_error-read
{
printf("Err code: %s\n", args[1]->kerror_error_code);
printf("Err msg client: %s server: %s\n", args[1]->kerror_client,
args[1]->kerror_server);
printf("Err e-text: %s\n", args[1]->kerror_e_text);
}
```

♦ ♦ ♦ 第 9 章

9 使用简单验证和安全层

本章介绍有关简单验证和安全层 (Simple Authentication and Security Layer, SASL) 的信息。

关于 SASL

简单验证和安全层 (Simple Authentication and Security Layer, SASL) 是一种为网络协议提供验证和可选安全性服务的框架。应用程序将调用 SASL 库 `/usr/lib/libsasl.so`，此库提供该应用程序与各种 SASL 机制之间的胶合层。验证过程及提供可选安全性服务时会使用 SASL 机制。此 SASL 版本源于 Cyrus SASL，但进行了几处更改。

SASL 提供以下服务：

- 装入任何插件
- 确定应用程序帮助选择安全机制时必需的安全选项
- 列出应用程序可用的插件
- 为特定验证从可用机制列表中选择最佳机制
- 在应用程序与所选机制之间路由验证数据
- 将有关 SASL 协商的信息返回给应用程序

SASL 参考信息

本节提供有关 SASL 实现的信息。

SASL 插件

SASL 插件提供对安全机制、用户标准化和辅助属性检索的支持。缺省情况下，动态装入的 32 位插件安装在 `/usr/lib/sasl` 中，64 位插件安装在 `/usr/lib/sasl/`*`$ISA`* 中。提供了以下安全机制插件：

`crammd5.so.1` CRAM-MD5，仅支持验证，不支持授权。

`digestmd5.so.1` DIGEST-MD5，支持验证、完整性、保密性和授权。

`gssapi.so.1` GSSAPI，支持验证、完整性、保密性和授权。GSSAPI 安全机制需要使用有效的 Kerberos 基础结构。

`plain.so.1` PLAIN，支持验证和授权。

此外，EXTERNAL 安全机制插件和 INTERNAL 用户标准化插件内置在 `libsasl.so.1` 中。EXTERNAL 机制支持验证和授权。如果外部安全源提供该机制，则该机制支持完整性和保密性。如有必要，INTERNAL 插件会将领域名称添加到用户名。

目前，Oracle Solaris 发行版不提供任何 `auxprop` 插件。要使 CRAM-MD5 和 DIGEST-MD5 机制插件在服务器端发挥完全功能，用户必须提供 `auxprop` 插件以检索明文口令。PLAIN 插件还需要额外支持才能验证口令。对口令验证的支持包括：对服务器应用程序的回调、`auxprop` 插件、`saslauthd` 或 `pwcheck`。Oracle Solaris 发行版中未提供 `salauthd` 和 `pwcheck` 守护进程。要实现更好的互操作性，可使用 `mech_list` SASL 选项将服务器应用程序限制为可完全运行的机制。

SASL 环境变量

缺省情况下，客户机验证名称设置为 `getenv("LOGNAME")`。客户机或插件可以重置此变量。

SASL 选项

通过使用可在 `/etc/sasl/`*`app`*`.conf` 文件中设置的选项，可以在服务器端修改 `libsasl` 和插件的行为。变量 *app* 是服务器定义的应用程序名称。服务器 *app* 的文档应指定该应用程序名称。

支持以下选项：

`auto_transition` 用户成功进行纯文本验证后自动将该用户转换到其他机制。

auxprop_login	列出要使用的辅助属性插件的名称。
canon_user_plugin	选择要使用的 canon_user 插件。
mech_list	列出允许服务器应用程序使用的机制。
pwcheck_method	列出用于验证口令的机制。目前，auxprop 是唯一允许的值。
reauth_timeout	设置对验证信息进行高速缓存以便进行快速重新验证的时间（以分钟为单位）。此选项由 DIGEST-MD5 插件使用。将此选项设置为 0 将禁用重新验证。

不支持以下选项：

plugin_list	列出可用机制。不使用该选项是因为它会更改动态装入插件的行为。
saslauthd_path	定义用于与 saslauthd 守护进程通信的 saslauthd 门的位置。Oracle Solaris 发行版中没有包括 saslauthd 守护进程。因此，也不包括此选项。
keytab	定义 GSSAPI 插件使用的 keytab 文件的位置。可使用 KRB5_KTNAME 环境变量而不是此选项来设置缺省的 keytab 位置。

Cyrus SASL 中不包括以下选项。但是，Oracle Solaris 发行版中添加了这些选项：

use_authid	创建 GSS 客户机安全上下文时获取客户机凭证，而不使用缺省凭证。缺省情况下，将使用缺省客户机 Kerberos 身份。
log_level	为服务器设置所需的日志记录级别。

♦ ♦ ♦ 第 10 章

10 配置网络服务验证

本章提供了有关如何使用安全 RPC 在 NFS 挂载中对主机和用户进行验证的信息，它涵盖了以下主题：

关于安全 RPC

安全 RPC（Remote Procedure Call，远程过程调用）通过验证机制保护远程过程。Diffie-Hellman 验证机制可验证发出服务请求的主机和用户。此验证机制使用数据加密标准 (Data Encryption Standard, DES) 加密。使用安全 RPC 的应用程序包括 NFS 和 NIS 命名服务。

NFS 服务和安全 RPC

NFS 支持多个主机通过网络共享文件。在 NFS 服务中，一个服务器可为多个客户机保存数据和资源。这些客户机具有服务器与客户机共享文件系统的访问权限。登录到客户机系统的用户可以通过从服务器挂载文件系统来访问这些文件系统。对于客户机系统上的用户，就好像这些文件是客户机的本地文件。NFS 的最常见用途之一是允许将系统安装在办公室中，同时将所有用户文件存储在中心位置。NFS 服务的某些功能（例如 `mount` 命令的 `-nosuid` 选项）可用于禁止未经授权的用户打开设备和文件系统。

NFS 服务使用安全 RPC 来验证通过网络发出请求的用户。此过程称为安全 *NFS*。Diffie-Hellman 验证机制 `AUTH_DH` 使用 DES 加密以确保授权访问。`AUTH_DH` 机制也称为 `AUTH_DES`。有关更多信息，请参见以下内容：

- 要设置和管理安全 NFS，请参见《在 Oracle Solaris 11.2 中管理网络文件系统 》中的“管理安全 NFS 系统”。

Kerberos 验证

Kerberos 是 MIT 开发的验证系统。此发行版中包括使用 RPCSEC_GSS 的 Kerberos V5 客户端和服务器端实现。有关更多信息，请参见如何配置 Kerberos NFS 服务器 [104]。

使用安全 NFS 的 DES 加密

数据加密标准 (Data Encryption Standard, DES) 加密功能使用 56 位密钥进行数据加密。如果两个凭证用户或主体知道同一 DES 密钥，他们就可以通过使用此密钥加密和解密文本进行秘密通信。DES 是一种相对快速的加密机制。

仅使用 DES 密钥的风险是入侵者可以收集足够的使用相同密钥加密的密文消息，从而能够发现密钥并对这些消息进行解密。因此，安全系统（例如安全 NFS）需要经常更改密钥。

Diffie-Hellman 验证和安全 RPC

入侵者很难破解用于验证用户的 Diffie-Hellman (DH) 方法。客户机和服务器都有自己的私钥，它们将其与公钥一起使用以设计公用密钥。私钥也称为密钥 (secret key)。客户机和服务器使用公用密钥相互通信。公用密钥使用公认的加密功能（例如 DES）进行加密。

验证成功的基础是发送系统能够使用公用密钥来加密当前时间。然后，接收系统可以进行解密，并根据其当前时间进行检查。客户机和服务器上的时间必须同步。可以使用网络时间协议 (Network Time Protocol, NTP) 来同步时钟。Oracle Solaris 软件中包括了由美国特拉华大学开发的 NTP 公共域软件。相关文档可从 NTP Documentation（NTP 文档）网站获取。

公钥和私钥存储在 NIS 数据库中。NIS 将密钥存储在 `publickey` 映射中。此文件包含所有潜在用户的公钥和私钥。

系统管理员负责设置 NIS 映射以及为每个用户生成公钥和私钥。私钥以加密形式与用户口令存储在一起。此过程使私钥只让用户知道。

使用安全 RPC 管理验证

通过规定必须验证才能使用已挂载的 NFS 文件系统，可提高网络的安全性。

以下任务列表列出了为 NIS 和 NFS 配置安全 RPC 的过程。

表 10-1 使用安全 RPC 管理验证任务列表

任务	说明	有关说明
1. 启动 keyserver。	确保可以创建密钥以对用户进行验证。	如何重新启动安全 RPC Keyserver [187]
2. 在 NIS 主机上设置凭证。	确保主机上的 root 用户可以在 NIS 环境中进行验证。	如何为 NIS 主机设置 Diffie-Hellman 密钥 [187]
3. 为 NIS 用户提供密钥。	使用户可以在 NIS 环境中进行验证。	如何为 NIS 用户设置 Diffie-Hellman 密钥 [188]
4. 通过验证共享 NFS 文件。	使 NFS 服务器可以使用验证安全地保护共享的文件系统。	如何通过 Diffie-Hellman 验证共享 NFS 文件 [189]

▼ 如何重新启动安全 RPC Keyserver

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **验证 keyserv 守护进程是否正在运行。**

```
# svcs \*keyserv\*
STATE    STIME    FMRI
disabled Dec_14   svc:/network/rpc/keyserv
```

2. **如果 keyserver 服务未联机，则启用该服务。**

```
# svcadm enable network/rpc/keyserv
```

▼ 如何为 NIS 主机设置 Diffie-Hellman 密钥

在 NIS 域中的每个主机上执行此过程。

开始之前 您必须承担 root 角色。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. **如果缺省命名服务不是 NIS，则将 publickey 映射添加到命名服务。**

   a. **确认命名服务的 config/default 值不是 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"
```

如果 `config/default` 值为 `nis`，您可以在此处停止。

b. **将命名服务的 `publickey` 设置为 `nis`。**

```
# svccfg -s name-service/switch setprop config/publickey = astring: "nis"
# svccfg -s name-service/switch:default refresh
```

c. **确认 `publickey` 值。**

```
# 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
```

在此系统上，列出 `publickey` 的值，因为它不同于缺省值 `files`。

2. **使用 `newkey` 命令创建一个新密钥对。**

```
# newkey -h hostname
```

其中，*hostname* 是客户机的名称。

例 10-1 在 NIS 客户机上为 `root` 设置新密钥

在以下示例中，分配了 "Name Service Security"（名称服务安全）权限配置文件的管理员将 `earth` 设置为安全 NIS 客户机。

```
# 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.
#
```

▼ 如何为 NIS 用户设置 Diffie-Hellman 密钥

对 NIS 域中的每个用户执行此过程。

开始之前 必须登录到 NIS 主服务器才能为用户生成新密钥。必须已为您分配了 "Name Service Security"（名称服务安全）权限配置文件。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 为用户创建新密钥。

   ```
   # newkey -u username
   ```

   其中，*username* 是用户的名称。系统将提示键入口令。您可以键入通用口令。私钥是使用此通用口令以加密形式存储的。

2. 指示用户登录并键入 **chkey -p** 命令。
   此命令允许用户使用只有本人知道的口令重新加密其私钥。

注 - 可以使用 chkey 命令为用户创建新密钥对。

例 10-2 在 NIS 中设置并加密新用户密钥

在此示例中，超级用户将设置密钥。

```
# 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.
#
```

然后，用户 jdoe 使用专用口令对该密钥重新加密。

```
% 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...
```

▼ 如何通过 Diffie-Hellman 验证共享 NFS 文件

此过程通过要求访问验证来保护 NFS 服务器上的共享文件系统。

开始之前 必须在网络中启用 Diffie-Hellman 公钥验证。要启用网络验证，请完成如何为 NIS 主机设置 Diffie-Hellman 密钥 [187]中的步骤。

您必须是指定有 System Management（系统管理）权限配置文件的管理员才能执行此任务。有关更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用所指定的管理权限”。

1. 在 NFS 服务器上，通过 Diffie-Hellman 验证共享文件系统。

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

其中，*filesystem* 是共享的文件系统。

`-o sec=dh` 选项表示现在需要通过 `AUTH_DH` 验证后才能访问文件系统。

2. 在 NFS 客户机上，通过 Diffie-Hellman 验证挂载文件系统。

```
# mount -F nfs -o sec=dh server:filesystem mount-point
```

server	共享 *filesystem* 的系统的名称
filesystem	共享的文件系统的名称，如 `opt`
mount-point	挂载点的名称，例如 `/opt`

`-o sec=dh` 选项通过 `AUTH_DH` 验证挂载文件系统。

♦ ♦ ♦ 附 录 A

用于 Kerberos 的 DTrace 探测器

本附录介绍 DTrace 探测器和参数结构。有关其使用示例，请参见“对 Kerberos 服务使用 DTrace” [175]。

Kerberos 中的 DTrace 探测器

探测器指的是 DTrace 可将请求绑定到的程序位置或活动，以便执行一组操作。探测器由提供者定义并实现。提供者指的是内核可装入的模块，可使其探测器跟踪数据。

这些探测器用于用户静态定义跟踪 (User Statically Defined Tracing, USDT)。USDT 探测器设计为在用户范围内检查 Kerberos 协议。未提供用于静态定义跟踪的内核探测器。

可以创建脚本，以便相应的 DTrace 探测器记录您想要的信息（例如，栈跟踪、时间戳或函数的参数）。当触发探测器时，DTrace 会从探测器中收集数据并报告回给用户。如果没有为探测器指定任何操作，DTrace 将记录每次触发探测器的时间及其在哪个 CPU 触发。

Kerberos DTrace 探测器是仿照 Kerberos 消息类型建立的，如 RFC4120: The Kerberos Network Authentication Service (V5) (http://www.ietf.org/rfc/rfc4120.txt)（RFC4120：Kerberos 网络验证服务 (V5)）中所述。这些探测器可供 `libkrb5/mech_krb5` 的使用者使用，包括那些通过 `libgss` 使用 `mech_krb5` 的应用程序。探测器的功能分为消息创建和使用以及发送和接收。有关 `libgss` 的更多信息，请参见 `libgss`(3LIB) 手册页。

要使用探测器，请指定 `kerberos` 提供者、探测器的名称（例如 `krb_message-recv`）和参数。有关示例，请参见“对 Kerberos 服务使用 DTrace” [175]。

Kerberos DTrace 探测器的定义

用于 `KRB_AP_REP` 的探测器：

```
kerberos$pid:::krb_ap_rep-make
kerberos$pid:::krb_ap_rep-read

args[0]        krbinfo_t *
args[1]        kaprepinfo_t *
```

用于 KRB_AP_REQ 的探测器：

```
kerberos$pid:::krb_ap_req-make
kerberos$pid:::krb_ap_req-read

args[0]        krbinfo_t *
args[1]        kapreqinfo_t *
args[2]        kticketinfo_t *
args[3]        kauthenticatorinfo_t *
```

用于 KRB_KDC_REP 的探测器：

```
kerberos$pid:::krb_kdc_rep-make
kerberos$pid:::krb_kdc_rep-read

args[0]        krbinfo_t *
args[1]        kdcrepinfo_t *
args[2]        kticketinfo_t *
```

用于 KRB_KDC_REQ 的探测器：

```
kerberos$pid:::krb_kdc_req-make
kerberos$pid:::krb_kdc_req-read

args[0]        krbinfo_t *
args[1]        kdcreqinfo_t *
```

用于 KRB_CRED 的探测器：

```
kerberos$pid:::krb_cred-make
kerberos$pid:::krb_cred-read

args[0]        krbinfo_t *
args[1]        kcredinfo_t *
```

用于 KRB_ERROR 的探测器：

```
kerberos$pid:::krb_error-make
kerberos$pid:::krb_error-read

args[0]        krbinfo_t *
args[1]        kerrorinfo_t *
```

用于 KRB_PRIV 的探测器：

```
kerberos$pid:::krb_priv-make
kerberos$pid:::krb_priv-read

args[0]        krbinfo_t *
args[1]        kprivinfo_t *
```

用于 KRB_SAFE 的探测器：

```
kerberos$pid:::krb_safe-make
kerberos$pid:::krb_safe-read

args[0]        krbinfo_t *
args[1]        ksafeinfo_t *
```

用于发送和接收消息的探测器

```
kerberos$pid:::krb_message-recv
kerberos$pid:::krb_message-send

args[0]        krbinfo_t *
args[1]        kconninfo_t *
```

Kerberos 中的 DTrace 参数结构

在某些情况下，部分参数的值可以为 0 或为空。在设计上，Kerberos 参数结构基本符合 RFC4120: The Kerberos Network Authentication Service (V5) (http://www.ietf.org/rfc/rfc4120.txt)（RFC4120：Kerberos 网络验证服务 (V5)）中所述的要求。

DTrace 中的 Kerberos 消息信息

```
typedef struct krbinfo {
uint8_t krb_version;                /* protocol version number (5) */
string krb_message_type;            /* Message type (AS_REQ(10), ...) */
uint64_t krb_message_id;            /* message identifier */
uint32_t krb_message_length;        /* message length */
uintptr_t krb_message;              /* raw ASN.1 encoded message */
} krbinfo_t;
```

注 - Kerberos 协议不具有消息标识符。krb_message_id 标识符特定于 Kerberos 提供者，并用于在 make/read 探测器与 send/recv 探测器之间链接消息。

DTrace 中的 Kerberos 连接信息

```
typedef struct kconninfo {
string kconn_remote;                /* remote host address */
string kconn_local;                 /* local host address */
uint16_t kconn_localport;           /* local port */
uint16_t kconn_remoteport;          /* remote port */
string kconn_protocol;              /* protocol (ipv4, ipv6) */
```

```
string kconn_type;                /* transport type (udp, tcp) */
} kconninfo_t;
```

DTrace 中的 Kerberos 验证者信息

```
typedef struct kauthenticatorinfo {
string kauth_client;              /* client principal identifier */
string kauth_cksum_type;          /* type of checksum (des-cbc, ...) */
uint32_t kauth_cksum_length;      /* length of checksum */
uintptr_t kauth_cksum_value;      /* raw checksum data */
uint32_t kauth_cusec;             /* client time, microseconds */
uint32_t kauth_ctime;             /* client time in seconds */
string kauth_subkey_type;         /* sub-key type (des3-cbc-sha1, ...) */
uint32_t kauth_subkey_length;     /* sub-key length */
uintptr_t kauth_subkey_value;     /* sub-key data */
uint32_t kauth_seq_number;        /* sequence number */
string kauth_authorization_data;  /* top-level authorization types
(AD-IF-RELEVANT, ... ) */
} kauthenticatorinfo_t;

typedef struct kticketinfo_t {
string kticket_server;            /* service principal identifier */
uint32_t kticket_enc_part_kvno;   /* key version number */
string kticket_enc_part_etype;    /* enc type of encrypted ticket */
string kticket_enc_flags;         /* ticket flags (forwardable, ...) */
string kticket_enc_key_type;      /* key type (des3-cbc-sha1, ...) */
uint32_t kticket_enc_key_length;  /* key length */
uintptr_t kticket_enc_key_value;  /* key data */
string kticket_enc_client;        /* client principal identifier */
string kticket_enc_transited;     /* list of transited Kerberos realms */
string kticket_enc_transited_type;  /* encoding type */
uint32_t kticket_enc_authtime;    /* time of initial authentication */
uint32_t kticket_enc_starttime;   /* ticket start time in seconds */
uint32_t kticket_enc_endtime;     /* ticket end time in seconds */
uint32_t kticket_enc_renew_till;  /* ticket renewal time in seconds */
string kticket_enc_addresses;     /* addresses associated with ticket */
string kticket_enc_authorization_data;  /* list of top-level auth types */
} kticketinfo_t;

typedef struct kdcreqinfo {
string kdcreq_padata_types;       /* list of pre-auth types */
string kdcreq_kdc_options;        /* requested ticket flags */
string kdcreq_client;             /* client principal identifier */
string kdcreq_server;             /* server principal identifier */
uint32_t kdcreq_from;             /* requested start time in seconds */
uint32_t kdcreq_till;             /* requested end time in seconds */
uint32_t kdcreq_rtime;            /* requested renewal time in seconds */
uint32_t kdcreq_nonce;            /* nonce for replay detection */
string kdcreq_etype;              /* preferred encryption types */
string kdcreq_addresses;          /* list of requested ticket addresses */
string kdcreq_authorization_data;  /* list of top-level auth types */
```

```
uint32_t kdcreq_num_additional_tickets; /* number of additional tickets */
} kdcreqinfo_t;

typedef struct kdcrepinfo {
string kdcrep_padata_types;         /* list of pre-auth types */
string kdcrep_client;               /* client principal identifier */
uint32_t kdcrep_enc_part_kvno;      /* key version number */
string kdcrep_enc_part_etype;       /* enc type of encrypted KDC reply */
string kdcrep_enc_key_type;         /* key type (des3-cbc-sha1, ...) */
uint32_t kdcrep_enc_key_length;     /* key length */
uintptr_t kdcrep_enc_key_value;     /* key data */
string kdcrep_enc_last_req;         /* times of last request of principal */
uint32_t kdcrep_enc_nonce;          /* nonce for replay detection */
uint32_t kdcrep_enc_key_expiration; /* expiration time of client's key */
string kdcrep_enc_flags;            /* ticket flags */
uint32_t kdcrep_enc_authtime;       /* time of authentication of ticket */
uint32_t kdcrep_enc_starttime;      /* ticket start time in seconds */
uint32_t kdcrep_enc_endtime;        /* ticket end time in seconds */
uint32_t kdcrep_enc_renew_till;       /* ticket renewal time in seconds*/
string kdcrep_enc_server;           /* server principal identifier */
string kdcrep_enc_caddr;            /* zero or more client addresses */
} kdcrepinfo_t;

typedef struct kapreqinfo {
string kapreq_ap_options;           /* options (use-session-key,... ) */
uint32_t kapreq_authenticator_kvno; /* key version number */
string kapreq_authenticator_etype;  /* enc type of authenticator */
} kapreqinfo_t;

typedef struct kaprepinfo {
uint32_t kaprep_enc_part_kvno;      /* key version number */
string kaprep_enc_part_etype;       /* enc type of encrypted AP reply */
uint32_t kaprep_enc_ctime;          /* client time in seconds */
uint32_t kaprep_enc_cusec;          /* client time, microseconds portion */
string kaprep_enc_subkey_type;      /* sub-key type */
uint32_t kaprep_enc_subkey_length;  /* sub-key length */
uintptr_t kaprep_enc_subkey_value;  /* sub-key data */
uint32_t kaprep_enc_seq_number;     /* sequence number */
} kaprepinfo_t;

typedef struct kerrorinfo {
uint32_t kerror_ctime;              /* client time in seconds */
uint32_t kerror_cusec;              /* client time, microseconds */
uint32_t kerror_stime;              /* server time in seconds */
uint32_t kerror_susec;              /* server time, microseconds */
string kerror_error_code;           /* error code (KRB_AP_ERR_SKEW, ...) */
string kerror_client;               /* client principal identifier */
string kerror_server;               /* server principal identifier */
string kerror_e_text;               /* additional error text */
string kerror_e_data;               /* additional error data */
} kerrorinfo_t;

typedef struct ksafeinfo {
uintptr_t ksafe_user_data;          /* raw application specific data */
```

```
uint32_t ksafe_timestamp;          /* time of sender in seconds */
uint32_t ksafe_usec;               /* time of sender, microseconds */
uint32_t ksafe_seq_number;         /* sequence number */
string ksafe_s_address;            /* sender's address */
string ksafe_r_address;            /* recipient's address */
string ksafe_cksum_type;           /* checksum type (des-cbc, ...) */
uint32_t ksafe_cksum_length;       /* length of checksum */
uintptr_t ksafe_cksum_value;       /* raw checksum data */
} ksafeinfo_t;

typedef struct kprivinfo {
uint32_t kpriv_enc_part_kvno;      /* key version number */
string kpriv_enc_part_etype;       /* enc type of encrypted message */
uintptr_t kpriv_enc_user_data;     /* raw application specific data */
uint32_t kpriv_enc_timestamp;      /* time of sender in seconds */
uint32_t kpriv_enc_usec;           /* time of sender, microseconds */
uint32_t kpriv_enc_seq_number;     /* sequence number */
string kpriv_enc_s_address;        /* sender's address */
string kpriv_enc_r_address;        /* recipient's address */
} kprivinfo_t;

typedef struct kcredinfo {
uint32_t kcred_enc_part_kvno;      /* key version number */
string kcred_enc_part_etype;       /* enc type of encrypted message */
uint32_t kcred_tickets;            /* number of tickets */
uint32_t kcred_enc_nonce;          /* nonce for replay detection */
uint32_t kcred_enc_timestamp;      /* time of sender in seconds */
uint32_t kcred_enc_usec;           /* time of sender, microseconds */
string kcred_enc_s_address;        /* sender's address */
string kcred_enc_r_address;        /* recipient's address */
} kcredinfo_t;
```

安全词汇表

Access Control List, ACL（访问控制列表）	与传统的 UNIX 文件保护相比，访问控制列表 (access control list, ACL) 可提供更为精细的文件安全性。例如，通过 ACL 可以让组获得对某个文件的读取权限，而仅允许该组中的一个成员获得对该文件的写入权限。
admin principal（admin 主体）	名称形式为 *username*`/admin` 的用户主体（如 `jdoe/admin`）。与一般用户主体相比，admin 主体可以拥有更多特权（例如，可以更改策略）。另请参见 principal name（主体名称）和 user principal（用户主体）。
AES	Advanced Encryption Standard（高级加密标准）。一种对称的 128 位块数据加密技术。美国政府在 2000 年 10 月采用该种算法的 Rijndael 变体作为其加密标准。AES 从而取代了 user principal（用户主体）加密方法成为政府的加密标准。
algorithm（算法）	加密算法。这是一种确立的递归计算过程，用于对输入执行加密或散列操作。
application server（应用服务器）	请参见 network application server（网络应用服务器）。
asynchronous audit event（异步审计事件）	异步事件在系统事件中属于少数。这些事件不与任何进程关联，因此没有任何进程可供阻塞并在以后唤醒。例如，初始系统引导和 PROM 进入和退出事件都是异步事件。
audit files（审计文件）	二进制审计日志。审计文件单独存储在一个审计文件系统中。
audit policy（审计策略）	决定要记录的审计事件的全局设置和按用户设置。通常，应用于审计服务的全局设置会影响审计迹所包括的可选信息。`cnt` 和 `ahlt` 这两个设置会影响系统在填充审计队列时执行的操作。例如，审计策略可能要求每条审计记录都包含一个序列号。
audit trail（审计迹）	来自所有主机的所有审计文件的集合。
authenticated rights profile（需要验证权限配置文件）	指定有这类 rights profile（权限配置文件）的用户或角色执行配置文件中的操作之前需要键入口令。此行为类似于 `sudo` 行为。口令的有效期可配置。

authentication（验证）	验证主体所声明的身份的过程。
authenticator（验证者）	当客户机从 KDC 请求票证以及从服务器请求服务时，会传递验证者。这些验证者包含使用仅对客户机和服务器公开的会话密钥所生成的信息，这些信息可以作为最新来源进行检验，从而表明事务是安全的。验证者可与票证一起使用来验证用户主体。验证者中包括用户的主体名称、用户主机的 IP 地址，以及时间戳。与票证不同，验证者只能使用一次，通常在请求访问服务时使用。验证者是使用特定客户机和服务器的会话密钥进行加密的。
authorization（授权）	1. 在 Kerberos 中，是指决定主体是否可以使用服务，允许主体访问哪些对象，以及可对每个对象执行的访问操作类型的过程。 2. 在用户权限管理中，是指可以指定给角色或用户（或嵌入权限配置文件中）的权限，此权限用于执行安全策略原本禁止的一类操作。授权在用户应用程序级别（而不是内核级别）实施。
basic set（基本特权集合）	登录时为用户进程指定的特权集合。在未修改的系统上，每个用户的初始可继承特权集合等同于登录时获取的基本特权集合。
Blowfish	一种对称块加密算法，它采用 32 位到 448 位的可变长度密钥。其作者 Bruce Schneier 声称 Blowfish 已针对密钥不经常更改的应用程序进行优化。
client principal（客户机主体）	(RPCSEC_GSS API) 是指使用受 RPCSEC_GSS 保护的网络服务的客户机（用户或应用程序）。客户机主体名称将以 `rpc_gss_principal_t` 结构的形式进行存储。
client（客户机）	狭义上讲，是指代表用户使用网络服务的进程，例如，使用 `rlogin` 的应用程序。在某些情况下，服务器本身即可是其他某个服务器或服务的客户机。 广义上讲，是指 a) 接收 Kerberos 凭证的主机，以及 b) 使用由服务器提供的服务的主机。 非正式地讲，是指使用服务的主体。
clock skew（时钟相位差）	所有参与 Kerberos 验证系统的主机上的内部系统时钟可以相差的最大时间量。如果任意两台参与主机之间的时间偏差超过了时钟相位差，则请求会被拒绝。可以在 `krb5.conf` 文件中指定时钟相位差。
confidentiality（保密性）	请参见 privacy（保密性）。
consumer（使用者）	在 Oracle Solaris 的加密框架功能中，使用者是指使用提供者提供的加密服务的用户。使用者可以是应用程序、最终用户或内核操作。例如，Kerberos、IKE 和 IPsec 便属于使用者。有关提供者的示例，请参见 provider（提供者）。
credential cache（凭证高速缓存）	包含从 KDC 接收的凭证的存储空间（通常为文件）。

credential（凭证）	包括票证及匹配的会话密钥的信息软件包。用于验证主体的身份。另请参见 ticket（票证）和 session key（会话密钥）。
cryptographic algorithm（密码算法）	请参见 algorithm（算法）。
DES	Data Encryption Standard（数据加密标准）。一种对称密钥加密方法，开发于 1975 年，1981 年由 ANSI 标准化为 ANSI X.3.92。DES 使用 56 位密钥。
device allocation（设备分配）	用户级别的设备保护。设备分配强制规定一次只能由一个用户独占使用一台设备。重用设备之前，将清除设备数据。可以使用授权来限制允许分配设备的用户。
device policy（设备策略）	内核级别的设备保护。设备策略在设备上作为两个特权集合实现。一个特权集合控制对设备的读取权限，另一个特权集合控制对设备的写入权限。另请参见 policy（策略）。
Diffie-Hellman protocol（Diffie-Hellman 协议）	也称为公钥密码学。Diffie 和 Hellman 于 1976 年开发的非对称密钥一致性协议。使用该协议，两个用户可以在以前没有任何密钥的情况下通过不安全的介质交换密钥。Diffie-Hellman 由 Kerberos 使用。
digest（摘要）	请参见 message digest（消息摘要）。
DSA	Digital Signature Algorithm（数字签名算法）。一种公钥算法，采用大小可变（512 位到 4096 位）的密钥。美国政府标准 DSS 可达 1024 位。DSA 的输入依赖于 SHA1。
ECDSA	Elliptic Curve Digital Signature Algorithm（椭圆曲线数字签名算法）。一种基于椭圆曲线数学运算的公钥算法。在生成相同长度的签名时，所需的 ECDSA 密钥大小明显小于 DSA 公钥大小。
effective set（有效特权集合）	当前对进程有效的特权集合。
flavor（特性）	以前，*security flavor*（安全特性）和 *authentication flavor*（验证特性）具有相同的含义，都是表示验证类型 (AUTH_UNIX, AUTH_DES, AUTH_KERB) 的特性。RPCSEC_GSS 也是一种安全特性，虽然它除了验证之外还提供完整性和保密性服务。
forwardable ticket（可转发票证）	一种票证，可供客户机在不需要完成远程主机上的完整验证过程的情况下用于请求此主机票证。例如，如果用户 `david` 登录到用户 `jennifer` 的计算机时获取了一张可转发票证，则 `david` 不必获取新的票证（从而对自身进行重新验证）即可登录到自己的计算机。另请参见 proxiable ticket（可代理票证）。
FQDN	Fully qualified domain name（全限定域名）。例如，`central.example.com`（与简单的 `denver` 相对）。
GSS-API	Generic Security Service Application Programming Interface（通用安全服务应用编程接口）。为各种模块化安全服务（包括 Kerberos

	服务）提供支持的网络层。GSS-API 可用于安全验证服务、完整性服务和保密性服务。另请参见 authentication（验证）、integrity（完整性）和 privacy（保密性）。
hardening（强化）	为了删除主机中固有的安全漏洞而对操作系统的缺省配置进行的修改。
hardware provider（硬件提供者）	在 Oracle Solaris 的加密框架功能中，是指设备驱动程序及其硬件加速器。硬件提供者使计算机系统不必执行开销很大的加密操作，从而可释放 CPU 资源以用于其他用途。另请参见 provider（提供者）。
host principal（host 主体）	服务主体的一个特定实例，其中将主体（由主名称 `host` 表示）设置为提供一系列网络服务，如 `ftp`、`rcp` 或 `rlogin`。例如，`host/central.example.com@EXAMPLE.COM` 便是一个 host 主体。另请参见 server principal（服务器主体）。
host（主机）	可通过网络进行访问的系统。
inheritable set（可继承特权集合）	进程可以通过调用 `exec` 而继承的特权集合。
initial ticket（初始票证）	直接颁发（即，不基于现有的票证授予票证）的票证。某些服务（如用于更改口令的应用程序）可能需要将票证标记为 `initial`，以便使其自身确信客户机知晓其密钥。这种保证非常重要，因为初始票证表明客户机最近已进行了自我验证（而并非依赖于存在时间可能较长的票证授予票证）。
instance（实例）	实例是主体名称的第二个部分，用于限定主体的主名称。对于服务主体，实例是必需的。实例就是主机的全限定域名，例如 `host/central.example.com`。对于用户主体，实例是可选的。但是请注意，`jdoe` 和 `jdoe/admin` 都是唯一的主体。另请参见 primary（主）、principal name（主体名称）、service principal（服务主体）和 user principal（用户主体）。
integrity（完整性）	一种安全服务，除了用于用户验证之外，还用于通过加密校验和来验证传输数据的有效性。另请参见 authentication（验证） 和 privacy（保密性）。
invalid ticket（无效票证）	尚未成为可用票证的以后生效的票证。应用服务器将拒绝无效票证，直到此票证生效为止。要使无效票证生效，必须在其开始时间已过后，由客户机通过 TGS 请求将其提供给 KDC，同时设置 `VALIDATE` 标志。另请参见 postdated ticket（以后生效的票证）。
KDC	Key Distribution Center（密钥分发中心）。具有以下三个 Kerberos V5 组件的计算机： ■ 主体和密钥数据库 ■ 验证服务

- 票证授予服务

每个领域都具有一个主 KDC，并且应该具有一个或多个从 KDC。

Kerberos

是指一种验证服务、此服务所使用的协议或者用于实现此服务的代码。

Oracle Solaris 中的 Kerberos 实现主要基于 Kerberos V5 实现。

虽然在技术方面有所不同，但是在 Kerberos 文档中经常会互换使用 "Kerberos" 和 "Kerberos V5"。

Kerberos（也可写成 Cerberus）在希腊神话中是指守护地狱之门的三头凶悍猛犬。

Kerberos policy（Kerberos 策略）

管理 Kerberos 服务中口令的使用的规则集合。这些策略可以控制主体的访问权限或票证参数（如生命周期）。

key（密钥）

1. 通常是指以下两种主要密钥类型之一：

- 对称密钥 – 与解密密钥相同的加密密钥。对称密钥用于对文件进行加密。
- 非对称密钥或公钥 – 在公钥算法（如 Diffie-Hellman 或 RSA）中使用的密钥。公钥包括仅对一个用户公开的私钥、服务器或通用资源所使用的公钥，以及包含这两者的私钥/公钥对。私钥 (private key) 也称为密钥 (secret key)。公钥也称为共享密钥或公用密钥。

2. 密钥表文件中的项（主体名称）。另请参见 keytab file（密钥表文件）。

3. 在 Kerberos 中，是指加密密钥，此类密钥分为以下三种类型：

- 私钥 – 由主体和 KDC 共享并在系统范围之外分发的加密密钥。另请参见 private key（私钥）。
- 服务密钥 – 此密钥与私钥的用途相同，但由服务器和服务使用。另请参见 service key（服务密钥）。
- 会话密钥 – 在两个主体之间使用的临时加密密钥，其生命周期仅限于单个登录会话的持续时间。另请参见 session key（会话密钥）。

keystore（密钥库）

密钥库包含用于应用程序检索的口令、口令短语、证书，以及其他验证对象。密钥库可特定于一种技术，或特定于多个应用程序使用的一个位置。

keytab file（密钥表文件）

包含一个或多个密钥（主体）的密钥表文件。主机或服务使用密钥表文件的方式与用户使用口令的方式大致相同。

kvno

Key version number（密钥版本号）。按照生成顺序跟踪特定密钥的序列号。kvno 最高则表示密钥最新。

least privilege（最小特权）	一种安全模型，该模型仅向指定进程提供超级用户功能的某个子集。最小特权模型为一般用户指定可以用来执行个人管理任务（如挂载文件系统和更改文件的所有权）的足够特权。另一方面，仅使用完成该任务所需的特权运行进程，而不是使用超级用户的完全功能模式（即所有特权）。对非 root 用户而言，可以包含由于编程错误而导致的损坏（如缓冲区溢出），该用户对重要功能（如读取或写入受保护的系统文件或停止计算机）没有访问权限。
limit set（限制特权集合）	对哪些特权可用于进程及其子进程的外部限制。
MAC	1. 请参见 message authentication code, MAC（消息验证代码）。 2. 也称为标签设置操作。在政府安全术语中，MAC 是指 Mandatory Access Control（强制访问控制）。例如，Top Secret（绝密）和 Confidential（机密）之类的标签便是 MAC。MAC 与 DAC 相对，后者是指 Discretionary Access Control（自主访问控制）。例如，UNIX 权限便是一个 DAC。 3. 在硬件中，是指 LAN 中的唯一系统地址。如果系统位于以太网中，则 MAC 是指以太网地址。
master KDC（主 KDC）	每个领域中的主要 KDC，包括 Kerberos 管理服务器 `kadmind`，以及验证和票证授予守护进程 `krb5kdc`。每个领域至少都必须具有一个主 KDC，可以具有多个 KDC 副本或从 KDC，这些 KDC 为客户机提供验证服务。
MD5	一种重复加密散列函数，用于进行消息验证（包含数字签名）。该函数于 1991 年由 Rivest 开发。其使用已过时。
mechanism（机制）	1. 指定加密技术以实现数据验证或保密的软件包。例如：Kerberos V5、Diffie-Hellman 公钥。 2. 在 Oracle Solaris 的加密框架功能中，是指用于特殊用途的算法的实现。例如，应用于验证的 DES 机制（如 CKM_DES_MAC）与应用于加密的 DES 机制（如 CKM_DES_CBC_PAD）不同。
message authentication code, MAC（消息验证代码）	MAC 可确保数据的完整性，并验证数据的来源。MAC 不能防止窃听。
message digest（消息摘要）	消息摘要是从消息中计算所得的散列值。此散列值几乎可唯一地标识消息。摘要对检验文件的完整性非常有用。
minimization（最小安装）	运行服务器所需的最小操作系统安装。不安装与服务器操作不直接相关的任何软件，或者在安装之后即删除。
name service scope（名称服务范围）	允许角色在其中执行操作的范围，即，由指定的命名服务（如 NIS 或 LDAP）提供服务的单个主机或所有主机。
network application server（网络应用服务器）	提供网络应用的服务器，如 `ftp`。一个领域可以包含多个网络应用服务器。

network policies（网络策略）
: 网络实用程序为了保护网络通信而配置的设置。有关网络安全性的信息，请参见《在 Oracle Solaris 11.2 中确保网络安全 》。

nonattributable audit event（无归属审计事件）
: 无法确定其触发者的审计事件，如 AUE_BOOT 事件。

NTP
: Network Time Protocol（网络时间协议）。由特拉华大学开发的软件，可用于在网络环境中管理准确时间或网络时钟同步，或者同时管理这两者。可以使用 NTP 在 Kerberos 环境中维护时钟相位差。另请参见 clock skew（时钟相位差）。

PAM
: Pluggable Authentication Module（可插拔验证模块）。一种框架，允许使用多种验证机制而不必重新编译运行这些机制的服务。PAM 可用于在登录时初始化 Kerberos 会话。

passphrase（口令短语）
: 一种短语，用于验证某个私钥是否是由口令短语用户创建。理想的口令短语应包含 10-30 个字符，请混合使用字母和数字字符，并且避免简单的文本结构和名称。使用私钥对通信执行加密和解密操作时，系统会提示您提供口令短语进行验证。

password policy（口令策略）
: 可用于生成口令的加密算法，还可以指与口令有关的更普遍的问题，如必须对口令进行更改的频率，允许的口令尝试次数以及其他安全注意事项。安全策略需要口令。口令策略可能要求使用 AES 算法对口令进行加密，并可能对口令强度提出进一步要求。

permitted set（允许特权集合）
: 可供进程使用的特权集合。

policy for public key technologies（公钥技术的策略）
: 在密钥管理框架 (Key Management Framework, KMF) 中，所实现的策略是管理证书的使用。KMF 策略数据库可以对由 KMF 库管理的密钥和证书的使用施加约束。

policy in the Cryptographic Framework（加密框架中的策略）
: 在 Oracle Solaris 的加密框架功能中，所实现的策略是禁用现有的加密机制。从而使这些机制不可使用。加密框架中的策略可能会阻止使用提供者（如 DES）提供的特殊机制，如 `CKM_DES_CBC`。

policy（策略）
: 一般而言，是指影响或决定决策和操作的操作规划或操作过程。对于计算机系统，策略通常表示安全策略。站点的安全策略是规则集合和相关措施，可用于定义所处理信息的敏感度并防止信息受到未经授权的访问。例如，安全策略可能要求对系统进行审计，必须分配设备才能使用，以及每六周必须更改一次口令。

  有关在 Oracle Solaris OS 特定区域中实施策略的信息，请参见 audit policy（审计策略）、policy in the Cryptographic Framework（加密框架中的策略）、device policy（设备策略）、Kerberos policy（Kerberos 策略）、password policy（口令策略）和 rights policy（权限策略）。

postdated ticket（以后生效的票证）
以后生效的票证直到创建之后的某一指定时间才能开始生效。此类票证对于计划在深夜运行的批处理作业等情况非常有用，因为在运行批处理作业之前无法使用该票证（即使被盗）。颁发以后生效的票证时，将以 `invalid` 状态颁发该票证，并在出现以下情况之前一直保持此状态：a) 票证开始时间已过，并且 b) 客户机请求 KDC 进行验证。通常，以后生效的票证在票证授予票证的截止时间之前会一直有效。但是，如果将以后生效的票证标记为 `renewable`，则通常会将其生命周期设置为等于票证授予票证的整个生命周期的持续时间。另请参见 invalid ticket（无效票证）和 renewable ticket（可更新票证）。

primary（主）
主体名称的第一部分。另请参见 instance（实例）、principal name（主体名称）和 realm（领域）。

principal name（主体名称）
1. 主体的名称，格式为 *primary/instance@REALM*。另请参见 instance（实例）、primary（主）和 realm（领域）。

2.(RPCSEC_GSS API) 请参见 client principal（客户机主体）和 server principal（服务器主体）。

principal（主体）
1. 参与网络通信并且具有唯一名称的客户机/用户或服务器/服务实例。Kerberos 事务涉及主体之间（服务主体与用户主体）或主体与 KDC 之间的交互。换言之，主体是 Kerberos 可为其指定票证的唯一实体。另请参见 principal name（主体名称）、service principal（服务主体）和 user principal（用户主体）。

2.(RPCSEC_GSS API) 请参见 client principal（客户机主体）和 server principal（服务器主体）。

principle of least privilege（最小特权原则）
请参见 least privilege（最小特权）。

privacy（保密性）
一种安全服务，其中传输的数据加密之后才会发送。保密性还包括数据完整性和用户验证。另请参见 authentication（验证）、integrity（完整性）和 service（服务）。

private key（私钥）
为每个用户主体提供的密钥，并且只对主体的用户和 KDC 公开。对于用户主体，密钥基于用户的口令。另请参见 key（密钥）。

private-key encryption（私钥加密）
采用私钥加密时，发送者和接收者使用相同的加密密钥。另请参见 public-key encryption（公钥加密）。

privilege escalation（特权升级）
可以访问在所指定权限（包括覆盖缺省设置的权限）允许的资源范围以外的资源。特权升级的结果是某个进程可以执行未经授权的操作。

privilege model（特权模型）
计算机系统上比超级用户模型更为严格的安全模型。在特权模型中，进程需要具有相应的特权才能运行。系统管理可以分为多个独立的部分，这些部分基于管理员在其进程中所具有的特权。可以将特权指定给管理员的登录过程。或者，可以指定特权只对特定命令有效。

privilege set（特权集合）

特权的集合。每个进程都有四个特权集合，用于确定进程是否可以使用特定特权。请参见 limit set（限制特权集合）、effective set（有效特权集合）、permitted set（允许特权集合）和 inheritable set（可继承特权集合）。

此外，特权的 basic set（基本特权集合）是指登录时为用户进程指定的特权集合。

privilege-aware（可识别特权）

在其代码中启用和禁用特权的程序、脚本和命令。在生产环境中，启用的特权必须提供给进程，例如，通过要求程序的用户使用将特权添加到程序中的权限配置文件。有关特权的完整说明，请参见 `privileges(5)` 手册页。

privilege（特权）

1. 通常是指在某个计算机系统上执行一般用户所无法执行的操作的能力。超级用户特权是向超级用户授予的所有 rights（权限）。特权用户或特权应用程序是指获得了额外权限的用户或应用程序。

2. Oracle Solaris 系统中的进程具有的独立权限。与 `root` 相比，特权可提供更为精细的进程控制。特权是在内核中定义和实施的。特权也称为进程特权或内核特权。有关特权的完整说明，请参见 `privileges(5)` 手册页。

privileged application（特权应用程序）

可以覆盖系统控制的应用程序。该应用程序可以检查安全属性（如特定的 UID、GID、授权或特权）。

privileged user（特权用户）

计算机系统上为其指定的权限高于一般用户权限的用户。另请参见 trusted users（可信用户）。

profile shell（配置文件 shell）

在权限管理中，角色（或用户）可通过该 shell 从命令行运行指定给角色权限配置文件的任何特权应用程序。配置文件 shell 版本与系统上可用的 shell 对应（例如 `bash` 的 `pfbash` 版本）。

provider（提供者）

在 Oracle Solaris 的加密框架功能中，是指为使用者提供的加密服务。例如，PKCS #11 库、内核加密模块和硬件加速器便是提供者。提供者可插入到加密框架中，因此也称为插件。有关使用者的示例，请参见 consumer（使用者）。

proxiable ticket（可代理票证）

可供服务用于代表客户机执行客户机操作的票证。因此，可以说服务充当客户机的代理。使用该票证，服务便可具有客户机的身份。服务可以使用可代理票证来获取其他服务的服务票证，但是不能获取票证授予票证。可代理票证与可转发票证之间的区别在于可代理票证只对单项操作有效。另请参见 forwardable ticket（可转发票证）。

public object（公共对象）

`root` 用户所拥有且全局可读的文件，如 `/etc` 目录中的任何文件。

public-key encryption（公钥加密）

一种加密方案，其中每个用户都有两个密钥：一个是公钥，一个是私钥。采用公钥加密时，发送者使用接收者的公钥对消息进行加密，而

	接收者则使用私钥对其进行解密。Kerberos 服务是一种私钥系统。另请参见 private-key encryption（私钥加密）。
QOP	Quality of Protection（保护质量）。用于选择与完整性服务或保密性服务结合使用的加密算法的参数。
RBAC	Role-based access control（基于角色的访问控制），Oracle Solaris 的一项用户权限管理功能。请参见 rights（权限）。
RBAC policy（RBAC 策略）	请参见 rights policy（权限策略）。
realm（领域）	1. 由单个 Kerberos 数据库以及一组密钥分发中心 (Key Distribution Center, KDC) 提供服务的逻辑网络。 2. 主体名称的第三部分。对于主体名称 `jdoe/admin@CORP.EXAMPLE.COM`，领域为 `CORP.EXAMPLE.COM`。另请参见 principal name（主体名称）。
reauthentication（重新验证）	执行计算机操作需要提供口令。通常，`sudo` 操作需要重新验证。需要验证权限配置文件可包含需要重新验证的命令。请参见 authenticated rights profile（需要验证权限配置文件）。
relation（关系）	在 `kdc.conf` 或 `krb5.conf` 文件中定义的配置变量或关系。
renewable ticket（可更新票证）	由于票证的生命周期过长会存在安全风险，因此可以将票证指定为 `renewable`。可更新票证有两个截止时间：a) 票证的当前实例的截止时间，b) 任意票证的最长生命周期。如果客户机需要继续使用某票证，则可在首次失效之前更新此票证。例如，某个票证的有效期为 1 小时，所有票证的最长生命周期为 10 小时。如果持有票证的客户机希望保留此票证的时间长于 1 小时，则必须更新此票证。当某个票证达到最长票证生命周期时，便会自动到期，并且无法更新。
rights policy（权限策略）	与命令关联的安全策略。当前，`solaris` 是 Oracle Solaris 的有效策略。`solaris` 策略可识别特权和扩展特权策略、授权及 `setuid` 安全属性。
rights profile（权限配置文件）	也称为配置文件。指的是可以指定给角色或用户的安全设置覆盖值的集合。权限配置文件可包括授权、特权、具有安全属性的命令和称为补充配置文件的其他权限配置文件。
rights（权限）	对超级用户模型（管理员对系统要么具有全部控制权要么毫无控制权）的替代。通过用户权限管理和进程权限管理，组织可划分超级用户的特权并将其指定给用户或角色。Oracle Solaris 中的权限实施方式有内核特权、授权和以特定 UID 或 GID 运行进程的能力。可在 rights profile（权限配置文件）和 role（角色）中收集权限。
role（角色）	一种用于运行特权应用程序的特殊身份，仅指定用户才能承担此身份。

RSA	获取数字签名和公钥密码系统的方法。该方法于 1978 年首次由其开发者 Rivest、Shamir 和 Adleman 介绍。
scan engine（扫描引擎）	第三方应用程序，驻留在外部主机上，可检查文件中是否含有已知病毒。
SEAM	这是 Solaris 系统上的 Kerberos 初始版本的产品名。该产品基于麻省理工学院开发的 Kerberos V5 技术。SEAM 现在称为 Kerberos 服务。其特性与 MIT 版本仍稍有不同。
secret key（密钥）	请参见 private key（私钥）。
Secure Shell（安全 Shell）	一种特殊协议，用于在不安全的网络中进行安全远程登录并提供其他安全网络服务。
security attributes（安全属性）	是指当超级用户以外的用户运行管理命令时，可使此命令成功执行的安全策略覆盖项。在超级用户模型中，`setuid root` 和 `setgid` 程序都是安全属性。将这些属性应用于某命令时，此命令便会成功执行，而与运行它的用户无关。在 privilege model（特权模型）中，内核特权及其他 rights（权限）会将 `setuid root` 程序替换为安全属性。特权模型与超级用户模型兼容，因为特权模型也可将 `setuid` 和 `setgid` 程序识别为安全属性。
security flavor（安全特性）	请参见 flavor（特性）。
security mechanism（安全机制）	请参见 mechanism（机制）。
security policy（安全策略）	请参见 policy（策略）。
security service（安全服务）	请参见 service（服务）。
seed（种子）	用于生成随机数的数字起动器。当起动器来自随机源时，种子称为随机种子。
separation of duty（职责分离）	least privilege（最小特权）的部分概念。职责分离可阻止一个用户执行或批准完成事务的所有操作。例如，在 RBAC 中，可以将登录用户的创建与安全覆盖的指定分隔开来。一个角色创建该用户。另一个角色可以将安全属性（如权限配置文件、角色和特权）指定给现有用户。
server principal（服务器主体）	(RPCSEC_GSS API) 提供服务的主体。服务器主体以 *service*@*host* 形式的 ASCII 字符串进行存储。另请参见 client principal（客户机主体）。
server（服务器）	为网络客户机提供资源的主体。 例如，如果通过 `ssh` 远程登录到系统 `central.example.com`，则该系统便是提供 `ssh` 服务的服务器。另请参见 service principal（服务主体）。

service key（服务密钥）	由服务主体和 KDC 共享，并在系统范围之外分发的加密密钥。另请参见 key（密钥）。
service principal（服务主体）	为一项或多项服务提供 Kerberos 验证的主体。对于服务主体，主名称是服务的名称（如 `ftp`），其实例是提供服务的系统的全限定主机名。另请参见 host principal（host 主体）和 user principal（用户主体）。
service（服务）	1. 通常由多台服务器提供给网络客户机的资源。例如，如果通过 `rlogin` 远程登录到计算机 `central.example.com`，则该计算机便是提供 `rlogin` 服务的服务器。 2. 除验证之外，还提供其他保护级别的安全服务（完整性或保密性）。另请参见 integrity（完整性）和 privacy（保密性）。
session key（会话密钥）	由验证服务或票证授予服务生成的密钥。生成会话密钥的目的是在客户机与服务之间提供安全事务。会话密钥的生命周期仅限于单个登录会话的持续时间。另请参见 key（密钥）。
SHA1	Secure Hashing Algorithm（安全散列算法）。该算法可以针对长度小于 2^{64} 的任何输入进行运算，以生成消息摘要。SHA1 算法是 DSA 的输入。
single-system image（单系统映像）	单系统映像用在 Oracle Solaris 审计中来描述使用相同命名服务的一组受审计系统。这些系统将其审计记录发送给某个中心审计服务器，可在该服务器中对记录进行比较，就像这些记录来自一个系统一样。
slave KDC（从 KDC）	主 KDC 的副本，可以执行主 KDC 的大多数功能。每个领域通常都具有若干个从 KDC（但仅有一个主 KDC）。另请参见 KDC 和 master KDC（主 KDC）。
software provider（软件提供者）	在 Oracle Solaris 的加密框架功能中，是指提供加密服务的内核软件模块或 PKCS #11 库。另请参见 provider（提供者）。
stash file（存储文件）	存储文件包含 KDC 主密钥的已加密副本。当重新引导服务器以便在 KDC 启动 `kadmind` 和 `krb5kdc` 进程之前自动验证 KDC 时，将使用此主密钥。由于存储文件中包含主密钥，因此，应该保证存储文件及其任何备份的安全。如果加密受到威胁，则可以使用此密钥来访问或修改 KDC 数据库。
superuser model（超级用户模型）	计算机系统上的典型 UNIX 安全模型。在超级用户模型中，管理员对系统要么具有全部的控制权要么毫无控制权。通常，为了管理计算机，用户可成为超级用户 (`root`)，并可执行所有管理活动。
synchronous audit event（同步审计事件）	审计事件中的大多数事件属于同步审计事件。这些事件与系统中的某个进程关联。与某个进程关联的无归属事件属于同步事件，如失败的登录。
TGS	Ticket-Granting Service（票证授予服务）。负责颁发票证的那部分 KDC。

TGT	Ticket-Granting Ticket（票证授予票证）。由 KDC 颁发的票证，客户机可使用此票证来请求其他服务的票证。
ticket file（票证文件）	请参见 credential cache（凭证高速缓存）。
ticket（票证）	用于安全地将用户身份传递给服务器或服务的信息包。一个票证仅对一台客户机以及某台特定服务器上的一项特殊服务有效。票证包含服务的主体名称、用户的主体名称、用户主机的 IP 地址、时间戳以及定义此票证生命周期的值。票证是通过由客户机和服务使用的随机会话密钥创建的。一旦创建了票证，便可重复使用此票证，直到其到期为止。票证与新的验证者同时出现时，仅用于验证客户机。另请参见 authenticator（验证者）、credential（凭证）、service（服务）和 session key（会话密钥）。
trusted users（可信用户）	指的是您决定允许其在一定信任级别下执行管理任务的用户。通常，管理员先为可信用户创建登录名，并指定与此类用户的信任级别和能力匹配的管理权限。之后，这些用户便能帮助配置和维护系统。此类用户也称为特权用户。
user principal（用户主体）	属于某个特定用户的主体。用户主体的主名称是用户名，其可选实例是用于说明相应凭证预期用法的名称（例如 `jdoe` 或 `jdoe/admin`）。也称为用户实例。另请参见 service principal（服务主体）。
virtual private network, VPN（虚拟专用网络）	通过使用加密和隧道连接公共网络上的用户来提供安全通信的网络。

索引

L

M

N

O

P

Q

R

S