在 Oracle® Solaris 11.2 中确保文件的安全和确认文件完整性

文件号码 E53951
2014 年 7 月

ORACLE

目录

使用本文档

- **概述** – 介绍如何保护合法文件、查看隐藏的文件权限以及查找和防止执行未授权文件。此外还介绍了如何随时间的推移验证 Oracle Solaris 系统上文件的完整性。
- **目标读者** – 系统管理员。
- **必备知识** – 站点安全要求。

产品文档库

有关本产品的最新信息和已知问题均包含在文档库中，网址为：http://www.oracle.com/pls/topic/lookup?ctx=E56344。

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

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

♦ ♦ ♦ 第 1 章

1 控制对文件的访问

本章介绍了如何保护 Oracle Solaris 中的文件，还介绍了如何保护系统免受其权限可能会危及系统安全的文件的影响。

注 - ZFS 是缺省的 OS 文件系统。要使用访问控制列表 (Access Control List, ACL) 保护 ZFS 文件，请参见《在 Oracle Solaris 11.2 中管理 ZFS 文件系统》中的第 7 章 “使用 ACL 和属性保护 Oracle Solaris ZFS 文件”。

本章包含以下主题：

使用 UNIX 权限保护文件

可以通过 UNIX 文件权限和 ACL 保证文件安全。带 sticky 位的文件和可执行文件要求特殊的安全措施。

用于查看和保证文件安全的命令

下表介绍了用于监视以及保证文件和目录安全的命令。

表 1-1 保证文件和目录安全的命令

命令	说明	手册页
`ls`	列出目录中的文件及其有关信息。	`ls(1)`
`chown`	更改文件的所有权。	`chown(1)`
`chgrp`	更改文件的组所有权。	`chgrp(1)`

命令	说明	手册页
chmod	更改文件的权限。可以使用符号模式（使用字母和符号）或绝对模式（使用八进制数字）更改文件的权限。	chmod(1)

文件和目录的所有权

传统的 UNIX 文件权限可以为三类用户指定所有权：

- **用户** – 文件或目录的所有者，通常为创建该文件的用户。文件的所有者可以决定谁拥有读取文件、写入文件（对文件进行更改）或执行文件（如果该文件为命令）的权限。
- **组** – 一组用户的成员。
- **其他用户** – 所有其他不是文件所有者和组成员的用户。

文件所有者通常可以指定或修改文件权限。此外，root 帐户可以更改文件的所有权。要覆盖系统策略，请参见例 1-2 “允许用户更改其自身文件的所有权”。

文件可以是七种类型之一。每种类型由一个符号显示：

-（减号）	文本或程序
b	块特殊文件
c	字符特殊文件
d	目录
l	符号链接
s	套接字
D	门
P	命名管道 (FIFO)

UNIX 文件权限

下表列出并说明了可以为文件或目录的每类用户授予的权限。

表 1-2 文件和目录权限

符号	权限	对象	说明
r	读	文件	指定的用户可以打开和读取文件内容。

符号	权限	对象	说明
		目录	指定的用户可以列出目录中的文件。
w	写	文件	指定的用户可以修改文件的内容或删除该文件。
		目录	指定的用户可以在目录中添加文件或链接。这些用户也可以删除目录中的文件或链接。
x	执行	文件	指定的用户可以执行文件（如果该文件为程序或 shell 脚本）。这些用户也可以使用一个 exec(2) 系统调用来运行程序。
		目录	指定的用户可以打开或执行目录中的文件。这些用户也可以使该目录以及该目录下的目录成为当前目录。
-	拒绝	文件和目录	指定的用户无法读写或执行文件。

这些文件权限可应用于常规文件，也可应用于特殊文件（如设备、套接字和命名管道 (FIFO)）。

对于符号链接，所应用的权限为链接指向的文件权限。

通过对目录设置受限文件权限，可以保护该目录及其子目录中的文件。但是请注意，root 角色有权访问系统中的所有文件和目录。

使用 setuid、setgid 和 Sticky 位的特殊文件权限

可执行文件和公共目录可以使用三种特殊类型的权限：setuid、setgid 和 sticky 位。设置这些权限之后，运行可执行文件的任何用户都应采用该可执行文件所有者（或组）的 ID。

设置特殊权限时必须非常小心，因为特殊权限会带来安全风险。例如，通过执行将用户 ID (user ID, UID) 设置为 0（这是 root 的 UID）的程序，用户可以获得 root 的功能。此外，所有用户可以为其拥有的文件设置特殊权限，这会带来其他安全问题。

应对系统中任何未经授权使用 setuid 权限和 setgid 权限获取 root 功能的情况进行监视。可疑权限为用户而不是 root 或 bin 授予管理程序的所有权。要搜索并列出所有使用此特殊权限的文件，请参见如何查找具有特殊文件权限的文件 [20]。

setuid 权限

对可执行文件设置 setuid 权限时，将对运行该文件的进程授予基于文件所有者的访问权限。该访问权限不是基于正在运行可执行文件的用户。使用此特殊权限，用户可以访问通常只有属主才可访问的文件和目录。

例如，passwd 命令的 setuid 权限使用户可以更改口令。具有 setuid 权限的 passwd 命令类似如下：

```
-r-sr-sr-x   1 root     sys       56808 Jun 17 12:02 /usr/bin/passwd
```

此特殊权限存在安全风险。一些确定的用户甚至可以在 setuid 进程执行完毕后，找到保持由该进程授予他们的权限的方法。

注 - 在程序中使用具有保留 UID (0-100) 的 setuid 权限可能无法正确设置有效的 UID。请使用 shell 脚本或避免将保留的 UID 用于 setuid 权限。

---

setgid 权限

setgid 权限与 setuid 权限类似。可将进程的有效组 ID (group ID, GID) 更改为拥有该文件的组，并基于授予该组的权限对用户授予访问权限。/usr/bin/mail 命令具有 setgid 权限：

```
-r-x--s--x   1 root    mail     71212 Jun 17 12:01 /usr/bin/mail
```

将 setgid 权限应用于目录时，该目录中已创建的文件将属于该目录所属于的组。这些文件不属于创建进程所属于的组。在目录中拥有写和执行权限的任何用户都可以在其中创建文件。但是，文件将属于拥有该目录的组，而不是用户所属于的组。

应对系统中任何未经授权使用 setgid 权限获取 root 功能的情况进行监视。可疑权限为非常规组而不是 root 或 bin 授予对此类程序的组访问权限。要搜索并列出所有使用此权限的文件，请参见如何查找具有特殊文件权限的文件 [20]。

Sticky 位

sticky 位是保护目录中文件的权限位。如果对目录设置了 sticky 位，则只有文件所有者、目录所有者或特权用户才可以删除文件。root 用户是特权用户的一个示例。sticky 位禁止用户从公共目录（如 /tmp）中删除其他用户的文件：

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

在 TMPFS 文件系统中设置公共目录时，务必手动设置 sticky 位。有关说明，请参见例 1-5 “在绝对模式下设置特殊文件权限”。

缺省 umask 值

创建文件或目录时，将使用一组缺省权限进行创建。系统缺省值为空。文本文件具有 666 权限，该权限对所有用户授予读写权限。目录和可执行文件具有 777 权限，该权限对所有用户授予读写和执行权限。通常，用户会覆盖其 shell 初始化文件（如 .bashrc 和 .kshrc.user）中的系统缺省值。管理员还可以设置 /etc/profile 文件中的缺省值。

由 umask 命令指定的值将从缺省值中减去。此进程的作用是以 chmod 命令授予权限的相同方式拒绝这些权限。例如，chmod 022 命令对组和其他用户授予写权限。umask 022 命令拒绝组和其他用户的写权限。

下表显示了一些典型 umask 值及其对可执行文件的影响。

表 1-3 不同安全级别的 umask 设置

安全级别	umask 设置	禁用的权限
许可 (744)	022	w（组和其他用户）
中等 (751)	026	w（组），rw（其他用户）
严密 (740)	027	w（组），rwx（其他用户）
严重 (700)	077	rwx（组和其他用户）

有关设置 umask 值的更多信息，请参见 umask(1) 手册页。

文件权限模式

使用 chmod 命令，可以更改文件的权限。您必须是 root 用户或是文件或目录的所有者，才能更改其权限。

可以使用 chmod 命令按照以下两种模式之一设置权限：

- **绝对模式** – 使用数字表示文件权限。使用绝对模式更改权限时，由八进制模式数字表示每个三元字节权限。绝对模式是设置权限的最常用方法。
- **符号模式** – 使用字母和符号的组合来添加或删除权限。

下表列出了在绝对模式下设置文件权限的八进制值。可按顺序以三个一组的形式，使用这些数字来设置所有者、组和其他用户的权限。例如，值 644 为所有者设置读写权限，为组和其他用户设置只读权限。

表 1-4 在绝对模式下设置文件权限

八进制值	设置文件权限	权限说明
0	---	无权限
1	--x	仅执行权限
2	-w-	只写权限
3	-wx	写和执行权限
4	r--	只读权限
5	r-x	读和执行权限
6	rw-	读写权限

八进制值	设置文件权限	权限说明
7	rwx	读写和执行权限

下表列出了在符号模式下设置文件权限所用的符号。符号可以指定要设置或更改其权限的用户、要执行的操作，以及要指定或更改的权限。

表 1-5 在符号模式下设置文件权限

符号	功能	说明
u	*who*	用户（所有者）
g	*who*	组
o	*who*	其他用户
a	*who*	全部
=	*operator*	赋值
+	*operator*	新增
-	*operator*	删除
r	*permissions*	读
w	*permissions*	写
x	*permissions*	执行
l	*permissions*	强制锁定，setgid 位打开，组执行位关闭
s	*permissions*	setuid 或 setgid 位打开
t	*permissions*	Sticky 位打开，对于其他用户，执行位打开

功能列中的名称 *who operator permissions* 指定用于更改文件或目录的权限的符号。

who 指定要更改其权限的用户。

operator 指定要执行的操作。

permissions 指定要更改的权限。

可以在绝对模式或符号模式下设置文件的特殊权限。但是，必须使用符号模式设置或删除目录的 setuid 权限。在绝对模式下，通过在权限三元字节的左侧添加新的八进制值，可设置特殊权限。请参见例 1-5 “在绝对模式下设置特殊文件权限”。下表列出了用于对文件设置特殊权限的八进制值。

表 1-6 在绝对模式下设置特殊文件权限

八进制值	特殊文件权限
1	Sticky 位
2	setgid
4	setuid

使用访问控制列表保护 UFS 文件

传统的 UNIX 文件保护可为以下三类用户提供读写和执行权限：文件所有者、文件组和其他用户。在 UFS 文件系统中，访问控制列表 (access control list, ACL) 通过允许您执行以下操作来提供更好的文件安全性：

- 为文件所有者、组、其他用户、特定用户和特定组定义文件权限
- 为上面的每一种类别定义缺省权限

注 - 对于 ZFS 文件系统中的 ACL 以及 NFSv4 文件的 ACL，请参见《在 Oracle Solaris 11.2 中管理 ZFS 文件系统 》中的第 7 章 “使用 ACL 和属性保护 Oracle Solaris ZFS 文件”。

例如，如果想要组中的每个用户都能够读取某文件，则只需要授予该组对该文件的读取权限即可。不过，如果您希望组中只有一个用户能够写入到该文件，可以使用 ACL。

有关 UFS 文件系统上的 ACL 的更多信息，请参见 Oracle Solaris 10 发行版的《系统管理指南：安全服务》。

防止可执行文件危及安全

程序读写栈中的数据。通常，程序从内存中专为代码指定的只读部分开始执行。一些可导致栈中的缓冲区溢出的攻击尝试向栈中插入新代码，从而致使程序执行该代码。删除栈内存的执行权限可使这些攻击无法成功。换句话说，大多数程序可以在不使用可执行栈的情况下正常工作。

64 位进程的栈通常不可执行。缺省情况下，32 位 SPARC 进程具有可执行栈。利用 `noexec_user_stack` 变量，可以指定 32 位进程的栈是否可执行。

设置此变量后，将向尝试执行其栈中代码的程序发送一个 `SIGSEGV` 信号。此信号通常将导致程序终止，同时进行核心转储。这样的程序还将生成一条警告消息，该消息中包括违例程序的名称、进程 ID 和运行该程序的用户的实际 UID。例如：

```
a.out[347] attempt to execute code on stack by uid 555
```

将 `syslog kern` 功能设置为 `notice` 级别时，该消息由 `syslog` 守护进程记录。缺省情况下，将在 `syslog.conf` 文件中设置此日志记录，这意味着，消息将发送到控制台和 `/var/adm/messages` 文件。有关更多信息，请参见 `syslogd(1M)` 和 `syslog.conf(4)` 手册页。

在查看可能的安全问题时，`syslog` 消息非常有用。通过设置 `noexec_user_stack` 变量，该消息还将确定依赖于可执行栈（已被禁止，无法正确执行操作）的有效程序。如果不

想记录任何消息，则可以在 /etc/system 文件中将日志变量 noexec_user_stack_log 设置为零。即使未记录消息，SIGSEGV 信号仍可能会导致执行程序终止，同时进行核心转储。

程序可以显式标记或阻止栈执行。程序中的 mprotect（） 函数将栈显式标记为可执行。有关更多信息，请参见 mprotect(2) 手册页。无论系统级设置是什么，通过 -M /usr/lib/ld/map.noexstk 编译的程序都将使栈不可执行。

保护文件

以下过程使用 UNIX 权限保护文件、查找有安全风险的文件，并保护系统免受这些文件的危害。

使用 UNIX 权限保护文件

以下任务列表列出了用于列出文件权限、更改文件权限，以及使用特殊文件权限保护文件的过程。

任务	参考
显示文件信息。	如何显示文件信息 [14]
更改本地文件所有权。	如何更改文件的所有者 [15] 如何更改文件的组所有权 [16]
更改本地文件权限。	如何在符号模式下更改文件权限 [17] 如何在绝对模式下更改文件权限 [18] 如何在绝对模式下更改特殊文件权限 [19]

▼ 如何显示文件信息

使用 ls 命令显示有关目录中所有文件的信息。

- **键入以下命令以显示当前目录中所有文件的长列表。**

```
% ls -la
```

-l 显示包括用户所有权、组所有权和文件权限的长格式。

-a 显示所有文件，包括以点 (.) 开头的隐藏文件。

有关 `ls` 命令的所有选项，请参见 `ls(1)` 手册页。

例 1-1 显示文件信息

在以下示例中，显示了 `/sbin` 目录中部分文件的列表。

```
% cd /sbin
% ls -l
total 4960
-r-xr-xr-x   1 root   bin      12756 Dec 19  2013 6to4relay
lrwxrwxrwx   1 root   root        10 Dec 19  2013 accept -> cupsaccept
-r-xr-xr-x   1 root   bin      38420 Dec 19  2013 acctadm
-r-xr-xr-x   2 root   sys      70512 Dec 19  2013 add_drv
-r-xr-xr-x   1 root   bin       3126 Dec 19  2013 addgnupghome
drwxr-xr-x   2 root   bin         37 Dec 19  2013 amd64
-r-xr-xr-x   1 root   bin       2264 Dec 19  2013 applygnupgdefaults
-r-xr-xr-x   1 root   bin        153 Dec 19  2013 archiveadm
-r-xr-xr-x   1 root   bin      12644 Dec 19  2013 arp
.
.
.
```

每一行按以下顺序显示了有关文件的信息：

- 文件类型 – 例如，`d`。有关文件类型的列表，请参见“文件和目录的所有权” [8]。
- 权限 – 例如，`r-xr-xr-x`。有关说明，请参见“文件和目录的所有权” [8]。
- 硬链接数 – 例如，`2`。
- 文件的所有者 – 例如，`root`。
- 文件的组 – 例如，`bin`。
- 文件的大小（以字节为单位） – 例如，`12644`。
- 文件创建日期或文件上次更改日期 – 例如，`Dec 19 2013`。
- 文件的名称 – 例如，`arp`。

▼ 如何更改文件的所有者

开始之前 如果您不是文件或目录的所有者，则必须指定有 Object Access Management（对象访问管理）权限配置文件。要更改作为 public object（公共对象）的文件，您必须成为 `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. 显示本地文件的权限。

```
% ls -l example-file
-rw-r--r--   1 janedoe   staff   112640 May 24 10:49 example-file
```

2. 更改文件的所有者。

```
# chown stacey example-file
```

3. 检验文件的所有者是否已更改。

```
# ls -l example-file
-rw-r--r--   1 stacey   staff   112640 May 26 08:50 example-file
```

要更改 NFS 挂载的文件的权限，请参见《在 Oracle Solaris 11.2 中管理网络文件系统 》中的第 5 章 “用于管理网络文件系统的命令”。

例 1-2 允许用户更改其自身文件的所有权

安全注意事项 – 您应该有合理理由将 `rstchown` 变量的设置更改为零。缺省设置阻止用户列出属于其他用户的文件以便规避空间配额。

在此示例中，在 `/etc/system` 文件中将 `rstchown` 变量的值设置为零。通过此设置，文件所有者可以使用 `chown` 命令将文件的所有权更改为另一用户。通过此设置，文件所有者还可以使用 `chgrp` 命令将文件的组所有权设置为非其所在的组。重新引导系统后，更改将生效。

```
set rstchown = 0
```

有关更多信息，请参见 `chown(1)` 和 `chgrp(1)` 手册页。

▼ 如何更改文件的组所有权

开始之前 如果您不是文件或目录的所有者，则必须指定有 Object Access Management（对象访问管理）权限配置文件。要更改作为 public object（公共对象）的文件，您必须成为 `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. 更改文件的组所有权。

```
% chgrp scifi example-file
```

有关设置组的信息，请参见《在 Oracle Solaris 11.2 中管理用户帐户和用户环境 》中的第 1 章 “关于用户帐户和用户环境”。

2. 检验文件的组所有权是否已更改。

```
% ls -l example-file
-rw-r--r--   1 stacey   scifi   112640 June 20 08:55  example-file
```

另请参见例 1-2 “允许用户更改其自身文件的所有权”。

▼ 如何在符号模式下更改文件权限

在以下过程中，用户更改其所拥有的文件的权限。

1. 在符号模式下更改权限。

```
% chmod who operator permissions filename
```

who	指定要更改其权限的用户。
operator	指定要执行的操作。
permissions	指定要更改的权限。有关有效符号的列表，请参见表 1-5 “在符号模式下设置文件权限”。
filename	指定文件或目录。

2. 检验文件的权限是否已更改。

```
% ls -l filename
```

注 - 如果您不是文件或目录的所有者，则必须指定有 Object Access Management（对象访问管理）权限配置文件。要更改作为 public object（公共对象）的文件，您必须成为 `root` 角色。

例 1-3 在符号模式下更改权限

在以下示例中，所有者删除了其他用户的读取权限。

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

在以下示例中，所有者为用户、组和其他用户添加了读取和执行权限。

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

在以下示例中，所有者为组成员添加了读取、写入和执行权限。

```
% chmod g=rwx example-file3
```

▼ 如何在绝对模式下更改文件权限

在以下过程中，用户更改其所拥有的文件的权限。

1. **在绝对模式下更改权限。**

```
% chmod nnn filename
```

nnn	按照该顺序指定将表示文件所有者、文件组和其他用户的权限的八进制值。有关有效八进制值的列表，请参见表 1-4 “在绝对模式下设置文件权限”。
filename	指定文件或目录。

注 - 如果使用 `chmod` 命令更改具有现有 ACL 条目的对象的文件或目录权限，则 ACL 条目也可能会发生更改。具体更改取决于 `chmod` 权限操作更改以及文件系统的 `aclmode` 和 `aclinherit` 属性值。

有关更多信息，请参见《在 Oracle Solaris 11.2 中管理 ZFS 文件系统 》中的第 7 章 “使用 ACL 和属性保护 Oracle Solaris ZFS 文件”。

2. **检验文件的权限是否已更改。**

```
% ls -l filename
```

注 - 如果您不是文件或目录的所有者，则必须指定有 Object Access Management（对象访问管理）权限配置文件。要更改作为 public object（公共对象）的文件，您必须成为 `root` 角色。

例 1-4 在绝对模式下更改权限

在以下示例中，管理员将对公众公开的目录的权限从 `744`（读、写、执行；只读；只读）更改为 `755`（读、写、执行；读和执行；读和执行）。

```
# ls -ld public_dir
drwxr--r--  1 jdoe   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 jdoe   staff    6023 Aug  5 12:06 public_dir
```

在以下示例中，文件所有者将可执行 shell 脚本的权限从读取和写入更改为读取、写入和执行。

```
% ls -l my_script
-rw------- 1 jdoe   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
```

```
% ls -l my_script
-rwx------ 1 jdoe   staff    6023 Aug  5 12:06 my_script
```

▼ 如何在绝对模式下更改特殊文件权限

开始之前 如果您不是文件或目录的所有者，则必须指定有 Object Access Management（对象访问管理）权限配置文件。要更改作为 public object（公共对象）的文件，您必须成为 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. **在绝对模式下更改特殊权限。**

   ```
   % chmod nnnn filename
   ```

   nnnn 指定用于更改文件或目录的权限的八进制值。最左侧的八进制值设置文件的特殊权限。有关特殊权限的有效八进制值的列表，请参见表 1-6 “在绝对模式下设置特殊文件权限”。

   filename 指定文件或目录。

   注 - 使用 chmod 命令更改具有 ACL 项的文件的文件组权限时，文件组权限和 ACL 掩码都将更改为新权限。请注意，新 ACL 掩码权限可以更改在文件中具有 ACL 项的其他用户和组的权限。使用 getfacl 命令以确保为所有 ACL 项都设置了适当的权限。有关更多信息，请参见 getfacl(1) 手册页。

   ---

2. **检验文件的权限是否已更改。**

   ```
   % ls -l filename
   ```

例 1-5 在绝对模式下设置特殊文件权限

在以下示例中，管理员设置了对 dbprog 文件的 setuid 权限。

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

在以下示例中，管理员设置了对 dbprog2 文件的 setgid 权限。

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

在以下示例中，管理员设置了对 public_dir 目录的 sticky 位。

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

防止程序受到安全风险

以下任务列表列出了有关查找系统中的危险可执行程序，以及禁止程序利用可执行栈的过程。

任务	说明	参考
查找具有特殊权限的文件。	查找设置了 setuid 位，但不归 root 用户拥有的文件。	如何查找具有特殊文件权限的文件 [20]
防止可执行栈溢出。	防止程序利用可执行栈。	如何禁止程序使用可执行栈 [21]
防止记录可执行栈消息。	停止记录可执行栈消息。	例 1-7 “禁止记录可执行栈消息”

▼ 如何查找具有特殊文件权限的文件

此过程查找可能未经授权在程序中使用 setuid 和 setgid 权限的情况。可疑可执行文件为用户而不是 root 或 bin 授予所有权。

开始之前 您必须成为 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. 使用 **find** 命令查找具有 **setuid** 权限的文件。

   ```
   # find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/filename
   ```

find *directory*	检查以指定的 *directory*（可以是根目录 (/)、/usr、/opt 等）开头的所有挂载路径。
-user root	仅显示由 root 拥有的文件。
-perm -4000	仅显示权限设置为 4000 的文件。
-exec ls -ldb	以 ls -ldb 格式显示 find 命令的输出。请参见 ls(1) 手册页。
/tmp/*filename*	包含 find 命令的结果的文件。

   有关更多信息，请参见find(1)。

2. 在 **/tmp/***filename* 中显示结果。

```
# more /tmp/filename
```

有关背景信息，请参见“setuid 权限” [9]。

例 1-6 查找具有 setuid 权限的文件

以下示例的输出显示，名为 rar 的组中的用户创建了一份 /usr/bin/rlogin 的个人副本，并将权限设置为 root 的 setuid。因此，/usr/rar/bin/rlogin 程序将使用 root 权限运行。

在调查了 /usr/rar 目录并删除了 /usr/rar/bin/rlogin 命令之后，管理员将来自 find 命令的输出归档。

```
# find /usr -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/tmp/ckprm
-rwsr-xr-x  1 root   sys   28000  Jul 14 14:14  /usr/bin/atq
-rwsr-xr-x  1 root   sys   32364  Jul 14 14:14  /usr/bin/atrm
-r-sr-xr-x  1 root   sys   41432  Jul 14 14:14  /usr/bin/chkey
-rwsr-xr-x  1 root   bin   82804  Jul 14 14:14  /usr/bin/cdrw
-r-sr-xr-x  1 root   bin    8008  Jul 14 14:14  /usr/bin/mailq
-r-sr-sr-x  1 root   sys   45348  Jul 14 14:14  /usr/bin/passwd
-rwsr-xr-x  1 root   bin   37724  Jul 14 14:14  /usr/bin/pfedit
-r-sr-xr-x  1 root   bin   51440  Jul 14 14:14  /usr/bin/rcp
---s--x---  1 root   rar   41592  Jul 24 16:14  /usr/rar/bin/rlogin
-r-s--x--x  1 root   bin  166908  Jul 14 14:14  /usr/bin/sudo
-r-sr-xr-x  4 root   bin   24024  Jul 14 14:14  /usr/bin/uptime
-r-sr-xr-x  1 root   bin   79488  Jul 14 14:14  /usr/bin/xlock
# mv /var/tmp/ckprm /var/share/sysreports/ckprm
```

▼ 如何禁止程序使用可执行栈

有关 32 位可执行栈的安全风险的说明，请参见“防止可执行文件危及安全” [13]。

开始之前 您必须成为 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 **/etc/system** 文件，添加以下行：

```
# pfedit /etc/system
...
set noexec_user_stack=1
set noexec_user_stack_log=1
```

2. 重新引导系统。

```
# reboot
```

例 1-7 禁止记录可执行栈消息

在此示例中，管理员禁用了可执行堆栈消息的日志记录，然后重新引导了系统。

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# reboot
```

另请参见 有关更多信息，请参阅以下链接：

- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_overview
- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_continued
- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_concluded

♦ ♦ ♦ 第 2 章

使用 BART 检验文件完整性

本章介绍了文件完整性检查工具 BART。BART 是一个命令行工具，您可以使用它验证系统上的文件在一段时间内的完整性。本章包含以下主题：

关于 BART

BART 是一个文件完整性扫描和报告工具，它使用加密强度校验和及文件系统元数据来确定变化。BART 能够识别损坏的或异常的文件，从而可帮助您检测安全违规或解决性能问题。使用 BART 可轻松且可靠地报告所部署的系统上安装的文件中的差异，从而减少管理系统网络的成本。

使用 BART，可以根据已知的基准线确定系统上已发生的文件级别更改。可以使用 BART 从完全安装并配置的系统创建基准线或控制清单。以后，可以将此基准线与系统快照进行比较，以生成一个报告，列出该系统安装之后发生的文件级别的更改。

BART 功能

BART 使用强大而灵活的简单语法。使用此工具，可以跟踪在一段时间内给定系统上的文件变化。您还可以跟踪相似系统间的文件差异。此类比较可帮助您找出损坏或异常的文件，或其软件过期的系统。

BART 的其他优点和用途包括：

- 您可以指定要监视哪些文件。例如，您可以监视本地定制设置，这有助于您轻松高效地重新配置软件。
- 您可以解决系统性能问题。

BART 组件

BART 创建两个主要文件，一个清单和一个比较文件（或称为报告）。使用可选的规则文件，您可以定制清单和报告。

BART 清单

清单是系统在特定时间的文件级快照。清单包含有关文件属性的信息，其中可能包括一些唯一标识信息，如校验和。`bart create` 命令的选项可针对特定的文件和目录。规则文件可提供更细粒度的过滤，如“BART 规则文件” [25]中所述。

注 - 缺省情况下，BART 会对根 (/) 目录下的所有 ZFS 文件系统进行编目。还会对其他文件系统类型（例如 NFS 或 TMPFS 文件系统，以及已挂载的 CD-ROM）进行编目。

可以在初始安装 Oracle Solaris 之后立即创建系统的清单。您还可以在配置系统后创建一个清单来满足您的站点的安全策略。这种类型的控制清单为您提供了一个基准线，供以后进行比较时使用。

基准线清单可用于跟踪在一段时间内同一系统上的文件完整性。它还可以用作与其他系统进行比较的基础。例如，您可以为网络上的其他系统创建快照，然后将这些清单与基准线清单进行比较。报告的文件差异指明了将其他系统与基准线系统进行同步所需执行的操作。

有关清单的格式，请参见“BART 清单文件格式” [35]。要创建清单，请使用 `bart create` 命令，如如何创建控制清单 [26]中所述。

BART 报告

BART 报告列出两个清单间的每文件差异。差异是指上述两个清单中所列出的某个特定文件的任何属性的变化。文件项的添加或删除也被视为差异。

为进行有效的比较，两个清单必须针对同一文件系统。创建和比较清单时还必须使用相同的选项和规则文件。

有关报告的格式，请参见“BART 报告” [37]。要创建报告，请使用 `bart compare` 命令，如如何比较同一个系统在一段时间内的清单 [29]中所述。

BART 规则文件

BART 规则文件是您创建的一个文件，用来过滤或咬定特定的文件和文件属性以将其包含在内或排除在外。然后，您在创建 BART 清单和报告时可以使用该文件。在比较清单时，使用规则文件有助于标记清单间的差异。

注 - 如果使用某个规则文件创建了一个清单，则在创建比较清单必须使用同一规则文件。在对清单进行比较时也必须使用该规则文件。否则，报告会列出许多无效的差异。

要使用规则文件来监视系统上的特定文件和文件属性，需要进行规划。在创建规则文件之前，请先确定系统上要监视的文件和文件属性。

规则文件中也可能包含由于用户错误而导致的语法错误和其他不明确信息。如果规则文件有错误，也会报告这些错误。

有关规则文件的格式，请参见“BART 规则文件格式” [36]和 `bart_rules(4)` 手册页。要创建规则文件，请参见如何通过使用规则文件定制 BART 报告 [33]。

关于使用 BART

`bart` 命令用来创建和比较清单。任何用户都可以运行此命令。不过，用户只能对他们有权访问的文件进行编目和监视。因此，用户和大多数角色都可以有效地对其起始目录中的文件进行编目，但 `root` 帐户可以对所有文件进行编目，包括系统文件。

BART 安全注意事项

任何人都可以读取 BART 清单和报告。如果 BART 输出包含敏感信息，请采取适当的措施来保护输出。例如，使用可生成具有限制性权限的输出文件的选项，或将输出文件置于受保护的目录中。

使用 BART

任务	说明	参考
创建 BART 清单。	生成有关系统上安装的每个文件的信息列表。	如何创建控制清单 [26]

任务	说明	参考
创建定制 BART 清单。	生成有关系统上安装的特定文件的信息列表。	如何定制清单 [28]
比较 BART 清单。	生成比较一个系统在一段时间内发生的更改的报告。 或者，生成将一个或几个系统与控制系统比较的报告。	如何比较同一个系统在一段时间内的清单 [29] 如何比较不同系统的清单 [31]
（可选）定制 BART 报告。	通过以下方法之一生成定制 BART 报告： ■ 指定属性 ■ 使用规则文件	如何通过指定文件属性定制 BART 报告 [33] 如何通过使用规则文件定制 BART 报告 [33]

▼ 如何创建控制清单

此过程介绍了如何创建用于比较的基准线或控制清单。基于一个中心映像安装多个系统时，可使用此类清单。或者，在需要确认安装完全一致时使用此类清单运行比较。有关控制清单的更多信息，请参见“BART 清单” [24]。要了解格式约定，请参见例 2-1 “BART 清单格式说明”。

注 - 不要尝试对联网的文件系统进行编目。使用 BART 监视联网的文件系统会占用大量的资源，但生成的清单价值很小。

开始之前 您必须成为 `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. **在根据您的站点的安全要求对 Oracle Solaris 系统进行定制后，请创建一个控制清单并将输出重定向到一个文件。**

   ```
   # bart create options > control-manifest
   ```

   `-R` 指定清单的根目录。由规则指定的所有路径都会被解释为此目录的相对路径。清单中报告的所有路径均为此目录的相对路径。

   `-I` 接受要在目录中列出的单个文件的列表（无论是从命令行执行此选项，还是从标准输入中读取此选项）。

   `-r` 此清单的规则文件的名称。`-` 参数从标准输入中读取规则文件。

   `-n` 禁用文件列表中所有常规文件的内容签名。此选项可用于改善性能。或者，可以在需要更改文件列表的内容时使用此选项，这与系统日志文件的情况类似。

2. **检查清单的内容。**

   有关格式的说明，请参见例 2-1 “BART 清单格式说明”。

3. （可选） 保护清单。

   用来保护系统清单的一种方法是将其置于只有 root 帐户可以访问的目录中。

```
# mkdir /var/adm/log/bartlogs
# chmod 700 /var/adm/log/bartlogs
# mv control-manifest /var/adm/log/bartlogs
```

   为清单选择一个有意义的名称。例如，使用创建清单的系统名称及日期，如 mach1-120313 中所示。

例 2-1 BART 清单格式说明

在本示例中，样例输出后面是对清单格式的说明。

```
# bart create
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (22:22:27)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 1024 40755 user::rwx,group::r-x,mask:r-x,other:r-x
3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090 0 0
.
.
.
/zone D 512 40755 user::rwx group::r-x,mask:r-x,other:r-x 3f81e892
154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334 0 0
.
.
.
```

每个清单都包括一个文件头和多个文件项。每个文件项均单独占据一行，具体取决于文件类型。例如，对于以上输出中的每个文件项，类型 F 指定文件，而类型 D 指定目录。同时列出的还有关于大小、内容、用户 ID、组 ID 和权限的信息。输出中的文件项按文件名的编码版本顺序排列，以便正确地处理特殊字符。所有项均按文件名以升序排列。所有非标准文件名（例如那些包含嵌入的换行符或制表符的文件名）都会在排序之前将非标准字符用引号括起来。

以 ! 开头的行提供有关清单的元数据。清单版本行指明清单规范版本。散列行指明所使用的散列机制。有关用作校验和的 SHA256 散列的更多信息，请参见 sha2(3EXT) 手册页。

日期行以日期格式显示清单的创建日期。请参见 date(1) 手册页。清单比较工具会忽略某些行。被忽略的行包括元数据、空白行、仅包含空格的行，以及以 # 开头的注释。

▼ 如何定制清单

可以通过以下方法之一定制清单：

- 指定子树

  指定单个子树是监视选定的重要文件（如 /etc 目录中的所有文件）的变化的高效方法。
- 指定文件名

  指定文件名是监视特别敏感的文件（如配置和运行数据库应用程序的文件）的高效方法。
- 使用规则文件

  通过使用规则文件创建和比较清单，您可以灵活地为多个文件或子树指定多个属性。从命令行中，您可以指定一个应用于清单或报告中的所有文件的一个全局属性定义。通过规则文件，您可以指定不全局应用的属性。

开始之前 您必须成为 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. **确定要在目录中列出和监视的文件。**
2. **使用下列选项之一创建一个定制清单：**

   - 指定子树：

     ```
     # bart create -R subtree
     ```
   - 指定一个或多个文件名：

     ```
     # bart create -I filename...
     ```

     例如：

     ```
     # bart create -I /etc/system /etc/passwd /etc/shadow
     ```
   - 使用规则文件：

     ```
     # bart create -r rules-file
     ```
3. **检查清单的内容。**
4. **（可选） 将清单保存在受保护的目录中供将来使用。**

   例如，请参见如何创建控制清单 [26]中的步骤 3。

提示 - 如果您使用了规则文件，请将规则文件与清单保存在一起。为进行有效的比较，在运行比较时必须使用该规则文件。

▼ 如何比较同一个系统在一段时间内的清单

通过比较清单随时间的变化，您可以找出损坏或异常的文件、检测安全违规或排除系统的性能问题。

开始之前 您必须成为 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. **创建一个要在系统上监视的文件的控制清单。**

   ```
   # bart create -R /etc > control-manifest
   ```

2. **（可选） 将清单保存在受保护的目录中供将来使用。**

   例如，请参见如何创建控制清单 [26]中的步骤 3。

3. **在以后的某个时间，准备一个与控制清单完全相同的清单。**

   ```
   # bart create -R /etc > test-manifest
   ```

4. **对第二个清单进行保护。**

   ```
   # mv test-manifest /var/adm/log/bartlogs
   ```

5. **比较这两个清单。**

   在对清单进行比较时请使用创建它们时所用的相同命令行选项和规则文件。

   ```
   # bart compare options control-manifest test-manifest > bart-report
   ```

6. **检查 BART 报告中的异常情况。**

例 2-2 跟踪同一系统中文件在一段时间内的变化

本示例说明了如何跟踪 /etc 目录在一段时间内的变化。通过此类比较，您可以查明系统上的重要文件是否受到安全威胁。

- 创建控制清单。

```
# cd /var/adm/logs/manifests
# bart create -R /etc > system1.control.090713
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (11:11:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
```

```
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
/.login F 1429 100644 owner@:read_data/write_data/append_data/read_xattr/write_x
attr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchronize
:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow,ev
eryone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4bf9d6d7 0 3 ff6251a473a53de68ce8b4036d0f569838cff107caf1dd9fd04701c48f09242e
.
.
.
```

- 之后，使用相同的命令行选项创建一个测试清单。

```
# bart create -R /etc > system1.test.101013
Version 1.1
! HASH SHA256
! Monday, October 10, 2013 (10:10:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
.
.
.
```

- 比较这些清单。

```
# bart compare system1.control.090713 system1.test.101013
/security/audit_class
mtime  4f272f59
```

此输出表明自控制清单创建以来，`audit_class` 文件的修改时间已发生变化。如果出现意外变化，您可以进一步进行调查。

▼ 如何比较不同系统的清单

通过比较来自不同系统的清单，您可以确定系统是否是以完全相同的方式安装的，或者系统是否已同步升级。例如，如果您为系统定制了特定的安全目标，该比较将查找代表您的安全目标的清单与来自其他系统的清单之间的任何差异。

开始之前 您必须成为 `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. **创建控制清单。**

   ```
   # bart create options > control-manifest
   ```

   有关选项，请参见 `bart`(1M) 手册页。

2. **（可选） 将清单保存在受保护的目录中供将来使用。**
   例如，请参见如何创建控制清单 [26]中的步骤 3。

3. **在测试系统上，使用相同的 `bart` 选项创建一个清单。**

   ```
   # bart create options > test1-manifest
   ```

4. **（可选） 将清单保存在受保护的目录中供将来使用。**

5. **要进行比较，请将清单复制到一个中央位置。**
   例如：

   ```
   # cp control-manifest /net/test-server/var/adm/logs/bartlogs
   ```

   如果测试系统不是 NFS 挂载的系统，请使用 `sftp` 或其他可靠方法将清单复制到一个中央位置。

6. **比较清单并将输出重定向到一个文件。**

   ```
   # bart compare control-manifest test1-manifest > test1.report
   ```

7. **检查 BART 报告中的异常情况。**

例 2-3 识别 `/usr/bin` 目录中的可疑文件

本示例比较两个系统上 `/usr/bin` 目录中的内容。

- 创建控制清单。

  ```
  # bart create -R /usr/bin > control-manifest.090713
  ! Version 1.1
  ! HASH SHA256
  ! Saturday, September 07, 2013 (11:11:17)
  ```

```
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
/7z F 509220 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribu
tes/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:rea
d_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:r
ead_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4dadc48a 0
2 3ecd418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
...
```

- 为需要与控制系统进行比较的每个系统创建一个相同的清单。

```
# bart create -R /usr/bin > system2-manifest.101013
! Version 1.1
! HASH SHA256
! Monday, October 10, 2013 (10:10:22)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
...
```

- 将清单复制到同一个位置。

```
# cp control-manifest.090713 /net/system2.central/bart/manifests
```

- 比较这些清单。

```
# bart compare control-manifest.090713 system2.test.101013 > system2.report
/su:
```

```
gid  control:3  test:1
/ypcat:
mtime  control:3fd72511  test:3fd9eb23
```

此输出表明 /usr/bin 目录中 su 文件的组 ID 与控制系统中的相应组 ID 不同。此信息可能表明测试系统上安装了不同的软件版本。因为 GID 已更改，所以很有可能是有人篡改了此文件。

▼ 如何通过指定文件属性定制 BART 报告

此过程对于从现有清单的输出中过滤特定文件属性非常有用。

开始之前 您必须成为 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. 确定要检查的文件属性。
2. 比较包含要检查的文件属性的两个清单。
   例如：

   ```
   # bart compare -i lnmtime,mtime control-manifest.121513 \
     test-manifest.010514 > bart.report.010514
   ```

   在命令行语法中使用逗号分隔每个文件属性。
3. 检查 BART 报告中的异常情况。

▼ 如何通过使用规则文件定制 BART 报告

通过使用规则文件，您可以为您关注的特定文件和文件属性定制 BART 清单。通过使用缺省 BART 清单上的不同规则文件，您可以为相同的清单运行不同的比较。

开始之前 您必须成为 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. 确定需要监视的文件和文件属性。
2. 创建包含合适指令的一个规则文件。
3. 使用所创建的规则文件创建控制清单。

   ```
   # bart create -r myrules1-file > control-manifest
   ```
4. （可选） 将清单保存在受保护的目录中供将来使用。

例如，请参见如何创建控制清单 [26]中的步骤 3。

5. 在其他系统上或在以后的某个时间创建一个完全相同的清单，或者以后的某个时间在其他系统上创建一个完全相同的清单。

```
# bart create -r myrules1-file > test-manifest
```

6. 使用同一个规则文件比较清单。

```
# bart compare -r myrules1-file control-manifest test-manifest > bart.report
```

7. 检查 BART 报告中的异常情况。

例 2-4 使用规则文件定制 BART 清单和比较报告

以下规则文件指示 `bart create` 命令列出 `/usr/bin` 目录中的文件的所有属性。此外，该规则文件还指示 `bart compare` 命令仅报告同一个目录的大小和内容的更改。

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

IGNORE all
CHECK size contents
/usr/bin
```

- 使用所创建的规则文件创建控制清单。

```
# bart create -r usrbinrules.txt > usr_bin.control-manifest.121013
```

- 在需要监视 `/usr/bin` 目录的变化时请准备一个完全相同的清单。

```
# bart create -r usrbinrules.txt > usr_bin.test-manifest.121113
```

- 使用同一个规则文件比较清单。

```
# bart compare -r usrbinrules.txt usr_bin.control-manifest.121013 \
usr_bin.test-manifest.121113
```

- 检查 `bart compare` 命令的输出。

```
 /usr/bin/gunzip:  add
/usr/bin/ypcat:
delete
```

之前的输出表明删除了 `/usr/bin/ypcat` 文件，添加了 `/usr/bin/gunzip` 文件。

BART 清单、规则文件和报告

本节介绍 BART 使用和创建的文件的格式。

BART 清单文件格式

每个清单文件项均单独占据一行，具体取决于文件类型。每个项都以 *fname*（即文件名）开头。为了避免因文件名中嵌入特殊字符导致的解析问题，已对文件名进行了编码。有关更多信息，请参见“BART 规则文件格式” [36]。

后续字段表示以下文件属性：

type
文件类型，可能值为：
- B，表示块设备节点
- C，表示字符设备节点
- D，表示目录
- F，表示文件
- L，表示符号链接
- P，表示管道
- S，表示套接字

size
以字节为单位的文件大小。

mode
表示文件权限的八进制数。

acl
文件的 ACL 属性。对于具有 ACL 属性的文件，其中包含 `acltotext（）` 的输出。

uid
此项的所有者的数字用户 ID。

gid
此项的所有者的数字组 ID。

dirmtime
目录的上次修改时间，以秒为单位，从 1970 年 1 月 1 日 00:00:00 UTC（国际协调时间）开始计算。

lnmtime
链接的上次修改时间，以秒为单位，从 1970 年 1 月 1 日 00:00:00 UTC 开始计算。

mtime
文件的上次修改时间，以秒为单位，从 1970 年 1 月 1 日 00:00:00 UTC 开始计算。

contents
文件的校验和值。此属性仅为常规文件指定。如果您关闭了上下文检查，或者无法计算校验和，则此字段的值为 `-`。

dest
符号链接的目标。

devnode
设备节点的值。此属性仅用于字符设备文件和块设备文件。

有关更多信息，请参见 bart_manifest(4) 手册页。

BART 规则文件格式

规则文件是文本文件，其中包含的行指定了哪些文件要包括在清单中以及哪些文件属性要包括在清单或报告中。该工具将忽略以 # 开头的行、空白行以及包含空格的行。

输入文件包含三种类型的指令：

- 子树指令，带有可选的模式匹配修饰符
- CHECK 指令
- IGNORE 指令

例 2-5 规则文件格式

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

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

注 - 所有指令将按顺序读取。后面的指令会覆盖前面的指令。

子树指令必须以绝对路径名开头，后面跟有零个或多个模式匹配语句。

BART 规则文件属性

CHECK 和 IGNORE 语句定义要跟踪或忽略的文件属性。每个清单开头的元数据列出了每个文件类型的属性关键字。请参见例 2-1 “BART 清单格式说明”。

all 关键字表示所有文件属性。

BART 引用语法

BART 所用的规则文件规范语言是用于表示非标准文件名的标准 UNIX 引用语法。嵌入的制表符、空格、换行符或特殊字符以八进制格式编码，以使工具能够读取文件名。这种不一致的引用语法会阻止在命令管道中正确处理某些文件名，如包含嵌入回车的文件

名。使用 rules 规范语言可表述复杂的文件名过滤条件；如果仅使用 shell 语法会很难表述这些条件，而且效率也不高。

有关更多信息，请参见 bart_rules(4) 手册页。

BART 报告

在缺省模式下，BART 报告会检查系统上安装的所有文件，但已修改的目录时间戳 (dirmtime) 除外：

```
CHECK all
IGNORE dirmtime
```

如果提供了规则文件，则全局指令 CHECK all 和 IGNORE dirmtime 会按照以上顺序自动前置到规则文件。

BART 输出

将返回以下退出值：

0	成功
1	处理文件时出现非致命错误，如权限问题
>1	出现致命错误，如无效的命令行选项

此报告机制提供了两种类型的输出：详细输出和程序输出：

- 详细输出为缺省输出，已本地化并出现在多行中。详细输出已经过国际化并具有可读性。如果使用 bart compare 命令对两个系统清单进行比较，则会生成一个文件差异列表。

  输出结构如下：

  filename attribute control:control-val test:test-val

filename	在控制清单与测试清单中不同的文件的名称。
attribute	进行比较的清单之间存在差异的文件属性的名称。*control-val* 出现在 *test-val* 之前。如果同一个文件中的多个属性出现差异，则每个差异都将记录在单独的一行中。

  以下是 /etc/passwd 文件中属性差异的示例。输出表明 size、mtime 和 contents 属性已发生变化。

  ```
  /etc/passwd:
  ```

```
size control:74 test:81
mtime control:3c165879 test:3c165979
contents control:daca28ae0de97afd7a6b91fde8d57afa
test:84b2b32c4165887355317207b48a6ec7
```

- 使用 `bart compare` 命令的 `-p` 选项将生成程序输出。该输出适用于程序操纵。输出结构如下：

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

filename	与缺省格式中的 *filename* 属性相同
attribute control-val test-val	每个文件在控制清单与测试清单之间存在差异的文件属性的说明

有关 `bart` 命令支持的属性的列表，请参见“BART 规则文件属性” [36]。

有关更多信息，请参见 `bart`(1M) 手册页。

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（管理主体）	名称形式为 *username*`/admin` 的用户主体（如 `jdoe/admin`）。与一般用户主体相比，管理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 进入和退出事件就属于异步事件。
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，是指可以分配给角色或用户（或在权限配置文件中嵌入）的一种权限，允许执行安全策略原本禁止的某类操作。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）	登录时为用户进程指定的特权集。在未修改的系统上，每个用户的初始可继承特权集等同于登录时获取的基本特权集。
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）	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实现。一个特权集控制对设备的读取权限，另一个特权集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）	当前对进程有效的特权集。
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（主机主体）	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` 便是一个主机主体。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）	进程可以通过调用 `exec` 而继承的特权集。
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）	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）	可供进程使用的特权集。
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）	特权的集合。每个进程都有四个特权集，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）、effective set（有效特权集）、permitted set（允许特权集）和 inheritable set（可继承特权集）。 此外，特权的 basic set（基本特权集）是指登录时为用户进程指定的特权集合。
privilege-aware（特权识别）	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` 操作要求重新验证。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。指的是可以分配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（主机主体）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，并分配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

N

P

Q

R

S

Y

Z