中标麒麟可信操作系统 V6.0

安全管理员手册

中标软件有限公司

上海市徐汇区番禺路 1028 号数娱大厦 10 层（200030）

北京市海淀区北四环西路 9 号银谷大厦 20 层（100190）

广州市天河北路 898 号信源大厦 16 层 1604 室（510898）

目录

中标软件有限公司 CHINA STANDARD SOFTWARE CO., LTD.

软件使用许可协议

尊敬的中标麒麟用户：

首先感谢您选用由中标软件有限公司开发并制作发行的中标麒麟产品。

请在打开本软件介质包之前，仔细阅读本协议条款以及所提供的所有补充许可条款（统称“协议”）。一旦您打开本软件介质包，即表明您已接受本协议的条款，本协议将立即生效，对您和本公司双方具有法律约束力。

1. 使用许可

按照已为之支付费用的用户数目及计算机硬件类型，中标软件有限公司（下称“中标软件”） 向您授予非排他、不可转让的许可，仅允许内部使用由中标软件提供的随附软件和文档以及任何错误纠正（统称“本软件”）。

一 软件使用许可

在遵守本协议的条款和条件的情况下，中标软件给予贵机构非独占、不可转让、有限的许可，允许贵机构至多使用软件的五（5）份完整及未经修改的二进制格式副本，而此种软件副本仅可安装于贵机构操作的电脑中。

一 教育机构使用许可

在遵守本协议的条款和条件的情况下，如果贵机构是教育机构，中标软件给予贵机构非独占、不可转让的许可，允许贵机构仅在内部使用随附的未经修改的二进制格式的软件。此处的“在内部使用”是指由在贵机构入学的学生、贵机构教员和员工使用软件。

一 字型软件使用

软件中包含生成字体样式的软件（“字型软件”）。贵机构不可从软件中分离字型软件。贵机构不可改动字型软件，以新增此等字型软件被作为软件的一部分交付予贵机构时所不具备的任何功能。贵机构不可将字型软件嵌入作为商业产品提供以换取收费或其他报酬的文件。

2. 限制

本软件受到版权（著作权）法、商标法和其他法律及国际知识产权公约的保护。中标软件和/或其许可方保留对本软件的所有权及所有相关的知识产权。对于中标软件或其许可方的任何商标、服务标记、标识或商号的任何权利、所有权或利益，本协议均不作任何授权。

3.关于复制、修改及分发

如果在所有复制品中维持本协议书不变，您可以且必须根据《GNU GPL-GNU 通用公共许可证》复制、修改及分发中标麒麟产品中遵守《GNU GPL-GNU 通用公共许可证》协议的软件，其他不遵守《GNU GPL-GNU 通用公共许可证》协议的中标麒麟产品必须根据符合相关法律之其他许可协议进行复制、修改及分发，但任何以中标麒麟产品为基础的衍生发行版未经中标软件有限公司的书面授权不能使用任何中标软件有限公司的商标或其他任何标志。

特别注意：该复制、修改及分发不包括本产品中包含的任何不适用《GNU GPL-GNU 通用公共许可证》的软件，如中标麒麟产品中包含的输入法软件、字库软件、第三方应用软件等。除非适用法律禁止实施，否则您不得对上述软件进行复制、修改（包括反编译或反向工程）、分发。

4. 有限担保

中标软件向您担保，自购买之日起九十（90）天内（以收据副本为凭证），本软件的存储介质（如果有的话）在正常使用的情况下无材料和工艺方面的缺陷。除上述内容外，本软件按“原样”提供。在本有限担保项下，您的所有补偿及中标软件的全部责任为由中标软件选择更换本软件介质或退还本软件的购买费用。

5. 担保的免责声明

除非在本协议中有明确规定，否则对于任何明示或默示的条件、陈述及担保，包括对适销性、对特定用途的适用性或非侵权性的任何默示的担保，均不予负责，但上述免责声明被认定为法律上无效的情况除外。

6. 责任限制

在法律允许范围内，无论在何种情况下，无论采用何种有关责任的理论，无论因何种方式导致，对于因使用或无法使用本软件引起的或与之相关的任何收益损失、利润或数据损失，或者对于特殊的、间接的、后果性的、偶发的或惩罚性的损害赔偿，中标软件或其许可方均不承担任何责任（即使中标软件已被告知可能出现上述损害赔偿）。根据本协议，在任何情况下，无论是在合同、侵权行为（包括过失）方面，还是在其他方面，中标软件对您的责任将不超过您就本软件所支付的金额。即使上述担保未能达到其基本目的，上文所述的限制仍然适用。

7. 终止

本协议在终止之前有效。您可以随时终止本协议，但必须销毁本软件的全部正本和副本。如果您未遵守本协议的任何规定，则本协议将不经中标软件发出通知立即终止。终止时，您必须销毁本软件的全部正本和副本。

8. 管辖法律

与本协议相关的任何诉讼均受适用的中华人民共和国法律管辖。任何其它国家和地区的选择法律的规则不予适用。

9. 可分割性

如果本协议中有任何规定被认定为无法执行，则删除相应规定，本协议仍然有效，除非删除防碍各方愿望的实现（在这种情况下，本协议将立即终止）

10. 完整性

本协议是您与中标软件就其标的达成的完整协议。它取代此前或同期的所有口头或书面往来信息、建议、陈述和担保。在本协议期间，有关报价、订单、回执或各方之间就本协议标的进行的其他往来通信中的任何冲突条款或附加条款，均以本协议为准。对本协议的任何修改均无约束力，除非通过书面进行修改并由每一方的授权代表签字。

11. 商标和标识

贵机构承认并与中标软件有着以下共识，即中标软件拥有中标软件、中标麒麟商标，以及所有与中标软件、中标麒麟相关的商标、服务标记、标识及其他品牌标识（“中标软

件标记”）。贵机构对中标软件标记的任何使用都应有利于中标软件。

12. 源代码

本软件可能包含源代码，其提供之唯一目的是在符合本协议条款之规定时供参考之用。源代码不可再分发，除非在本协议中有明确规定。

13. 因侵权而终止

如果本软件成为或在任一方看来可能成为任何知识产权侵权索赔之标的，则任一方可立即终止本协议。

14. Java 技术限制

贵机构不可更改“Java 平台界面”（简称“JPI”，即指明为“java”包或“java”包的任何子包中的类），无论通过在 JPI 中创建额外的类，还是通过其他方式导致对 JPI 中的类进行增添或更动，均为不可。如果贵机构创建一个额外的类以及一个或多个相关的 API，而它们（i）扩展 Java 平台的功能；并且（ii）可供第三方软件开发者用于开发可调用上述额外 API 的额外软件，则贵机构必须迅即广泛公布对此种 API 的准确说明，以供所有开发者免费使用。贵机构不可创建、或授权贵机构的被许可人创建以任何方式标示为“java”、“javax”、“sun”的额外的类、界面、子包或 Sun 在任何命名约定中指明的类似约定。参见 Java 运行时环境二进制代码许可的适当版本（目前位于 http://www.java.sun.com/jdk/index.html），以了解可与 Java 小程序和应用程序共同分发的运行时代码的可供情况。

中标麒麟可信操作系统 V6.0 产品介绍

为满足政府、国防、金融、电力、机要、保密等领域对操作系统的高安全性需求，中标软件有限公司（以下简称“中标软件”）基于多年来在操作系统安全和可信计算方面的技术积累，研制推出了国内首款自主可控、高安全等级的可信操作系统软件产品-中标麒麟可信操作系统 V6.0。

结合可信计算技术和操作系统安全技术，中标麒麟可信操作系统 V6.0 通过信任链的建立及传递实现对平台软硬件的完整性度量；提供基于三权分立机制的多项安全功能(身份鉴别、访问控制、数据保护、安全标记、可信路径、安全审计等)和统一的安全控制中心；全面支持国内外可信计算规范（TCM/TPCM、TPM2.0）；兼容主流的软硬件和自主 CPU 平台；提供可持续性的安全保障，防止软硬件被篡改和信息被窃取，系统免受攻击；为业务应用平台提供全方位的安全保护，保障关键应用安全、可信和稳定的对外提供服务。

中标软件还提供基于 Linux 操作系统的安全评估、安全优化、安全加固等安全服务和系统安全定制开发业务。

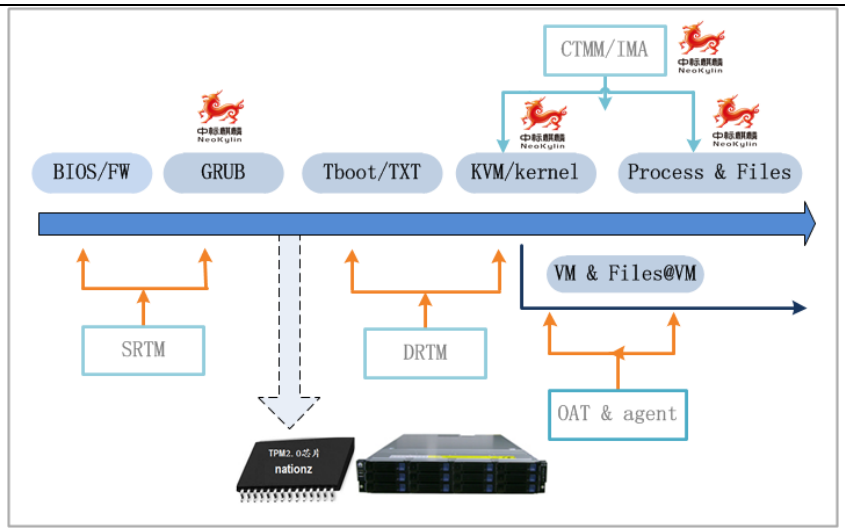
主要特性

■ 操作系统高安全等级

中标麒麟可信操作系统 V6.0 严格遵照可信计算技术规范（TCM/TPCM、TPM2.0）、GB/T 20272-2006 技术要求和国际 CC 标准等进行研制开发。通过操作系统安全的国家标准 GB/T 20272-2006 第四级（结构化保护级）测评认证并获得销售许可。

■ 可信计算实现内核级

国内首款全面支持 TCM/TPCM 和 TPM2.0 可信计算规范的可信操作系统，支持通用和专用可信密码芯片/模块；基于中标软件可信度量模块 CTMM（CS2C Trusted Measure Module）提供可信引导、可信启动和可信运行控制等功能；通过信任链的创建传递过程，实现对平台软硬件的完整性度量；提供基于可信芯片的上层可信功能和图形化的可信管理中心；并实现信任链从物理主机到虚拟化平台的拓展，提供对虚拟机的完整性度量。



■ 安全功能和机制全面

基于 LSM 的安全子系统框架，提供基于三权分立机制的多项安全功能，包括身份鉴别、自主访问控制、强制访问控制、数据机密性和完整性保护、安全标记、可信路径、安全审计等。针对不同的应用场景，系统支持细粒度的强制访问控制 SELinux 和轻量级强制访问控制 SMACK。

■ 系统管理配置灵活

内置主流数据库、中间件和应用服务器的安全策略，同时提供多种图形化安全策略配置和管理工具;基于图形化的安全控制中心实现系统安全可信功能模块化的集中配置和管理，界面友好，简洁易用；用户可以方便快捷完成系统的安全管理。

■ 良好的兼容性

中标麒麟可信操作系统 V6.0 适用于从服务器应用到桌面办公等各种环境，支持各类通用和专业应用；并内置默认的安全策略，实现系统安全和易用的结合，具有良好的软、硬件兼容性。系统支持 64 位应用程序，提供丰富的硬件驱动程序，中标软件有限公司还可协助第三方硬件厂商完成驱动程序的研发和移植，实现专用和特定硬件设备的支持。

系统要求

512MB 物理 RAM（推荐使用 1G 以上 RAM）

5G 以上可用磁盘空间

800x600 以上显示分辨率（推荐采用 1024x768 或更高分辨率）

硬件平台

Intel x86-64（AMD64）

自主 CPU 平台（龙芯、申威、兆芯、众志、Arm64 等）

获得更多的信息

如果出现了本手册不能解决的问题，可以通过如下的方式获得帮助：

阅读和打印 man 页以及 info 页。（man 页和 info 页是系统文档，可以帮助您了解系统提供了哪些可用命令以及如何使用它们）；

- 使用 GNOME 帮助浏览器；
- 登录 www.cs2c.com.cn 网站，查阅相关资料。

技术支持

请您按照中标麒麟可信操作系统 V6.0 产品包装或以下联系方式获取中标软件提供的技术支持服务，包括：

- 所有服务均以远程方式执行；
- 产品安装支持；
- 5＊8 小时电话，邮件，网站、传真等支持；
- 同版本补丁升级服务；
- 远程电话、邮件、网站、传真等支持服务只针对中标麒麟相关产品的安装、使用的问题提供支持，不包含对第三方软硬件的支持服务；
- 服务期按照合同规定起止日期内提供服务。

如果您有其它额外的技术支持需求，请致电中标软件有限公司，我们承诺为您提供优质的服务。

公司网址：www.cs2c.com.cn

客户热线：400-706-1825

电子邮件：support@cs2c.com.cn

公司电话：上海（021）51098866　　北京（010）51659955　广州（020）38182526

公司传真：上海（021）51062866　　北京（010）62800607　广州（020）38182529

第 1 章 双因子认证

1. 功能介绍

双因子认证系统是一套高可靠的可灵活定制的用户权限认证系统，功能分两部分：

1、 在用户指定情况下可以生成指定用户的双因子认证数据，并完成相关环境配置。

2、 对指定了使用双因子认证的用户在登录时进行用户权限验证。

2. 权限管理

双因子的权限管理是遵照三权分立系统规定的权限原则执行的，与传统的 Linux 中用户权限管理有一些差异。安全管理员可以操作任意非安全管理员用户的所有身份鉴别信息；非安全管理员（包括系统管理员和审计管理员）可以完全自由操作自己的身份鉴别信息。普通用户可以在授权范围内操作自己的身份鉴别信息。

3. 配置与使用

3.1 双因子认证配置

您可以通过终端命令行来配置双因子认证。

3.1.1 命令配置双因子认证

非安全管理员的配置：

（1）在终端下运行：douauth –i 。如图 1-1 所示。

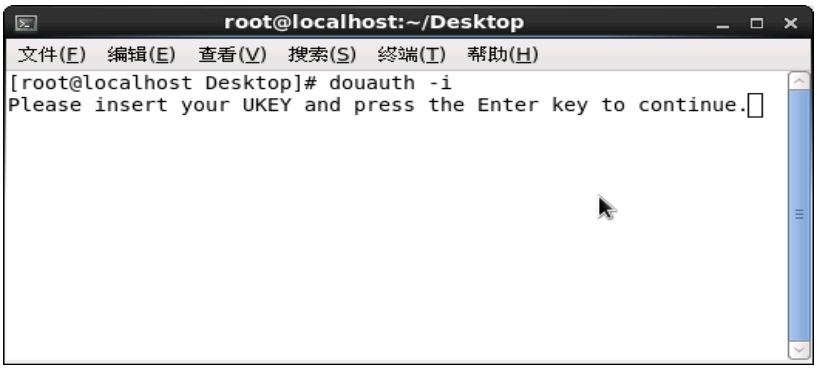


图 1-1 终端配置双因子认证

（2）插入 UKEY，并按 Enter 键继续下步操作。如图 1-2 所示。

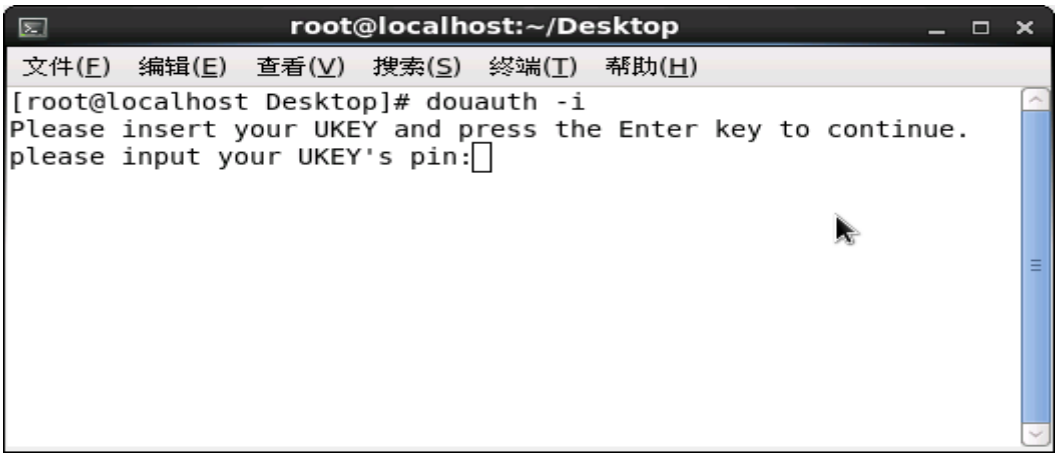


图 1-2 插入 UKEY

如果插入的 UKEY 与用户不对应或者输入了错误的 pin 码，则会提示错误。图 1-3，为输入错误的 pin 码时提示的错误。

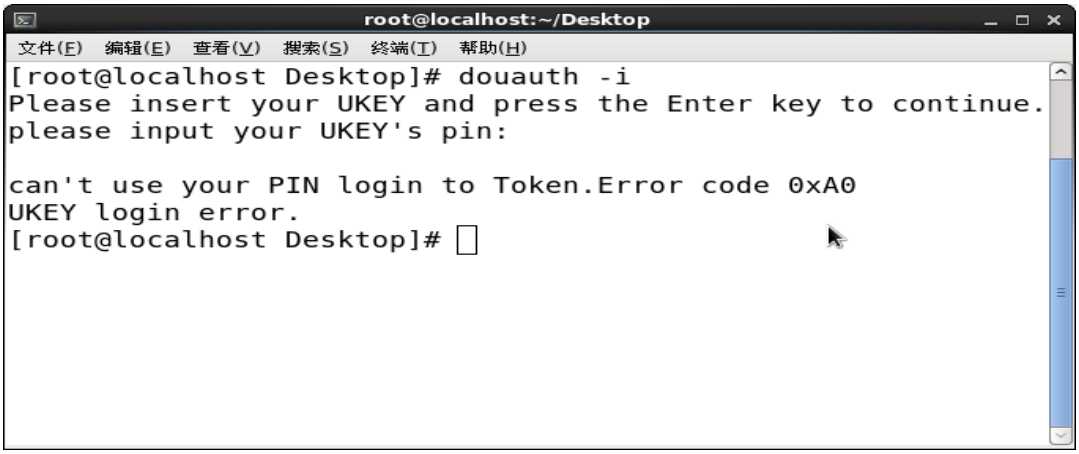


图 1-3 输入错误的 pin 码

（3）插入正确的 UKEY，并输入正确的 pin 码，则可以按照提示开启或者关闭 UKEY，如图 1-4 所示。

图 1-4 开启或关闭 UKEY

（4）如果需要开启 UKEY，则需要进行步骤 6。否则会提示用户 UKEY 认证功能被关闭，如图 1-5 所示。

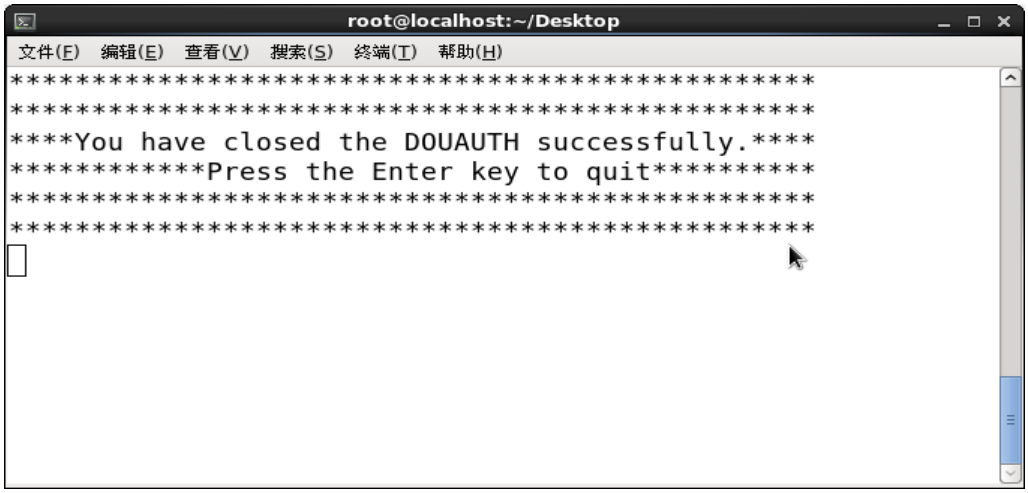


图 1-5 UKEY 已关闭

（5）提示是否选择 UKEY 与系统同步的功能。如图 1-6 所示。

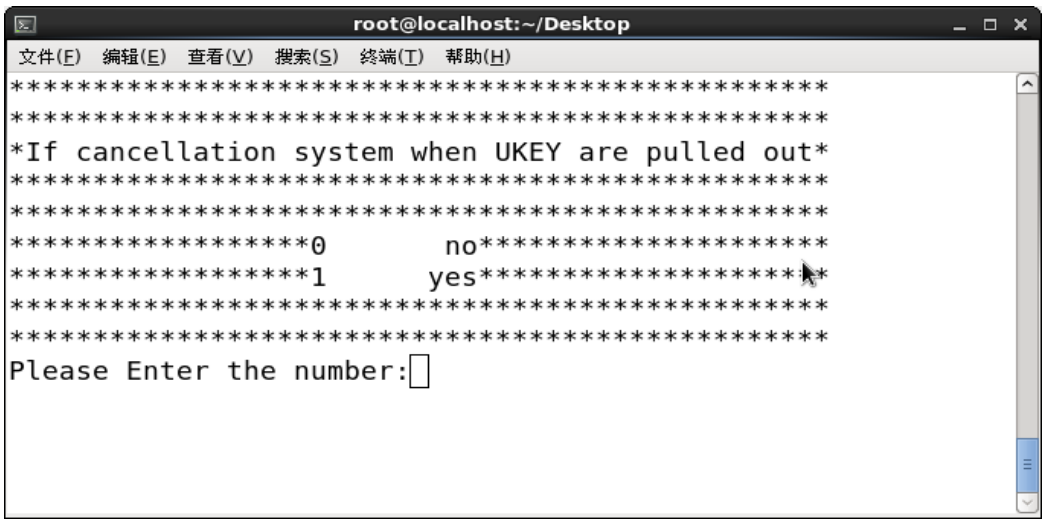


图 1-6 设置 UKEY 同步功能

（6）最后，提示双因子认证开启成功，如图 1-7 所示。

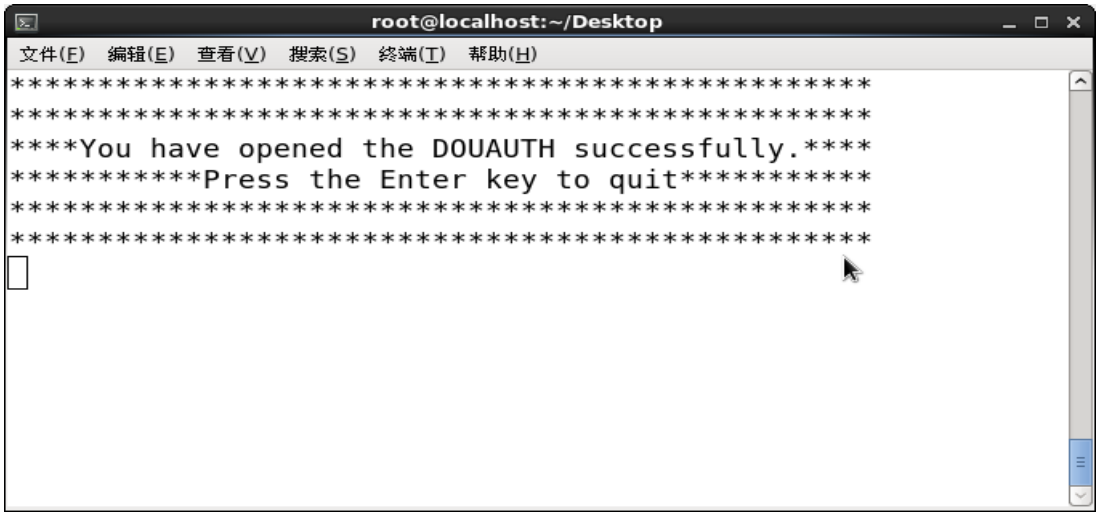


图 1-7 双因子认证设置成功

安全管理员配置：

（1）安全管理员和非安全管理员的前三步配置是一样的，这里不再赘述。插入正确的 UKEY 并输入正确的 pin 码之后，提示用户选择配置安全管理员的双因子认证还是普通用户的双因子认证。如图 1-8 所示。

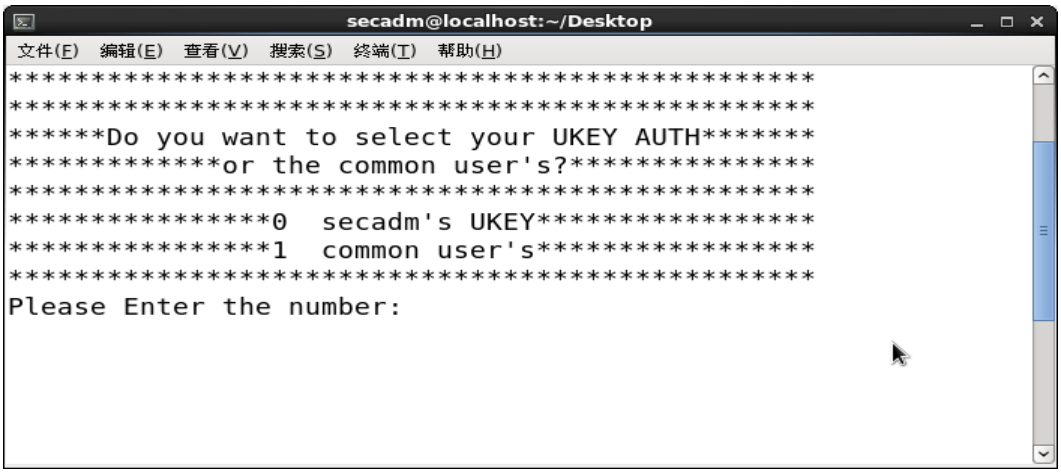


图 1-8 选择需要配置的用户

（2）如果选择安全管理员的配置，则步骤和非安全管理员相同。否则，进入如图 1-9 界面，输入普通用户的用户名。

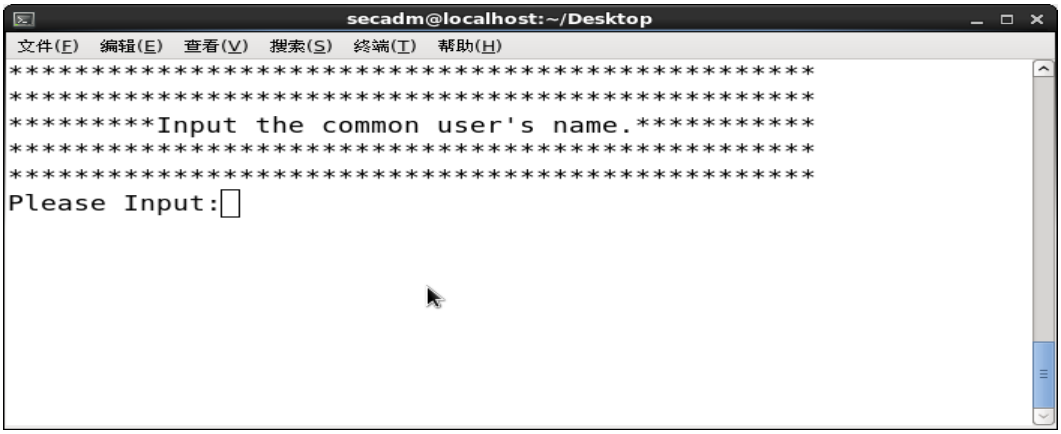


图 1-9 输入普通用户用户名

（3）进入选择开启或关闭双因子认证界面，如图 1-4 所示。

（4）如果选择关闭双因子，则如图 1-5 所示，成功关闭双因子认证。否则，需要选择是否强制配置双因子认证，如图 1-10 所示。

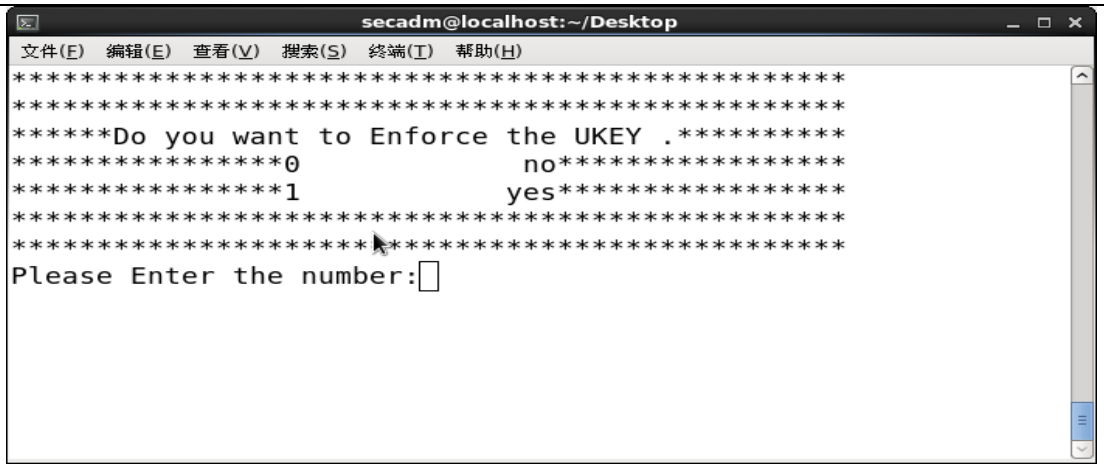


图 1-10 选择是否强制配置双因子认证

（5）选择是否设置 UKEY 和系统同步，如图 1-6 所示。

（6）提示开启双因子认证成功，如图 1-7 所示。

3.2 双因子认证使用

开启双因子认证的用户，开机时需要插入 USBKEY 设备进行权限验证，验证失败时将无法登录系统。

当系统设置为默认安全级别时，为了保证高安全性，双因子登录系统要求管理员必须启用双因子认证配置，否则拒绝任何管理员用户登录。管理员在进行模式切换操作时，如果有管理员没有启用双因子认证配置，系统将按一个预制的 UKEY 设备信息自动启用对应管理员用户的双因子认证配置。

4. 注意

对特权用户使用双因子认证，要注意对认证设备和口令的保存，一旦丢失相关认证因子，将会导致系统无法登录。

第 2 章 安全策略管理

1. 概述

本章将主要介绍中标麒麟可信操作系统 V6.0 中提供的安全策略的支持，安全策略是基于 MAC（强制访问控制）、RBAC（角色访问控制）、TE（类型实施）和 MLS（多层安全）来实现，通过这些我们达到了进程级的访问控制和进程最小原则。下面详细介绍了供安全管理员进行安全配置工具的命令行界面，这些命令主要用于获取/设置主客体的安全标记，对用户进行角色管理，安全策略调整等操作。在中标麒麟可信操作系统中，这些安全管理命令只有安全管理员才能成功执行，其它用户执行将失败或者无效。

2. 基本概念

主体和客体：主体指的是访问者(一般为一个进程)，客体指的是被访问的资源。根据资源的类型不同，在策略配置语言中被分为不同的客体，如 dir、file、process 等。

安全上下文：指标记在所有事物上的扩展属性，包括 SELinux 用户、角色、域或类型，如果使用 MLS(多级安全，以下会有介绍)策略还包括安全级。

类型和域：是分配给一个对象并决定谁可以访问这个对象，域对应进程，类型对应其他对象。

域转换：进程以给定的进程类型运行的能力称为域转换，需满足三个转换条件：一个进程新的域类型有对一个可执行文件类型的 entrypoint 访问权限，进程的当前（或以前）域类型对入口点文件类型拥有 execute 访问权限，进程的当前域类型对新的域类型拥有 transition 访问权限。

角色：和一些域相关联，决定哪些域可以使用。

安全策略用户：和标准 linux 用户对应，影响哪个域可以进入。

3. 安全策略模式

我们的安全策略提供了两种模式：允许模式和强制模式，系统在正常启动后就进入了允许模式。允许模式是为了方便我们服务器配置，即使我们的安全策略没有允许这样的操作仍可以执行，但会被审计下来；强制模式是只要没有这样的规则操作就不被允许。我们

的安全服务器系统在普通模式下使用的就是允许模式的策略，在默认模式下则使用强制模式的策略来加固我们服务器的安全。

3.1 当前安全策略模式查询

我们提供了一些工具命令可以查询当前的安全策略模式，如/usr/sbin/getenforce 和/usr/sbin/sestatus。所有用户都可以通过这些命令进行查询。

getenforce 命令输出若为 Permissive 则表示当前处于允许模式，若为 Enforcing 则表示当前处于强制模式。

sestatus 命令的输出可能为以下内容：

SELinux status： enabled

SELinuxfs mount： /selinux

Current mode： permissive

Mode from config file： permissive

Policy version： 24

Policy from config file： mls

它不仅反应了当前安全策略的模式，同时也可以得到其他安全策略的信息：安全策略开启状态、安全策略文件系统挂载位置、配置文件中的默认模式、安全策略版本和安全策略目录名称。

3.2 安全策略模式修改

修改当前安全策略模式的命令如下：

setenforce 1 设置为强制模式

setenforce 0 设置为允许模式

系统中三个管理员都可以使用 setenforce 1 进入到强制模式，但只有安全管理员可以使用 setenforce 0 进入到允许模式。

3.3 禁用安全策略

在系统出现故障时我们也可以先禁用我们的安全策略进入到维护模式进行修复。进入方法如下：

1. 系统启动时进入 grub 菜单。

2. 输入 grub 密码修改内核启动参数，将参数 selinux=1 enforcing=1 修改为 selinux=1

enforcing=0 1。

3. 启动系统。

这样系统就直接进入到维护模式，由 root 用户对系统进行修复。

4. 安全策略配置命令使用

为了方便用户对安全策略进行配置，我们也提供了以下命令对当前安全策略进行配置：

4.1 查看标记命令

系统中一些常用命令的-Z 选项都提供了查看文件或进程安全上下文的功能，如：ls -Z id -Z 和 ps -Z，这些命令加了-Z 选项后就会显示该文件或进程的安全上下文，如 ls -Z 一个文件，输出以下内容：

-rw-r--r--. root root sysadm_u:object_r:admin_home_t:s0

其中 sysadm_u 就表示该文件属于 sysadm_u 安全策略用户，object_r 表示该文件属于客体类型的角色，admin_home_t 表示该文件的类型，我们的策略就是根据这些标记来制定规则进行访问控制的。

4.2 修改标记命令

安全管理员可以通过一些命令对系统中所有文件的安全上下文进行操作，如：chcon、setfiles 和 restorecon 命令。

/bin/ls 命令为 bin_t 类型，安全管理员可以输入 chcon -t sbin_t /bin/ls 将 bin_t 修改为 sbin_t 类型。

chcon 命令的-u 参数可以修改该文件安全上下文中的安全策略用户，-r 参数可以修改它的角色，-l 参数可以修改它的安全级别，其他具体使用可以查看它的帮助手册。

setfiles 命令可以根据我们的安全策略的标记文件对系统中所有文件进行标记。如：

setfiles /etc/selinux/mls/contexts/files/file_contexts /

restorecon 命令可以恢复该文件的默认安全上下文，如：restorecon -r filename。

在对文件的安全上下文标记进行操作时首先都会检查这个标记的正确性，然后再去进行操作，具体使用可以查看用户手册。

4.3 用户、角色和域的关系

登录程序如（login，sshd）负责映射 Linux 用户到安全策略用户，在登录时，如果安全策略用户标识符恰好和一个 Linux 用户标识符完全相同，匹配的安全策略用户标识符成为初始 shell 进程安全上下文的用户标识符。安全管理员也可使用 semanage 命令完成映射，如：semanage login -a -s charlie_u charlie，表示 Linux 用户 charlie 作为安全策略用户 charlie_u 登录系统。

使用 semanage login -l 命令可以查看，如下图：

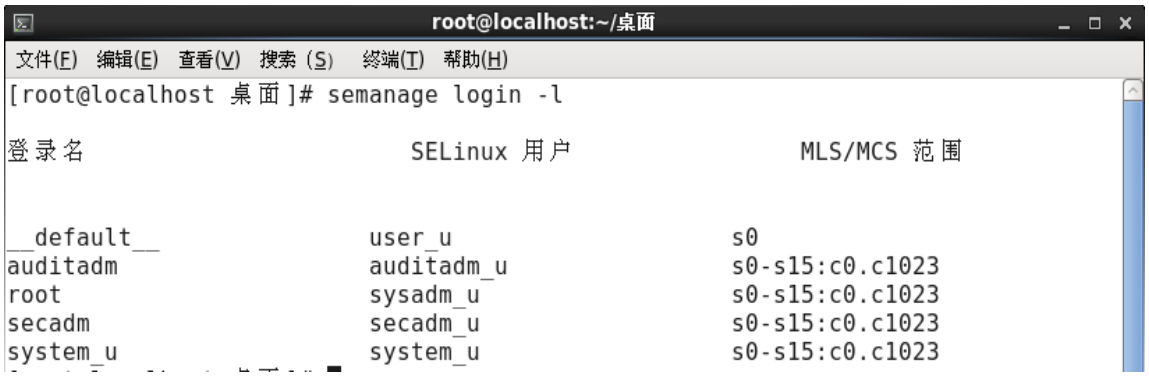


图 2-1 用户对应关系

一个用户可以对应多个角色，但在同一时刻只能作为其中一个角色，可以通过策略管理工具 semanage 输入 semanage user -a -R mgr_r -R charlie_r -P user charlie_u 来创建一个可以作为角色 mgr_r 和 charlie_r 的安全策略用户 charlie_u。

可通过 semanage user -l 进行查看，如下图：

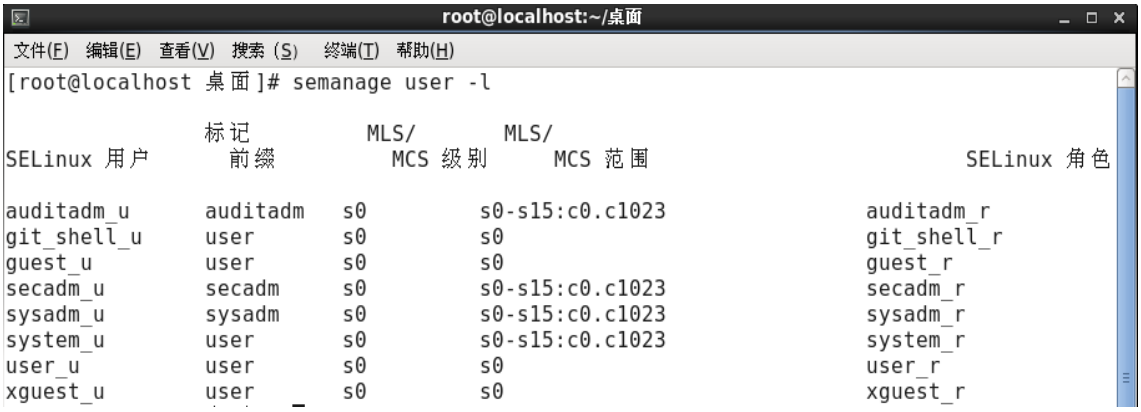


图 2-2 角色对应关系

角色仅仅是一套域类型的集合，方便与用户建立联系，具体的访问控制都是通过进程的类型和客体的类型根据制定的规则来进行控制。

4.4 端口配置

系统中所有端口同样也被标记了类型，安全管理员可以通过

semanage port -a -t vnc_port_t -p udp 222

来添加一个标记了类型的端口。

可通过 semanage port -l 进行查看，如下图：

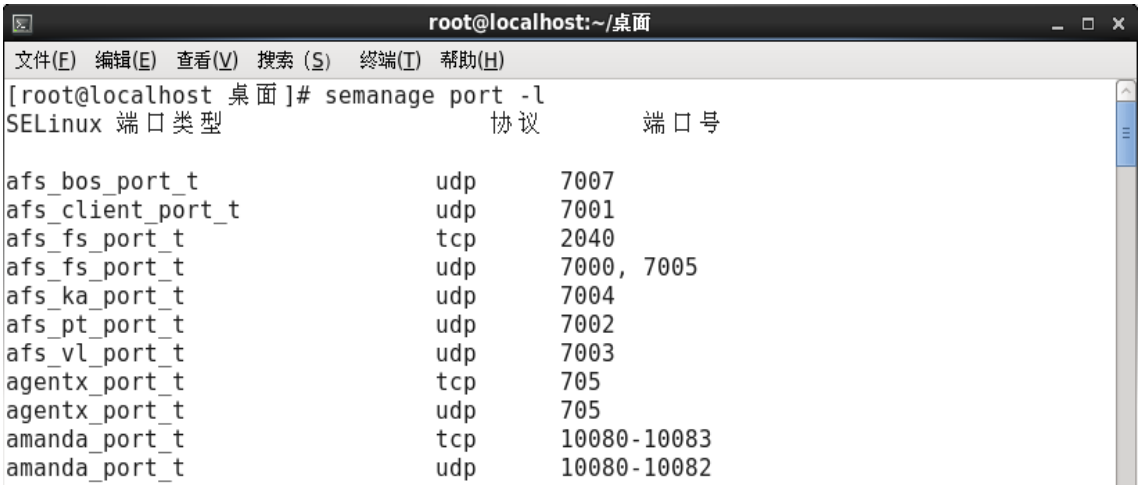


图 2-3 端口对应关系

5 安全策略调整

对于专业安全管理员在获得我们安全策略源码后自己对系统的安全策略进行灵活配置，打造适合自己的安全策略。

5.1 策略规则分类

5.1.1 访问控制规则

访问控制约束由 4 元组<主体类型，目标客体类型，目标客体安全类，访问向量>来确定。其中，安全类指明目标对象所属的安全类别（如目录、文件、socket 等），当目标类型属于不同的安全类时，对应的访问许可也不相同；访问向量指明了满足条件后，主体可以对客体进行的操作。

5.1.2 进程类型转换规则

类型转换是 TE 策略的重要组成部分。系统在执行 fork 操作创建新进程时，子进程的类型继承父进程的类型，在执行 exec 操作变换执行影像时，根据执行影像的类型和进程类型转换规则，判断新进程是否发生类型转换。

典型情况下，TE 类型的转换仅考虑当前进程和被执行映象的 TE 类型即可，即仅考虑 TE 策略本身的决定因素。系统定义的标准类型转换规则格式为：

<源类型（进程当前类型）；目标类型（当前执行的可执行文件类型）；process；新类型（进程转换后的类型）>。

process 为关键字，指明此规则为进程的类型转换规则。该规则在每次 exec 执行时均会判断，根据当前进程类型以及当前执行的可执行文件类型来确定新进程的类型。

5.1.3 角色许可类型转换规则

角色许可类型规则作为进程类型转换规则的辅助规则存在。设计该规则的目的是为了使角色参与到 TE 策略的访问控制中。

角色许可类型转换约束由 4 元组<角色，相关客体类型，process，结果类型>构成。角色许可类型规则与上面的进程类型转换关系类似，不同点在于条件中的第一项不是进程的类型，而是当前登录的角色。这里缺省的认为当前进程的类型为 user_t，即用户登录后尚未发生类型转换。只有进程的源类型是 user_t 时，这类约束条件才会起作用。

角色许可类型规则定义了该角色的进程允许具有的类型集合。在角色参与到进程类型转换规则进行转换判断时，要查询角色许可类型集合。即如果当前进程类型为 user_t，并且查找到匹配的进程类型转换规则，得出新进程类型结果后，要查询角色许可类型规则，判断当前登录角色是否允许此类转换发生。如果角色许可类型集合包含该类型，则允许转换的发生；否则，转换不能发生。例如 secadm 角色登录时，user_t 可以转换到 secadm_t（执行某些安全管理命令时）。

5.1.4 文件客体生成规则

文件客体生成规则定义了在创建文件客体时，应赋予新文件何种安全属性。相应的文件类型生成规则为：

<当前进程类型；父目录类型；file；新类型>

规则的含义为：当进程在某目录下生成新文件时，可以根据当前进程类型以及目录结点类型来确定新生成文件/目录的类型。

5.2 策略规则查询

为了方便对策略规则进行调整，我们提供了 sesearch 命令查询当前生效的策略规则。使用方法如下：

5.2.1 查询访问控制规则

如：sesearch -s secadm_t -t bin_t -c file --allow 表示查询 secadm_t 类型进程对 bin_t 类型的文件具有的允许权限。

进程类型转换规则

如：sesearch -s sysadm_t -t passwd_exec_t --type 表示 sysadm_t 类型进程在执行 passwd_exec_t 类型文件时可能转换到的进程类型。

角色许可类型转换规则

如：sesearch --role_trans

显示所有角色自动转化的规则。

5.2.2 文件客体生成规则

如：sesearch -s sysadm_t -t tmp_t -c file --type 表示 sysadm_t 类型进程在 tmp_t 目录下生成文件的默认类型规则。

5.3 客体类别

与文件相关的客体类别如下表所示：

客体类别	描述
blk_file	块文件
chr_file	字符文件
dir	目录
fd	文件描述符
fifo_file	命名管道
file	普通文件
filesystem	文件系统（如一个真实的分区）
lnk_file	符号链接
sock_file	UNIX 域套接字

表 2-1 文件客体类型

客体类别 file 和 dir 分别代表普通文件和目录，普通文件就是那些存储数据的文件，它们是大多数系统上最常见的客体了，目录在 Linux 中也是一个特定的文件，它是独一无

二的，因为它们可能还包含有其他客体。

lnk_file 客体类别代表符号链接，大多数情况下，它非常重要，它可以区别普通文件和符号链接，这样可以预防常见的攻击，恶意进程和用户可以创建符号链接，这样可能引起某个进程访问或修改本不是它们打算要访问或修改的文件，独立的 lnk_file 客体类别允许编写预防这些攻击类型的策略。

客体类别 fifo_file 和 sock_file 表示用于 IPC 的特定文件，fifo_file 客体类别代表 fifo 文件，也叫做命名管道，sock_file 客体类别联合 UNIX 域套接字控制创建、访问等与文件有关的客体的能力，我们讨论 UNIX 域套接字客体类别及它们与下一节将要谈到的套接字文件的关系。

在 Linux 中，设备通常是通过在/dev/目录下的特定文件来进行访问的，这些文件通过主/次设备号表示块和字符设备。字符设备是程序以字节流形式读或写入数据的设备，块设备是将数据以更大块进行传递的设备，chr_file 和 blk_file 客体类别分别表示字符设备和块设备。

最后两个客体类别是文件系统和文件描述符，它们不是典型的 Linux 客体，filesystem 客体类别表示挂载的文件系统，这个客体类别控制全局操作如挂载或查询限额，例如：使用 filesystem 客体类别，我们可以只运行挂载支持存储安全上下文的文件系统。所有特定类型的文件系统（如 ext4）在策略中都使用相同的 fs_use 语句获取默认的标记定义，第 10 章 "客体标记"中将会描述 fs_use，挂载分区时如果使用了 context mount 选项，默认的类型可能会被覆盖掉。

文件描述符表示打开的与文件有关的客体，存在于进程中，即使与文件有关的客体明显不同，它们表示内核数据结构，通常认为文件描述符是文件有关的客体的基础，的确，标准 Linux 访问控制不能单独在文件描述符上提供访问控制，这种策略忽略了文件描述符是可以在进程之间进行传递的资源的事实，通常，子进程会从父进程那里继承文件描述符，这个继承并不总是有利的，在许多 Linux 编程指南中都警告最好减少文件描述符继承，特别是后台进程，为了标识这个和其它问题，我们以 fd 客体类别为例，在安全策略中，它代表文件描述符，使用这个客体类别阻止文件描述符在进程间传递或继承就成为可能，值得注意的是，有权使用文件描述符并不意味着就可以访问与文件有关的客体，进程必须对这些文件也要有访问许可才行。

与网络相关的客体类别如下表所示：

客体类别	描述
association	IPsec 安全联盟
key_socket	PF_KEY 协议家族的套接字，用于管理 IPsec 中的密钥
Netif	网络接口（如 eth0）
netlink_audit_socket	用于控制审核的 Netlink 套接字
netlink_dnrt_socket	用于控制 DECnet 路由的 Netlink 套接字
netlink_firewall_socket	用于创建用户空间防火墙过滤器的 Netlink 套接字
netlink_ip6fw_socket	用于创建用户空间防火墙过滤器的 Netlink 套接字
netlink_kobject_uevent_socket	用于在用户空间接收内核事件通知的 Netlink 套接字
netlink_nflog_socket	用于接收 Netfilter 日志消息的 Netlink 套接字
netlink_route_socket	用于控制和管理网络资源如路由表和 IP 地址的 Netlink 套接字
netlink_selinux_socket	用于接收策略载入通知，强制模式切换和清空 AVC 缓存的 Netlink 套接字
netlink_tcpdiag_socket	用于监视 TCP 连接的 Netlink 套接字
netlink_socket	所有其它的 Netlink 套接字
netlink_xfrm_socket	用于获取、管理和设置 IPsec 参数的 Netlink 套接字
Node	代表一个 IP 地址或一段 IP 地址的主机
packet_socket	协议在用户空间执行的原始套接字
rawip_socket	既不是 TCP 也不是 UDP 的 IP 套接字
tcp_socket	TCP 套接字
udp_socket	UDP 套接字
unix_dgram_socket	本地机器上（unix 域）的 IPC 数据报套接字
unix_stream_socket	本地机器上（unix 域）的 IPC 流套接字

表 2-2 网络客体类型

node，netif，packet_socket，rawip_socket，tcp_socket，udp_socket 和 socket 客体类别控制典型的网络访问，netif 客体类别代表网络接口，每个有名字的网络接口（如 eth0，eth1）都是通过 netif 客体类别的实例进行表现的，由 IP 地址或地址段进行标识的网络上的远程主机是通过 node 客体类别表现的，使用 node 客体类别，我们可以限制主机（通过 IP 地址）上的哪个进程可以使用网络，前面列出的不同种类的套接字客体类别代表协议套接字类型，成功发送或接收网络数据需要所有有关的 netif，node，socket 客体类别实例的许可。

标准网络套接字是由协议分配的（在调用 socket(2)系统调用时确定的），不同的 socket 客体类别允许我们限制应用程序发送或接收数据包的类型，这对限制应用程序发送原始数据包的能力特别有用。客体类别 tcp_socket 和 udp_socket 分别代表 TCP 和 UDP 套接字，rawip_socket 客体类别代表发送原始 IP 数据包的套接字，packet_socket 客体类别代表发送其它类型的原始数据包的套接字，所有其它的套接字都由 socket 客体类别表示。

使用 IP 安全（IPsec）的通讯拥有额外的资源，由客体类别 association 和 key_socket 表示，IPsec 安全联盟是为通讯提供安全服务的连接，association 客体类别代表 IPsec 联盟，IPsec 需要通过密钥管理（PF_KEY）套接字管理密钥，它通过 key_socket 客体类别进行表现。

Linux 系统上的本地通讯可以使用 unix 域套接字（PF_UNIX）实现，这些套接字通常用于本地 IPC，面向连接的套接字也叫做流套接字，通过 unix_stream_socket 客体类别进行表现，数据报套接字通过 unix_dgram_socket 表现，unix 域套接字可以与文件系统上的某个特定文件关联起来，让其它应用程序很容易就连接到套接字，这个文件通过 sock_file 客体类别表现，它是一个与文件有关的客体类别，前面已经说过了。

安全策略中最后一组套接字是 Netlink 套接字，这些套接字最初是开发用于在 Linux 最后提供一个标准意义的网络配置，现在它们常常用于在内核与用户空间之间通讯，基于协议类型的不同有多种表示 Netlink 套接字的客体类别，常见的 netlink_socket 表示无特定客体类别的保留协议。

与 IPC 相关的客体类别如下表所示：

客体类别	描述
ipc	已经没有使用了
msg	消息队列中的消息

msgq	消息队列
sem	信号量
shm	共享内存段

表 2-3 IPC 客体类型

与 IPC 有关的客体类别代表 System V IPC 资源（参考表 3-3），msgq 和 msg 客体类别分别代表消息队列和消息队列中的消息，sem 客体类别代表信号量，shm 客体类别代表共享内存段，注意对关于所有 System V IPC 资源的全局系统信息的访问是通过 system 类别上的许可进行控制的。

其它杂类客体类别如下表所示：

客体类别	描述
capability	Linux 中表示权利的特权
process	SELinux 中的进程
security	内核中的 SELinux 安全服务器
system	整个系统

表 2-4 其他客体类型

capability 客体类别代表标准 Linux 访问控制模式下的进程权利，这个客体类别允许 SELinux 控制授给某个进程的权利，这些权利包括否决任意访问控制（许可模式）和发送原始网络数据包，这个客体类别和它的许可允许控制每个进程是否可以使用标准 Linux 授予它的权利。

剩下的两个客体类别 security 和 system 分别代表对 SELinux 安全服务器的特定资源和系统的访问，它们是唯一的，这些客体类别每一个都只有一个实例，反映出只有一个安全服务器和系统。

5.4 操作许可

标准 Linux 对应了读、写、执行三类许可，而在我们的安全策略中这些许可被扩展为更多的类型，可以更细粒度地进行控制。

与文件有关的客体类别许可如下表所示：

许可	描述
append	附加到文件内容（即用 o_append 标记打开）
create	创建一个新文件
entrypoint*	通过域转换，可以用作新域的入口点的文件
execmod*	使被修改过的文件可执行（含有写时复制的意思）
execute	执行，与标准 Linux 下的 x 访问权一致
execute_no_trans*	在访问者域转换的执行文件（即没有域转换）
getattr	获取文件的属性，如访问模式（例如：stat，部分 ioctls）
ioctl	ioctl（2）系统调用请求
link	创建一个硬链接
lock	设置和清除文件锁
mounton	用作挂载点
quotaon	允许文件用作一个限额数据库
read	读取文件内容，对应标准 Linux 下的 r 访问权
relabelfrom	从现有类型改变安全上下文
relabelto	改变新类型的安全上下文
rename	重命名一个硬链接
setattr	改变文件的属性，如访问模式（例如：chmod，部分 ioctls）
swapon	不赞成使用。它用于将文件当做换页/交换空间
unlink	移除硬链接（删除）
write	写入文件内容，对应标准 Linux 下的 w 访问权

表 2-5 文件客体许可类型

进程客体类别许可如下表所示：

许可	描述
dyntransition	允许进程动态地转移到新的上下文中

execheap	产生一个堆栈可执行体
execmem	产生一个匿名的映像或可写的私有文件映像可执行体
execstack	产生进程堆栈可执行体
fork	派生两个进程
getattr	通过/proc/[pid]/attr/目录获取进程的属性
getcap	获取这个进程允许的 Linux 能力
getpGID	获取进程的组进程 ID
getsched	获取进程的优先级
getsession	获取进程的会话 ID
noatsecure	禁用清除安全模式环境，允许进程在 execve(2)上禁用 glibc 的安全模式特性
ptrace	跟踪程序执行的父进程或子进程
rlimitnh	在 execve(2 上)继承进程资源限制
setcap	为进程设置允许的 Linux 能力
setcurrent	设置当前的进程上下文，当进程试图执行一个动态域转换时，这是第一个检查的能力
setexec	下一次调用 execve(2)时覆盖默认的上下文
setfscreate	允许进程设置由其创建的客体的上下文
setpGID	设置进程的组进程 ID
setrlimit	改变进程硬性资源限制
setsched	设置进程的优先级
share	允许与克隆的或派生的进程共享状态
siginh	在 execve(2)上继承信号状态
sigkill	发送 sigkill 信号
sigchld	发送 sigchld 信号
signal	发送一个非 sigkill，sigstop 或 sigchld 的信号

signull	不发送信号测试另一个进程的存在性
sigstop	发送 sigstop 信号
transition	在 execve(2)上转换到一个新的上下文

表 2-6 进程客体许可类型

5.5 策略语言编写

策略是一套指导 SELinux 安全引擎计算安全决策的规则，它定义了文件客体的类型、进程的域、使用限制进入域的角色及访问许可的规则表达式等。

5.5.1 声明语句

类型声明：type type_name [alias alias_set];alias 表示别名，通常用于策略改变时保证一致性。

属性声明和关联：attribute file_type; type shadow_t file_type; 将关联的文件类型声明为一个属性，授予该属性访问权就可以访问关联的文件类型。

角色声明：role user_r;

5.5.2 规则语句

TE allow 规则：allow user_t bin_t ： file {read execute};user_t 表示源类型（主体或域），bin_t 为目标类型（客体），file 是客体分类，read execute 表示主体对客体允许的操作。一个域为 user_t 的进程可以对类型为 bin_t 的文件进行 read 和 execute。

默认域转换规则：type_transition user_t passwd_exec_t：process passwd_t;

角色关联类型：role user_r types passwd_t;

角色关联用户：user joe roles {user_t};

角色允许：allow staff_r sysadm_r;在域转换时改变角色。

角色转换：role_transition sysadm_r http_exec_t system_r；一个角色为 sysadm_r 的进程在执行类型为 http_exec_t 的文件时将尝试转换为 system_r 角色，要转换成功角色允许也是必需的。

角色控制：dominance {role super_r{role sysadm_r;role secadm_r;}}角色 super_r 控制后两者角色，拥有两者合并后的类型。

5.5.3 约束语句

约束是不考虑策略中 allow 规则对具体的许可实现全局的约束。constrain 是基于角色、

用户、源和目标安全上下文的类型做更多的访问限制，validatetrans 是通过在进程的新旧上下文之间定义基于约束的关系限制改变客体上下文的能力，MLS 的约束使用 mlsconstrain 和 mlsvalidatetrans。constrain process transition (u1==u2)，用户 u1 等于 u2 时进程可以转换。

5.6 m4 宏

安全策略中使用 m4 宏来提高策略语言的模块化和可重用性，将一些经常使用的策略语言定义为宏，方便策略语言的编写，在编译时通过预编辑器将它展开。

5.6.1 客体权限类型宏

这种类型的宏主要作用是将常用的多个权限集合或客体集合组成一些特定的宏，方便在策略语句中被调用、理解和管理，关于客体和权限的示例分别如下：

1) define(`dir_file_class_set'， `{ dir file lnk_file sock_file fifo_file chr_file blk_file }')，表示用 dir_file_class_set 代替后面这些客体类型的集合。

2) define(`stat_file_perms'， `{ getattr }')

define(`x_file_perms'， `{ getattr execute }')

用 stat_file_perms 代替 getattr 权限，用 x_file_perms 代替 getattr 和 execute 的权限集合。

5.6.2 文件类型宏

这种类型宏主要是对于不同文件类型常用的允许语句定义为一些宏，使用这些宏可以简化整个策略。根据文件类型的不同又分为以下几类：

1) 目录文件宏

```
define(`getattr_dirs_pattern'，`
    allow $1 $2：dir search_dir_perms;
    allow $1 $3：dir getattr_dir_perms;')
```

其中$1、$2、$3 分别代表调用这个宏的三个参数，$1 表示域类型，$2 表示装载目录的目录类型，$3 表示目录类型，search_dir_perms 和 getattr_dir_perms 是上面介绍的权限集合的 2 个宏，对应{getattr search}和{getattr}两个权限集。这个宏的作用是允许$1 域可以得到$3 类型目录的属性。

2) 普通文件宏

```
define(`setattr_files_pattern'，`
        allow $1 $2：dir search_dir_perms;
```

allow $1 $3：file setattr_file_perms;')

其中$1 表示域类型，$2 表示装载文件的目录类型，$3 表示文件类型。这个宏的作用是允许$1 域可以设置$3 类型文件的属性。

3) 符号链接文件宏

define(`read_lnk_files_pattern'，`

allow $1 $2：dir search_dir_perms;

allow $1 $3：lnk_file read_lnk_file_perms;')

其中$1 表示域类型，$2 表示装载符号链接文件的目录类型，$3 表示符号链接文件类型，read_lnk_file_perms 对应{read getattr}权限集。这个宏的作用是允许$1 域可以得到$3 类型符号链接文件的属性并可以读这个文件。

4) 有名或无名管道文件宏

define(`append_fifo_files_pattern'，`

allow $1 $2：dir search_dir_perms;

allow $1 $3：fifo_file append_fifo_file_perms;')

其中$1 表示域类型，$2 表示装载管道文件的目录类型，$3 表示管道文件类型，append_fifo_file_perms 对应{ getattr append lock ioctl }权限集。这个宏的作用是允许$1 域可以得到$3 类型管道文件的属性、追加文件内容、设置文件锁状态、IO 控制。

5) socket 文件宏

define(`write_sock_files_pattern'，`

allow $1 $2：dir search_dir_perms;

allow $1 $3：sock_file write_sock_file_perms;')

其中$1 表示域类型，$2 表示装载 socket 文件的目录类型，$3 表示 socket 文件类型，write_sock_file_perms 对应{ getattr write append }权限集。这个宏的作用是允许$1 域可以得到$3 类型 socket 文件的属性、写文件、追加文件内容。

6) 块设备文件宏

define(`rw_blk_files_pattern'，`

allow $1 $2：dir search_dir_perms;

allow $1 $3：blk_file rw_blk_file_perms;')

其中$1 表示域类型，$2 表示装载块设备文件的目录类型，$3 表示块设备文件类型，rw_blk_file_perms 对应{ getattr read write append ioctl lock }权限集。这个宏的作用是允许

$1 域可以得到$3 类型块设备文件的属性、读写文件、追加文件内容、IO 控制、设置文件锁状态。

7) 字符设备文件宏

```
define(`create_chr_files_pattern'，`
        allow $1 self：capability mknod;
        allow $1 $2：dir add_entry_dir_perms;
        allow $1 $3：chr_file create_chr_file_perms;')
```

其中$1 表示域类型，$2 表示装载字符设备文件的目录类型，$3 表示字符设备文件类型，add_entry_dir_perms 对应{ getattr search lock ioctl write add_name }权限集，create_chr_file_perms 对应{ getattr create }。这个宏的作用是允许$1 域可以在$2 文件夹下创建一个字符设备文件。

8) 文件类型转换宏

```
define(`filetrans_add_pattern'，`
        allow $1 $2：dir { list_dir_perms add_entry_dir_perms };
        type_transition $1 $2：$4 $3;')
```

其中$1 表示域类型，$2 表示目录类型，$3 表示新的类型，$4 表示客体，list_dir_perms 对应{ getattr search read lock ioctl }权限集，add_entry_dir_perms 上面已介绍。这个宏的作用是$1 进程在类型为$2 的目录下尝试创建一个$4 的客体时，默认类型为$3。

5.6.3 域转换类型宏

这种类型的宏主要定义了各种类型的域转换，如：

```
define(`domain_transition_pattern'，`
        allow $1 $2：file { getattr read execute };
        allow $1 $3：process transition;
        dontaudit $1 $3：process { noatsecure siginh rlimitinh };')
```

其中$1 表示当前域类型，$2 表示可执行文件类型，$3 表示新的域类型。这个宏的作用是$1 进程可以执行$2 文件，并允许域转换到$3，不记录尝试继承父进程资源、信号的行为。

5.6.4 进程通信类型宏

这种类型的宏主要用于进程间通信许可。

```
define(`stream_connect_pattern'，`
```

allow $1 $2：dir search_dir_perms;

allow $1 $3：sock_file { getattr write };

allow $1 $4：unix_stream_socket connectto;')

其中$1 表示当前域类型，$2 表示容器类型，$3 表示连接$4 的 socket 类型文件，$4 表示字节流 socket 的一个类型。这个宏的作用是$1 域通过字节流 socket 可以连接到$4 类型。

define(`dgram_send_pattern'，`

allow $1 $2：dir search_dir_perms;

allow $1 $3：sock_file { getattr write };

allow $1 $4：unix_dgram_socket sendto;')

这个宏和上面类似，区别在于这个使用数据报类型 socket。

5.6.5 生成类型宏

这种类型的宏是为了生成上下文、用户、安全级等，如：

define(`gen_context'，`

$1`'ifdef(`enable_mls'，`：$2')`'

ifdef(`enable_mcs'，`

＃如果$3 和空相同的话就返回空，不同返回$3。

：s0`'ifelse(`$3',,，`：$3')'

)')dnl

其中$1 表示安全上下文的三元素(用户、角色、类型或域)，$2 敏感级，$3 表示范畴。这个宏表示如果定义了 mls，安全上下文为$1：$2，如果定义了 mcs，并且$3 不为空的话，安全上下文为$1：s0：$3。

5.6.6 mls_mcs 类型宏

这种类型的宏主要定义关于 mls 和 mcs 的内容。如：

define(`mls_systemlow'，`s0')

define(`mls_systemhigh'，`s`'decr(mls_num_sens)：c0.c`'decr(mls_num_cats)')

定义 mls_systemlow 为 s0，mls_num_sens 对应于在 build.conf 文件中定义的安全级别，mls_systemhigh 就对应于 s(mls_num_sens-1) ：c0.c(mls_num_cats-1)。

5.6.7 模板类型宏

这种类型的宏主要用于加载模块，定义模板等，都比较复杂，以后再具体介绍。

5.6.8 宏调用

宏 apache_domain 的使用方法如下：

apache_domain(sys)

宏 apache_domain 调用时，其内容中的$1 将用 sys 替换，例如：

type httpd_$1_htaccess_t，file_type，rootfile;

上面的语句将变为：

type httpd_sys_htaccess_t，file_type，rootfile;

其他常用的策略语言也会被定义为宏接口方便调用，熟悉这些语句后就可以对我们的默认策略规则进行修改，也可编写一个新的策略模块加入到我们的策略规则集中。

5.7 添加策略模块

我们可以通过安全审计员使用 audit2allow –a –M policy_module 命令来产生一个策略模块，这个策略模块是针对审计中规则不允许的操作产生允许的规则，你必须确认你确实要允许这样的操作，因为这样的规则可能会带来安全隐患。之后安全管理员可以通过 semodule –i policy_module.pp 将这个策略模块载入到系统中。

同样我们也可以自己编写一个策略模块，一个策略模块主要有私有策略文件（.te），外部接口文件（.if），标记策略文件（.fc）三个文件组成，编译生成（.pp）文件，再通过 semodule -i *.pp 加入到整个策略中。也可和其他模块一起预编译生成（.conf）文件，再通过 checkpolicy 生成二进制策略文件再加入到内核中。

5.8 布尔变量

布尔变量使得部分 SELinux 策略在运行时可被改变，不需要任何 SELinux 策略书写知识。这允许变更，如允许服务不用重新加载或重新编辑 SELinux 策略而访问 NFS 文件系统。

5.8.1 布尔变量列表

对于布尔变量列表中每一个的解释以及是 on 或是 off，请作为 root 用户运行命令 semanage boolean -l。下例没有列出所有布尔变量：

```
#  /usr/sbin/semanage  boolean -l
SELinux  boolean Description


ftp_home_dir  -> off    Allow ftp  to  read and write  files  in the user home
directories
```

```
xen_use_nfs    -> off      Allow xen to    manage nfs files
xguest_connect_network-> on      Allow xguest to    configure    Network    Manager
```

列给出了布尔变量的名称。Description 列给出了布尔变量是 on 或 or，以及它们的功能。

下例中，布尔变量 ftp_home_dir 是 off，防止 FTP 后台程序(vsftpd)在用户的主目录读写文件：

```
ftp_home_dir  -> off      Allow ftp    to    read and write    files    in the user home directories
```

getsebool –a 命令列出布尔变量，无论是 on 还是 off，但没有给出每一个的详细描述。下例没有列出所有布尔变量：

```
$    /usr/sbin/getsebool -a
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core -->    on
```

运行 getsebool *boolean-name* 命令，只列出 *boolean-name* 布尔变量的状态：

```
$    /usr/sbin/getsebool allow_console_login
allow_console_login --> off
```

使用单倍行距列表列出多个布尔变量：

```
$ /usr/sbin/getsebool allow_console_login allow_cvs_read_shadow    allow_daemons_dump_core
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core -->    on
```

5.8.2 配置布尔变量

setsebool *boolean-name x* 命令设置布尔变量为 on 或 off，其中 *boolean-name* 是一个布尔变量名，若 *x* 是 on，表示启用该布尔变量，若为 off，表示关闭该布尔变量。

下例描述了如何配置 httpd_can_network_connect_db 布尔变量。

1） 默认情况下，httpd_can_network_connect_db 布尔变量取值为 off，防止 Apache HTTP 服务器脚本和模块连接到数据库服务器：

```
$    /usr/sbin/getsebool httpd_can_network_connect_db
httpd_can_network_connect_db --> off
```

2） 要临时启用 Apache HTTP 服务器脚本和模块连接到数据库服务器，请作为 root 用户运行 setsebool httpd_can_network_connect_db on 命令：

3） 使用 getsebool httpd_can_network_connect_db 命令验证布尔变量是否取值为 on：

```
$  /usr/sbin/getsebool httpd_can_network_connect_db
httpd_can_network_connect_db -->  on
```

这使 Apache HTT 服务器脚本和模块可以连接到数据库服务器。

4） 这种变更在重启后不会被保持。要在重启后保持永久变更，请作为 root 用户运行 setsebool -P *boolean-name* on 命令。

```
#  /usr/sbin/setsebool -P httpd_can_network_connect_db on
```

5） 要临时恢复到默认行为，请作为 root 用户运行 setsebool httpd_can_network_connect_db off 命令。对重启后能保持的变更，请运行 setsebool -P httpd_can_network_connect_db off 命令。

5.8.3 NFS 与 CIFS 的布尔变量

默认情况下， NFS 在客户端的挂载 被标记为 NFS 文件系统策略中定义的默认上下文。通用策略中，这种默认的上下文使用 nfs_t 类型。同样，默认情况下，客户端挂载的 Samba 共享被标记为策略定义的默认上下文。通用策略中，这种默认的上下文使用 cifs_t 类型。

根据策略配置，服务不能读取标记为 nfs_t 或 cifs_t 类型的文件。这可以防止文件系统被标记为正在安装的类型，然后被其他服务读写。布尔变量通过设置为 on 或 off 或来控制哪些服务允许访问 nfs_t 和 cifs_t 类型。

setsebool 和 semanage 命令必须以 root 用户运行。setsebool -P 命令生成永久变更。若您不想在重启后维持永久变更，请不要使用-P 选项：

Apache HTTP 服务器

允许访问 NFS 文件系统（文件被标记为 **nfs_t** 类型）：

```
/usr/sbin/setsebool -P  httpd_use_nfs  on
```

允许访问 Samba 文件系统（文件被标记为 **cifs_t** 类型）：

```
/usr/sbin/setsebool  -P httpd_use_cifs on
```

Samba

输出 NFS 文件系统：

```
/usr/sbin/setsebool -P  samba_share_nfs  on
```

FTP (vsftpd)

允许访问 NFS 文件系统：

```
/usr/sbin/setsebool -P allow_ftpd_use_nfs  on
```

允许访问 Samba 文件系统：

```
/usr/sbin/setsebool -P allow_ftpd_use_cifs  on
```

其他服务

关于其他服务的与 NFS 相关的布尔变量列表：

```
/usr/sbin/semanage  boolean -l | grep nfs
```

关于其他服务的与 Samba 相关的布尔变量列表：

```
/usr/sbin/semanage  boolean -l | grep cifs
```

5.9 文件系统标记

默认情况下，当安装支持额外属性的文件系统时，每个文件的上下文是从文件的 *security.selinux* 扩展属性获得的。基于文件系统类型，从策略配置中为文件系统中不支持扩展属性的文件分配一个唯一的默认安全上下文。

使用 mount -o context 命令重载已有扩展属性或为不支持扩展属性的文件系统指定一个不同的默认上下文。若您不相信文件系统提供正确属性时，这是很有用的。例如，多系统中使用的可删除媒体。mount -o context 命令也可用于支持标记不支持扩展属性的文件系统，如文件分配表（FAT）或 NFS 文件系统。用 context 指定的上下文不写入磁盘：当没有 context 选项（若该文件系统在第一个位置有扩展属性）挂载文件系统时，原始上下文被保留并被查看。

5.9.1 指定文件系统上下文

当安装所希望的文件系统时，若要安装具有指定上下文的文件系统并重载已有上下文，或为不支持扩展属性的文件系统指定一个不同的默认上下文，请作为 root 用户使用 mount -o context=*SELinux_user:role:type:level* 命令。上下文变更不写入磁盘。默认情况下，客户端的 NFS 文件系统挂载被标记为 NFS 文件系统策略定义的默认上下文。通用策略中，这种默认上下文使用 nfs_t 类型。当没有额外的安装选项时，这可以防止通过其他服务共享 NFS 文件系统，如 Apache HTTP 服务器。下例挂载一个 NFS 文件系统使得它能够通过 Apache HTTP 服务器被共享：

```
#  mount  server:/export  /local/mount/point -o\
context="system_u:object_r:httpd_sys_content_t:s0"
```

这个文件系统中新创建的文件和目录看上去具有用-o context 指定的 SELinux 上下文，然而，既然这些情况下上下文变更不写入磁盘，所以如果 context 选项用于下一次挂载，并且如果指定 相同的上下文。则仅仅保存用 context 选项指定的上下文。

类型增强是策略中使用的主要权限控制。对于多数情况，SELinux 用户和角色可以被忽略，因此，当重载有-o context 的 SELinux 上下文时，请使用 SELinux system_u 用户

和 object_r 角色，并关注类型。若您不是在使用 MLS 策略或多类别安全，请使用 s0 层次。

 注意：当用 context 选项挂载文件系统时，上下文变更（用户或进程引起的）是禁止的。例如，在用 context 选项挂载的文件系统上运行 chcon 命令，将导致 Operation not supported 错误。

5.9.2 变更默认上下文

在支持扩展属性的文件系统上，当访问磁盘中缺少 SELinux 上下文的文件时，当做其有 SELinux 策略定义的默认上下文。

在通用策略中，默认上下文使用 file_t 类型。如果希望使用一种不同的默认上下文，请使用 defcontext 选项挂载文件系统。

下例挂载一个新创建的文件系统(在/dev/sda2 上)到新创建的/test/目录。假设/etc/selinux/targeted/contexts/files/ 中没有为/test/目录定义上下文的规则：

```
# mount /dev/sda2 /test/ -o defcontext="system_u:object_r:samba_share_t:s0"
```

本例中：

- defcontext 选项定义了 system_u:object_r:samba_share_t:s0 是"无标记文件的默认安全上下文"[6]。
- 当挂载时，文件系统的根目录(/test/)被当做已被标记为 defcontext（该标记没有存储在磁盘上）指定的上下文。这影响到在 /test/下创建的文件的标记：新文件继承 samba_share_t 类型，这些标记都存储于磁盘中。
- 用 defcontext 选项挂载文件系统时在**/test/**下创建的文件保持它们的标记。

5.9.3 挂载 NFS 文件系统

默认情况下，挂载在客户端的 NFS 文件系统被标记为 NFS 文件系统策略定义的默认上下文。通用策略中，默认上下文使用 nfs_t 类型。根据策略配置，如 Apache HTTP 服务器 和 MySQL 等服务不能读取标记为 nfs_t 类型的文件。这可以防止标记为这种类型的文件系统被挂载然后被其他服务读和输出。

若您愿意挂载 NFS 文件系统，并且用另一个服务读或输出那个文件系统，请在挂载时使用 context 选项来重载 nfs_t 类型。请使用下面的上下文选项来挂载 NFS 文件系统，从而使 NFS 文件系统可以通过 Apache HTTP 服务器被共享。

```
mount server:/export /local/mount/point -o\
context="system_u:object_r:httpd_sys_content_t:s0"
```

因为这些情况下，不会将上下文变更写入磁盘，所以若下一次挂载时使用了 context 选项并指定了相同的上下文，则只有用 context 选项指定的上下文被保持。

作为用 context 选项挂载文件系统可替代的方法，布尔变量可以设置为 on 来允许服务

访问标记为nfs_t类型的服务系统。更多关于配置布尔变量允许服务访问nfs_t类型的方法，请参阅 *“NFS 与 CIFS 布尔变量”*

5.9.4 多 NFS 挂载

当从相同 NFS 输出挂载多个文件系统时，试图为每次挂载都重载不同的 SELinux 上下文，将引起后续加载命令的失败。下例中，NFS 服务器有一个简单的输出/export，/export 有两个子目录 web/ 和 database/。下面的命令试图从一个 NFS 输出试图挂载两次，并重载每个上下文：

```
# mount server:/export/web /local/web -o\
context="system_u:object_r:httpd_sys_content_t:s0"

# mount server:/export/database /local/database -o\
context="system_u:object_r:mysqld_db_t:s0"
```

第二次挂载命令失败，下面是记录到/var/log/messages 日志中的消息：

```
kernel: SELinux: mount invalid. Same superblock, different security settings for (dev 0:15, type nfs)
```

要从一个 NFS 输出挂载多个文件系统，并且每次挂载都有不同的上下文，请使用-o nosharecache,context 选项。下例从一个 NFS 输出中挂载多个文件系统，并且每次挂载使用不同的上下文（允许单个服务访问每个文件系统）。

```
# mount server:/export/web /local/web -o\
nosharecache,context="system_u:object_r:httpd_sys_content_t:s0"

# mount server:/export/database /local/database -o\
nosharecache,context="system_u:object_r:mysqld_db_t:s0"
```

本例中，server:/export/web 被挂载到/local/web/，所有文件都标记为 httpd_sys_content_t 类型，允许 Apache HTTP 服务器访问。server:/ export/database 被挂载到/local/database，允许 MySQL 访问。这些类型变更不被写入磁盘。

重要：选项允许您以不同上下文（例如，多次挂载到/export/web）多次挂载到一个输出的同一个子目录。请不要将一个输出以不同上下文多次挂载到同一子目录，因为这将引起重叠挂载，其中的文件在两个不同上下文下都是可访问的。

5.9.5 使上下文挂载持久

要令上下文挂载在重新挂载和重启时保持持久，请为文件系统在/etc/fstab 下增加入口，并且使用要求的上下文作为挂载选项。下例为 NFS 上下文挂载增加了一个到/etc/fstab

的入口。

```
server:/export    /local/mount/ nfs context="system_u:object_r:httpd_sys_content_t:s0" 0    0
```

5.10 文件安全上下文

这些章节描述了当复制、移动、归档文件与目录时，SELinux 上下文的变化。同样，它解释了如何在复制和归档时保持上下文。

5.10.1 复制文件及目录

当复制文件或目录时，若文件或目录不存在，则创建新文件或新目录。新文件或目录的上下文是基于默认标记规则的，而不是基于原始文件或目录的上下文（除非使用了用于保持原有标记的选项）。

例如：用户主目录中创建的文件被标记为 user_home_**t** 类型。

```
$    touch file1
$    ls    -Z file1
-rw-rw-r--        user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

若一个文件被复制到另一目录，如/etc/，按照默认标记规则，为/etc/目录创建新文件。复制文件（无额外选项）不保持原始上下文。

```
$    ls    -Z file1
-rw-rw-r--        user1 group1 unconfined_u:object_r:user_home_t:s0 file1
#    cp file1 /etc/
$    ls    -Z /etc/file1
-rw-r--r--      root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

当 file1 被复制到/etc/时，若/etc/file1 不存在，则将其创建为新文件/etc/file1。如上例所示，按照默认标记规则，/etc/file1 被标记为 etc_t 类型。

当一个文件复制到一个已存在的文件时，保持已有文件的上下文，除非用户指定 cp 选项来保持原始文件的上下文，如--preserve=context. SELinux 策略可以阻止在复制过程中保持上下文。

不保持 SELinux 上下文的复制

当使用 cp 命令复制文件时，若没有给定任何选项，则从目标、父目录中继承类型：

```
$    touch file1
$    ls    -Z file1
-rw-rw-r--        user1 group1 unconfined_u:object_r:user_home_t:s0 file1
$    ls    -dZ /var/www/html/
drwxr-xr-x    root    root    system_u:object_r:httpd_sys_content_t:s0    /var/www/html/
```

```
#   cp file1   /var/www/html/
$   ls   -Z /var/www/html/file1
-rw-r--r--   root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
```

本例中，file1 在用户的主目录中创建，被标记为 user_home_t 类型。/var/www/html/目录被标记为 httpd_sys_content_t 类型，如 ls -dZ /var/www/html/命令所示。当 file1 被复制到/var/www/html/时，它继承了 httpd_sys_content_t 类型，如 ls -Z /var/www/html/file1 命令所示。

复制时保持 SELinux 上下文

使用 cp --preserve=context 命令在复制时保持上下文：

```
$   touch file1
$   ls   -Z file1
-rw-rw-r--      user1 group1 unconfined_u:object_r:user_home_t:s0 file1
$   ls   -dZ /var/www/html/
drwxr-xr-x   root   root   system_u:object_r:httpd_sys_content_t:s0   /var/www/html/
#   cp --preserve=context file1   /var/www/html/
$   ls   -Z /var/www/html/file1
-rw-r--r--   root   root   unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
```

本例中，file1 在用户的主目录创建，被标记为 user_home_t 类型。/var/www/html/目录被标记为 httpd_sys_content_t 类型，如 ls -dZ /var/www/html/命令所示。使用 --preserve=context 选项在复制操作时保持 SELinux 上下文。如 ls -Z /var/www/html/file1 命令所示，当复制文件到/var/www/ html/时，可以保持类型 file1 user_home_t。

复制与变更上下文

使用 cp -Z 命令来变更复制上下文的目的地。下例是在用户的主目录中执行的：

```
$   touch file1
$   cp -Z system_u:object_r:samba_share_t:s0   file1 file2
$   ls   -Z file1 file2
-rw-rw-r--      user1 group1 unconfined_u:object_r:user_home_t:s0 file1
-rw-rw-r--      user1 group1 system_u:object_r:samba_share_t:s0 file2
$   rm   file1 file2
```

本例中，上下文是用-Z 选项定义的。没有-Z 选项，file2 将被标记为 unconfined_u:object_r:user_home_t 上下文。

复制一个文件覆盖已有文件

当复制一个文件覆盖已有文件时，已有文件的上下文被保持（除非使用了保持上下文

的选项）。例如：

```
#  touch /etc/file1
#  ls  -Z /etc/file1
-rw-r--r--     root root unconfined_u:object_r:etc_t:s0   /etc/file1
#  touch /tmp/file2
#  ls  -Z /tmp/file2
-rw-r--r--   root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
#  cp /tmp/file2   /etc/file1
#  ls  -Z /etc/file1
-rw-r--r--     root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

本例中创建了两个文件，标记为 etc_t 类型的/etc/file1 和标记为 user_tmp_t 类型的/tmp/file2。cp /tmp/file2 /etc/file1 命令用 file2 重写 file1。复制后，ls -Z /etc/file1 命令表明 file1 已被标记为 etc_t 类型，而不是取代了/etc/file1 的/tmp/file2 的 user_tmp_t 类型。

重要：复制文件和目录，而不是移动他们。这有助于确保其被标记为正确的 SELinux 上下文。不正确的 SELinux 上下文可以防止进程访问这样的文件和目录。

5.10.2 移动文件和目录

当移动文件和目录时，文件和目录保持它们当前的 SELinux 上下文。多数情况下，这对于它们移动的目的地是不正确的。下例描述了如何将文件从用户的主目录移动到 Apache HTTP 服务器使用的/var/www/html/目录。因为文件被移动了，文件不再继承正确的 SELinux 上下文：

1） 运行无参数 cd 命令进入您的主目录。若您已在主目录中，请运行 touch file1 命令来创建一个文件。该文件被标记为 user_home_t 类型：

```
$  ls  -Z file1
-rw-rw-r--     user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

2） 运行 ls -dZ /var/www/html/ 命令来查看/var/www/html/目录的 SELinux 上下文：

```
$  ls  -dZ /var/www/html/
drwxr-xr-x  root  root  system_u:object_r:httpd_sys_content_t:s0  /var/www/html/
```

默认情况下，/var/www/html/目录被标记为 httpd_sys_content_t 类型。/var/www/html/目录下创建的文件和目录都继承这个类型，同样，它们也被标记为这个类型：

3） 作为 root 用户，运行 mv file1 /var/www/html/命令将 file1 移动到/var/www/html/目录。文件被移动后，保持现有的 user_home_t 类型：

```
#  mv  file1  /var/www/html/
#  ls  -Z /var/www/html/file1
-rw-rw-r--      user1 group1 unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
```

默认情况下，Apache HTTP 服务器不能读取标记为 user_home_t 类型的文件。若包含一个网页的所有文件都被标记为 user_home_t 类型，或者另外一种 Apache HTTP 服务器不能读取的类型，则无权限通过 Firefox 或基于文本的 Web 浏览器访问它们。

 重要：使用 mv 命令移动文件和目录可能会导致错误的 SELinux 上下文，防止如 Apache HTTP 服务器和 Samba 的进程访问这样的文件和目录。

5.10.3 检查默认的 SELinux 上下文

使用/usr/sbin/matchpathcon 命令检查文件和目录的 SELinux 上下文是否正确。根据 matchpathcon(8)手册页："matchpathcon 查询系统策略并输出与文件路径相关的默认安全上下文。

下例描述了使用/usr/sbin/matchpathcon 命令验证/var/www/html/目录中的文件都被正确标记：

1） 作为 root 用户，运行 touch /var/www/html/file{1,2,3}命令创建 3 个文件(file1、file2 和 file3)。这些文件从/var/www/html/目录继承了 httpd_sys_content_t 类型：

```
#  touch /var/www/html/file{1,2,3}
#  ls  -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2） 作为 root 用户，运行 chcon -t samba_share_t /var/www/html/file1 命令将 file1 的类型改变为 samba_share_t。注意：Apache HTTP 服务器不能读取标记为 samba_share_t 的文件或目录。

3） /usr/sbin/matchpathcon -V 选项比较当前 SELinux 上下文和 SELinux 策略中正确的、默认上下文。运行/usr/sbin/matchpathcon -V /var/www/html/*命令来检查/var/www/html/目录中的所有文件：

```
$  /usr/sbin/matchpathcon -V  /var/www/html/*
/var/www/html/file1 has context unconfined_u:object_r:samba_share_t:s0,  should be
system_u:object_r:httpd_sys_content_t:s0
/var/www/html/file2 verified.
/var/www/html/file3 verified.
```

下面/usr/sbin/matchpathcon 命令的输出解释了 file1 被标记为了 samba_share_t 类型，但应被标记为 httpd_sys_content_t 类型。

```
/var/www/html/file1 has context unconfined_u:object_r:samba_share_t:s0,  should be
system_u:object_r:httpd_sys_content_t:s0
```

要解决标记问题并允许 Apache HTTP 服务器访问 file1，作为 root 用户，请运行 /sbin/restorecon -v /var/www/html/file1 命令：

```
#  /sbin/restorecon  -v /var/www/html/file1
restorecon reset  /var/www/html/file1 context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

5.10.4 用 tar 归档文件

tar 不保持默认的扩展属性。因为 SELinux 上下文被存储在扩展属性中，当归档文件时将丢失上下文。使用 tar –selinux 来创建保持上下文的归档。若一个 Tar 归档包含没有扩展属性的文件，或您要扩展属性以便与系统的默认设置相匹配，请通过 /sbin/restorecon:tar 运行归档。

```
$  tar -xvf archive.tar | /sbin/restorecon -f -
```

 注意：根据目录，您可能需要作为 root 用户来运行/sbin/restorecon 命令。

下例描述了如何创建一个保持 SELinux 上下文的 Tar 归档：

1） 作为 root 用户，运行 touch /var/www/html/file{1,2,3}命令创建 3 个文件(file1、file2 和 file3)。这些文件从/var/www/html/目录继承 httpd_sys_content_t 类型：

```
#  touch /var/www/html/file{1,2,3}
#  ls  -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2） 运行 cd /var/www/html/命令进入到/var/www/html/目录。一旦在此目录中，作为 root 用户，运行 tar --selinux -cf test.tar file{1,2,3}命令创建一个名为 test.tar 的 Tar 归档。

3） 作为 root 用户，运行 mkdir /test 命令创建一个新目录，然后运行 chmod 777 /test/命令允许所有用户访问/test/目录。

4） 运行 cp /var/www/html/test.tar /test/命令将 test.tar 文件复制到/test/目录。

5） 运行 cd /test/命令进入/test/目录。一旦在此目录中，请运行 tar -xvf test.tar 命令

来提取 Tar 归档。

6） 运行 ls -lZ /test/命令查看 SELinux 上下文。httpd_sys_content_t 类型被保持，而不是变为 default_t，使--selinux 没被用到：

```
$ ls -lZ /test/
-rw-r--r-- user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r-- user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r-- user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file3
-rw-r--r-- user1 group1 unconfined_u:object_r:default_t:s0 test.tar
```

7） 若不再需要/test/目录，作为 root 用户，请运行 rm -ri /test/命令删除之，同时也删除其中的所有文件。

更多关于 tar 的信息，如保持所有扩展属性的--xattrs 选项，请参阅 tar(1)手册页。

5.10.5 用 star 归档文件

star 不保持默认的扩展属性。因为 SELinux 上下文被存储在扩展属性中，当归档文件时将丢失上下文。使用 star -xattr -H=exustar 来创建保持上下文的归档。默认情况下不安装 *Star* 包。要安装 star，请作为 root 用户运行 yum install star 命令。

下例描述了如何创建一个保持 SELinux 上下文的 Star 归档：

1） 作为 root 用户，运行 touch /var/www/html/file{1,2,3}命令创建 3 个文件(file1、file2 和 file3)。这些文件从/var/www/html/目录继承了 httpd_sys_content_t 类型：

```
# touch /var/www/html/file{1,2,3}
# ls -Z /var/www/html/
-rw-r--r-- root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r-- root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r-- root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2） 运行 cd /var/www/html/命令进入/var/www/html/目录。一旦进入该目录，作为 root 用户，请运行 star -xattr -H=exustar -c -f=test.star file{1,2,3}命令创建一个名为 test.star 的 Star 归档：

```
# star -xattr -H=exustar -c -f=test.star file{1,2,3}
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
```

3） 作为 root 用户，运行 mkdir /test 命令创建一个新目录，然后运行 chmod 777 /test/命令，从而允许所有用户可访问/test/目录。

4） 运行 cp /var/www/html/test.star /test/命令将 test.star 文件复制到/test/目录。

5） 运行 cd /test/命令进入/test/目录。一旦进入此目录，请运行 star -x -f=test.star 命

令来提取 Star 归档。

```
$ star -x -f=test.star
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
```

6） 运行 ls -lZ /test/命令查看 SELinux 上下文。httpd_sys_content_t 类型被保持，而不是变更到 default_t，使--selinux 没有被用到：

```
$ ls -lZ /test/
-rw-r--r-- user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r-- user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r-- user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file3
-rw-r--r-- user1 group1 unconfined_u:object_r:default_t:s0 test.star
```

7） 若不再需要/test/目录，作为 root 用户，请运行 rm -ri /test/命令删除之，同时删除其中的所有文件。

8） 若不再需要 star， 作为 root 用户，请运行 yum remove star 命令来删除包。

更多关于 star 的信息，请参阅 star(1)手册页。

5.11 信息收集工具

这些工具是提供格式化输出地命令行工具。它们很难作为命令行管道的一部分，但它们能快速提供收集到的和具有很好格式的信息。

avcstat

本命令提供了自启动后访问向量缓冲的简要输出。您可以通过指定几秒钟的时间间隔来实时观看统计信息。这样提供了自初始输出后更新的统计信息。使用的统计信息文件是/selinux/avc/cache_stats，您可以用-f /path/to/file 选项指定一个不同的缓冲文件。

```
[root@localhost ~]# avcstat
Lookups hits misses allocs reclaims frees
47517410 47504630 12780 12780 12176 12275
```

seinfo

本工具在描述策略分解时非常有用，如：类、类型、布尔变量、允许规则等的数量。seinfo 是一个命令行工具，它将 policy.conf 文件或二进制策略文件作为输入。

seinfo 的输出会随二进制文件和源文件的不同而不同。例如，策略源文件使用{ }括号将多个规则元素分组到一行中。属性也有相似的效果，一个属性扩展到一个或多个类型。因为这些被扩展了，且不再与二进制策略文件有关，它们在搜索结果中的返回值为 0。 然而，规则的数量极大的增加，因为每个使用括号的原行规则现在变成了很多不同的行。

一些项目不出现在二进制策略中。例如，neverallow 规则只在策略编译时而不是在运行时被检查；因为初始 SID 需要在启动时先于内核加载的策略，所以初始的 SID 不是二进制策略的一部分。

```
[root@localhost ]# seinfo
Statistics for policy file: /etc/selinux/targeted/policy/policy.24
Policy Version & Type: v.24 (binary, mls)
Classes: 77 Permissions: 229
Sensitivities: 1 Categories: 1024
Types: 3001 Attributes: 244
Users: 9 Roles: 13
Booleans:158 Cond. Expr.: 193
Allow: 262796 Neverallow: 0
Auditallow: 44 Dontaudit: 156710
Type_trans: 10760 Type_change: 38
Type_member: 44 Role allow: 20
Role_trans: 237 Range_trans: 2546
Constraints: 62 Validatetrans: 0
Initial SIDs: 27 Fs_use: 22
Genfscon: 82 Portcon: 373
Netifcon: 0 Nodecon: 0
Permissives: 22 Polcap: 2
```

seinfo 命令也能列出域属性类型的数量，估计出不同限制进程的数量：

```
# seinfo -adomain -x | wc -l
550
```

并非所有域的类型都是限制的。要知道非限制域的数量，请使用 unconfined_domain 属性：

```
# seinfo -aunconfined_domain_type -x | wc -l
52
```

许可域可以用--permissive 选项计数。

```
# seinfo --permissive -x | wc -l
31
```

从上述命令中删除| wc -l 选项来查看完整列表。

sesearch

您可以使用 sesearch 命令来搜索策略中的特定类型。您可以搜索策略源文件或二进制

文件。例如：

```
[scott@localhost ~]$ sesearch --role_allow -t httpd_sys_content_t \/etc/selinux/targeted/
policy/policy.24
Found 20 role allow rules:
allow system_r sysadm_r;
allow sysadm_r system_r;
allow sysadm_r staff_r;
allow sysadm_r user_r;
allow system_r git_shell_r;
allow system_r guest_r;
allow logadm_r system_r;
allow system_r logadm_r;
allow system_r nx_server_r;
allow system_r staff_r;
allow staff_r logadm_r;
allow staff_r sysadm_r;
allow staff_r unconfined_r;
allow staff_r webadm_r;
allow unconfined_r system_r;
allow system_r unconfined_r;
allow system_r user_r;
allow webadm_r system_r;
allow system_r webadm_r;
allow system_r xguest_r;
```

sesearch 命令能提供 *allow* 规则的数量：

```
# sesearch --allow | wc -l
262798
```

dontaudit 规则的数量是:

```
# sesearch --dontaudit | wc -l
156712
```

6. MLS 管理

MLS 策略实现了基于 BLP 模型的多级安全策略。MLS 策略下进程主体的安全标记包含一个当前安全级以及一个安全级调整范围，客体的安全标记包含一个安全级。每个安全

级由安全级别、安全类别组成：安全级别是从 0 到 15 的整数；安全类别则是一个 256 个元素组成的集合的子集；同时系统中定义两个特殊的安全标记：SystemLow 和 SystemHight。安全级中可不包含安全类别集，表示安全类别集为空。

7.1 MLS 支配关系

MLS 策略定义安全级的支配关系如下：

1、若安全级 label1 的安全级别大于等于安全级 label2 的安全级别，并且安全级 label1 的安全类别包含安全级 label2 的安全类别，则称 label1 支配 label2，label2 被 label1 支配；

2、若安全级 label1 为 high，则不论安全级 label2 的值为多少，均有 label1 支配 label2；

3、若安全级 label1 为 low，则不论安全级 label2 的值为多少，均有 label1 被 label2 支配；

4、若安全级 label1 为 equal，则不论安全级 label2 的值为多少，均有 label1 支配 label2 且 label1 被 label2 支配；

7.2 MLS 访问控制规则

MLS 策略的访问控制判断规则如下：

1、“简单安全”规则，即在进程 P 对客体 O 进行读访问时，只有满足进程 P 的当前安全级支配 O 的安全级时，访问才被允许；

2、“同级写”规则，即在进程 P 对客体 O 进行写访问时，只有满足进程 P 的当前安全级与被客体 O 的安全级相互支配时，访问才被允许。

安全级调整规则：进程的安全级中包含有安全级调整范围（rangelow-rangehigh），当进程对客体进行读写访问时，进程的当前安全级以及安全级调整范围可以根据如下规则调整：

读调整：当进程 P 对客体 O 进行读访问时，若进程 P 的当前安全级不支配 O 的安全级，但进程 P 的 rangehigh 安全级支配 O 的安全级，则调整进程 P 的当前安全级和 rangelow 的安全类别集等于客体的安全类别集与 rangelow 的安全类别集的并集，安全级别等于客体 O 的安全级别与 rangelow 的安全级别的最大值，允许读访问该客体 O，否则拒绝；如果进程主体 P 的当前安全级支配客体 O 的安全级，则允许访问，并且如果进程主体的 rangelow 不支配客体 O 的安全级，则调整进程的 rangelow，ranglelow 的安全类别集等于客体 O 的安全类别集与 rangelow 的安全类别集的并集，ranglelow 的安全级别等于客体 O 的安全级别与 rangelow 的安全级别的最大值；

写调整：当进程 P 对客体 O 进行写访问时，要求同级写。即如果进程 P 的 rangehigh 支配客体 O 的安全级，并且客体 O 的安全级支配进程的 rangelow，则调整进程 P 的当前安全级和 rangehigh 为客体 O 的安全级，允许写访问该客体，否则拒绝。

7. sVirt

sVirt 是内置在中标麒麟可信操作系统 V6.0 中集成了 SELinux 和虚拟化的一种技术。使用虚拟机时，sVirt 应用强制访问控制 (MAC) 技术来增强安全性。集成这些技术的主要原因就是要增强安全性，并加固系统，防止管理程序中的缺陷可能用作针对主机或其他虚拟机的攻击媒介。

本章介绍 sVirt 与虚拟技术是如何集成到中标麒麟可信操作系统 V6.0 中的。

非虚拟化环境

在非虚拟化环境中，主机物理上相互分离，每个主机都有一个独立的环境，由 Web 服务器或 DNS 服务器等服务组成。这些服务直接与他们自己的用户空间、主机内核和物理主机进行通讯，直接向网络提供它们的服务。

图 2-4 描绘一个非虚拟化的环境：

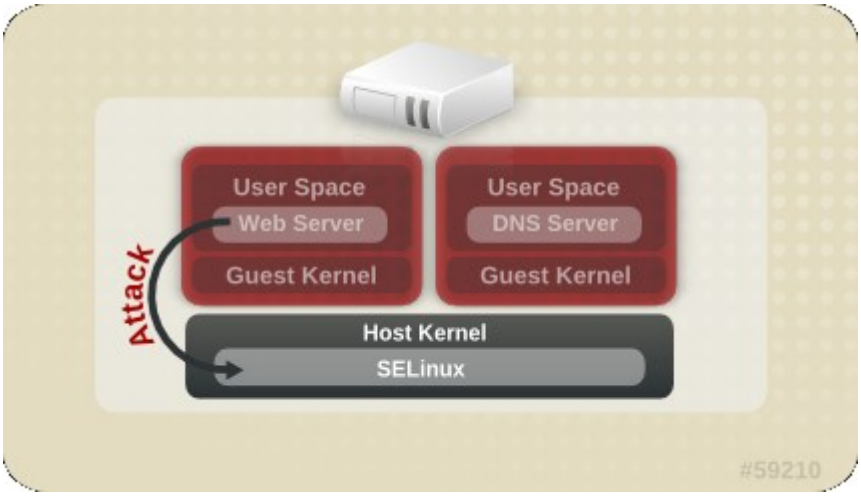


图 2-4 非虚拟化环境

虚拟化环境

在虚拟化环境中，多个操作系统可封装（作为“guests”）在一个主机内核和物理主机中。图 2-5 描绘一个虚拟化环境：

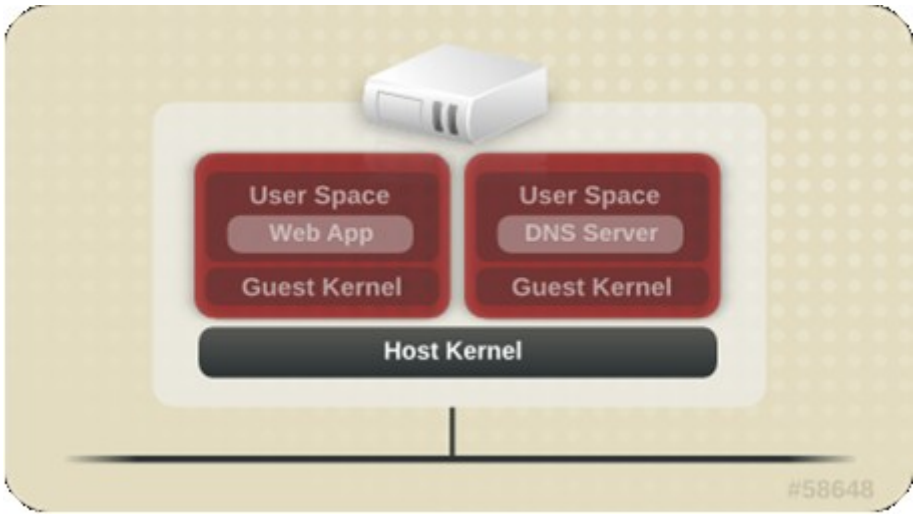


图 2-5 虚拟化环境

8.1 安全与虚拟化

服务并未虚拟化时，机器在物理上是分离的。受影响的机器通常包含任何利用行为，这些行为具有明显的网络攻击异常。当服务组合到虚拟化环境中时，系统中会出现另外的缺陷。如果管理程序中有一个客户实例可利用的缺陷，此客户可能不仅可以攻击主机，也可以攻击运行在那个主机上的其他客户。这不是假设，管理程序中已经存在缺陷。这些攻击可延伸的范围超过已受攻击的用户，还可能使其他客户面临攻击。

sVirt 尽量隔离客户，限制他们在利用时启动进一步攻击。图 2-6 中描述了这种情况，其中攻击无法摆脱虚拟机，也不能扩展到另一个主机程序。

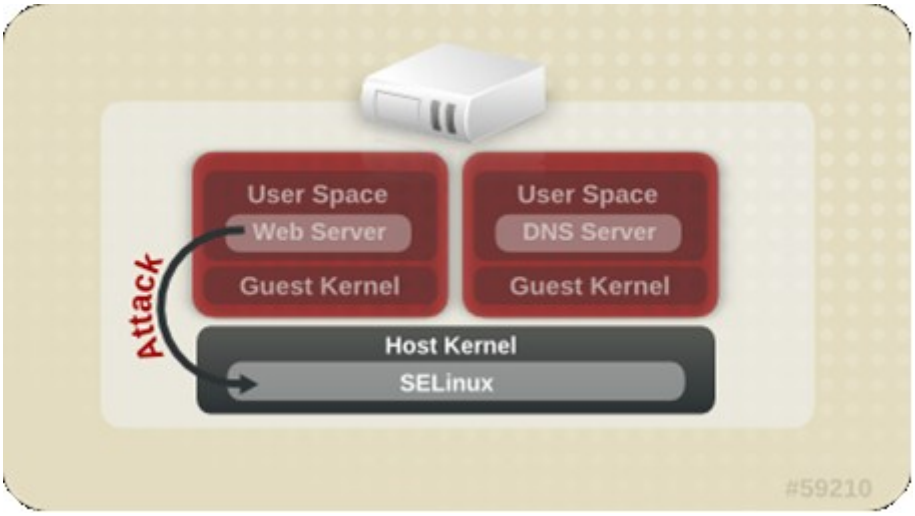


图 2-6 虚拟机攻击

SELinux 实施强制访问控制(MAC)过程中引入一种可插入的虚拟化实例安全框架。sVirt 框架允许唯一标记客户及其资源。标记后，可以应用规则，拒绝在不同客户之间进行访问。

8.2 sVirt 标记

如同 SELinux 保护下的其他服务，sVirt 使用基于进程的机制和约束，从而在用户程序之上提供额外的安全层。通常使用下，您甚至应该不会注意到 sVirt 在后台工作。本节描述 sVirt 的标记特性。

如下面输出所示，使用 sVirt 时，每个虚拟机（VM）进程都被标记，并且使用动态生成的级别来运行。每个进程使用不同级别来与其他 VM 进行隔离。

```
#  ps -eZ | grep  qemu
system_u:system_r:svirt_t:s0:c87,c520 27950 ?    00:00:17  qemu-kvm
system_u:system_r:svirt_t:s0:c639,c757 27989 ? 00:00:06  qemu-system-x86
```

为匹配进程，实际的磁盘镜像自动标记，如以下输出所示：

```
#  ls -lZ /var/lib/libvirt/images/*
system_u:object_r:svirt_image_t:s0:c87,c520     image1
```

表 列出了使用 sVirt 时可分配的不同标记：

表 2-7 sVirt 标记

类型	SELinux	描述
虚拟机进程	system_u:system_r:svirt_t:MCS1	MCS1 是随机选择的 MCS 域. 目前支持近 500,000 个标记.
虚拟机镜像	system_u:object_r:svirt_image_t:	MOCnSly1 具有相同 MCS 域的 svirt_t 进程 不能读/写 这些镜像文件和设备.
虚拟机共享读/写内容	system_u:object_r:svirt_image_t:	sA0ll 允许 svirt_t 进程写入 svirt_image_t:s0 文件和设备.
虚拟机共享只读内容	system_u:object_r:svirt_content_	t: Asl0l svirt_t 进程可以读带有此标记的文件/设备。

类型	SELinux	描述
虚拟机镜像	system_u:object_r:virt_content_t:	镜像存在时使用的默认标记 。不允许任何 *svirt_t* 虚拟进程读带有此标记的文件/设备。

使用 sVirt 时，也可执行静态标记。静态标记允许管理员选择用于虚拟机的特定标记，包括 MCS/MLS 域。运行静态标记虚拟机的管理员负责设置镜像文件上的正确标记。虚拟机始终带这个标记启动，且 sVirt 系统永远不会修改静态标记虚拟机的标记。这允许 sVirt 组件在 MLS 环境中运行。您也可以根据需求，在一个系统上使用不同安全级运行多个虚拟机。

8. 故障排除

本章介绍 SELinux 拒绝访问时所发生的情况；发生问题最常见的 3 个原因；找到正确标记信息的位置；分析 SELinux 拒绝信息；用 audit2allow 创建定制策略模块。

9.1 访问拒绝时所发生的情况

允许或不允许访问等 SELinux 决策将被缓存。这种缓存被称为访问向量缓存（AVC）。SELinux 拒绝访问时，将拒绝信息写入日志。这些拒绝信息也被称为"AVC 拒绝信息"，并被写入另一位置的日志，这取决于正在运行的后台程序。

表 2-8 拒绝信息日志

后台程序	日志位置
auditd on	/var/log/audit/audit.log
auditd off; rsyslogd on	/var/log/messages
setroubleshootd, rsyslogd, 且 auditd on	/var/log/audit/audit.log. 易于阅读的拒绝信息也发送到/var/log/messages

若您正在运行 X Window 系统，*setroubleshoot* 和 *setroubleshoot-server* 包已安装，并且 setroubleshootd 和 auditd 后台程序正在运行，SELinux 拒绝访问时显示一个警告：

图 2-7 SElinux 警告信息

单击'Show'，将显示 SELinux 拒绝访问原因的详细分析，以及对此访问可能的解决方案。若您未运行 X Window 系统，SELinux 拒绝访问时，影响则不太明显。例如，用户浏览您的网站可能收到与下面类似的错误：

```
Forbidden
You don't have permission to access file name on this server
```

对于这些情况，若 DAC 规则(标准 Linux 权限)允许访问，请针对"SELinux is preventing"和"denied"错误分别检查 var/log/ messages 和 /var/log/audit/audit.log。这可以通过以 Linux root 用户身份运行下面的命令来完成。

```
grep "SELinux is preventing" /var/log/messages grep "denied" /var/log/audit/audit.log
```

9.2 三种引起问题的最常见原因

以下章节介绍三种引起问题的最常见原因：标记问题、为服务配置布尔值和端口，扩展 SELinux 规则。

9.2.1 标记问题

在运行 SELinux 的系统上，所有进程和文件都被标记上包含安全信息的标记。这种信息称为 SELinux 上下文。若这些标记有错误，访问可能被拒绝。若应用程序未正确标记，它迁移到的进程可能没有正确标记，可能引起 SELinux 拒绝访问，还会拒绝能创建错误标记文件的进程。

引起标记问题的常见原因是为服务使用了非标准目录。如，管理员不使用用于网站的/var/www/html/，而是想使用/srv/ myweb/。中标麒麟 企业版 Linux 6 中, /srv/目录标记为 var_t 类型，用创建的文件和目录以及/srv/继承该类型。同样，新创建的顶层目录（如/myserver/）可以用 default_t 类型标记。SELinux 防止 Apache HTTP 服务器(httpd) 访问这两种类型。为允许访问，SELinux 必须确认 httpd 可访问 srv/myweb/中的文件。

```
# /usr/sbin/semanage fcontext -a -t httpd_sys_content_t \ "/srv/myweb(/.*)?"
```

这个 semanage 命令向 SELinux 文件上下文配置增加/srv/myweb/目录（它下面的文件和目录）上下文。semanage 命令不改变上下文。作为 Linux root 用户，运行 restorecon 命令来应用这些变更：

```
# /sbin/restorecon -R -v /srv/myweb
```

有关向文件上下文配置添加上下文的详细信息，请参阅第“永久更改： semanage fcontext”。

什么是正确上下文?

matchpathcon 命令检查文件路径的上下文，并将其与此路径的默认标记进行比较。下例演示在包含未正确标记文件的目录中使用 matchpathcon：

```
$ /usr/sbin/matchpathcon -V /var/www/html/*
/var/www/html/index.html has context unconfined_u:object_r:user_home_t:s0, should be system_u:object_r:httpd_sys_content_t:s0
/var/www/html/page1.html has context unconfined_u:object_r:user_home_t:s0, should be system_u:object_r:httpd_sys_content_t:s0
```

本例中，index.html 和 page1.html 文件都标记为 user_home_t 类型。此类型用于用户主目录下的文件。使用 mv 命令，从您的主目录移动文件，可能会导致文件标记为 user_home_t 类型。主目录外面应该不存在这个类型。使用 restorecon 命令将这些文件恢复为它们正确的类型：

```
# /sbin/restorecon -v /var/www/html/index.html
restorecon reset /var/www/html/index.html context unconfined_u:object_r:user_home_t:s0->system_u:object_r:httpd_sys_content_t:s0
```

要恢复目录下所有文件的上下文，使用-R 选项：

```
# /sbin/restorecon -R -v /var/www/html/
restorecon reset /var/www/html/page1.html context unconfined_u:object_r:samba_share_t:s0->system_u:object_r:httpd_sys_content_t:s0
restorecon reset /var/www/html/index.html context unconfined_u:object_r:samba_share_t:s0->system_u:object_r:httpd_sys_content_t:s0
```

有关 matchpathcon 的详细示例，请参阅“检查默认 SELinux 上下文”。

9.2.2 限制服务如何运行？

服务可以以多种方式运行。为此，您必须通知 SELinux 您运行服务的方式。若不了解这个类型，这可以通过布尔值允许在运行时更改部分 SELinux 策略来实现。这允许不用重新加载或重新编译 SELinux 策略服务进行更改，比如允许服务访问 NFS 文件系统。另外，在非默认端口号运行服务需要通过 semanage 命令更新策略配置。

例如：要允许 Apache HTTP 服务器与 MySQL 通讯，将 httpd_can_network_connect_db 布尔值设置为 on：

```
# /usr/sbin/setsebool -P httpd_can_network_connect_db on
```

如果某个特定服务的访问被拒绝，使用 getsebool 和 grep 命令来查看是否存在允许访

问的布尔值。例如，使用 getsebool -a | grep ftp 命令来查找与 FTP 相关的布尔值：

```
$ /usr/sbin/getsebool -a | grep ftp
allow_ftpd_anon_write --> off
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
```

要查看布尔变量列表及它们取值 on 还是 off 的情况，运行/usr/sbin/getsebool -a 命令。要查看布尔变量列表，每个变量的详细说明，以及它们取值为 on 还是 off 的情况，请作为 root 用户运行/usr/sbin/semanage boolean -l 命令。

端口号

根据策略配置，可能只允许服务在特定端口号上运行。若试图更改服务运行所在的端口号而不更改策略，可能会导致服务无法启动。例如：以 root 用户身份运行 semanage port -l | grep http 命令来列出与 http 相关的端口：

```
# /usr/sbin/semanage port -l | grep http
http_cache_port_t tcp 3128, 8080, 8118
http_cache_port_t udp 3130
http_port_t tcp 80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t tcp 5988
pegasus_https_port_t tcp 5989
```

http_port_t 端口类型定义了端口。Apache HTTP 服务器可以监听此例中的 TCP 端口：80、443、488、8008、8009 和 8443。若管理员配置 httpd.conf，使 httpd 监听 9876 端口 (Listen 9876)，但是并没有更新策略，使之对此加以反映，则 service httpd start 命令将会失败：

```
# /sbin/service httpd start
Starting httpd: (13)Permission denied: make_sock: could not bind to address [::]:9876
(13)Permission denied: make_sock: could not bind to address 0.0.0.0:9876
no listening sockets available, shutting down
Unable to open logs
[FAILED]
```

与下面相似的 SELinux 拒绝信息记录到/var/log/audit/audit.log：

```
type=AVC msg=audit(1225948455.061:294): avc: denied { name_bind } for
pid=4997 comm="httpd" src=9876 scontext=unconfined_u:system_r:httpd_t:s0
```

```
tcontext=system_u:object_r:port_t:s0 tclass=tcp_socket
```

要允许 httpd 监听一个 http_port_t 端口类型未列出的端口，运行 semanage port 命令向策略配置中添加一个端口：

```
#  /usr/sbin/semanage port -a -t http_port_t  -p tcp 9876
```

-a 选项添加了一条新记录，-t 选项定义了一个类型，-p 选项定义了一个协议。最后一个参数是要添加的端口号。

9.2.3 演变中的规则与损坏的应用程序

应用程序可能会损坏，引起 SELinux 拒绝访问。同样，SELinux 规则也在演变中，SELinux 可能没又发现正在以某种方式运行的一个应用程序，即使应用程序照常工作，这也可能引起 SELinux 拒绝访问。例如：若 PostgreSQL 发布了新版本，那么它执行的动作可能是当前策略以前还未发现的，这可能造成即使应该允许访问也会拒绝访问。

请对于这些情况，访问拒绝后，使用 audit2allow 来创建自定义策略模块来允许访问。有关 audit2allow 的详细信息，请参阅 “允许访问: audit2allow”。

9.3 修复问题

以下章节帮助解决故障修复问题。问题涵盖：检查 Linux 权限，SELinux 应用规则检查 Linux 权限；SELinux 拒绝访问，但未记录拒绝的可能原因；关于服务的手册页，包含了关于标记和布尔值的信息；许可域，允许一个进程运行在许可域中而不是整个系统；如何查找和浏览拒绝消息；分析拒绝消息；用 audit2allow 创建定制策略模块。

9.3.1 Linux 权限

拒绝访问时，检查标准 Linux 权限。如*概述*中所述，多数操作系统使用随意访问控制(DAC)系统来控制访问，允许用户控制他们所拥有的文件的权限。检查完 DAC 规则后才检查 SELinux 策略规则。如果 DAC 规则首先拒绝了访问，则不使用 SELinux 策略规则。

如果拒绝访问且未记录 SELinux 拒绝信息，使用 ls -l 命令来查看标准 Linux 权限：

```
$  ls -l  /var/www/html/index.html
-rw-r----- 1  root root 0  2009-05-07 11:06  index.html
```

本例中，root 用户和组拥有 index.html。root 用户具有读写权限(-rw)，根组成员具有读(-r-)权限。其他用户没有访问权限(---)。在默认情况下，这些权限不允许 httpd 读取该文件。要解决这个问题，使用 chown 命令更改所有者和组。此命令必须以 root 用户身份运行。

```
#  chown  apache:apache /var/www/html/index.html
```

这假定为默认配置，其中 httpd 以 Linux apache 用户身份运行。若您作为另一不同用户运行 httpd，请以 apache:apache 替换那个用户。

9.3.2 静默拒绝的可能原因

特定情况下，当 SELinux 拒绝访问时，AVC 拒绝消息可能不记入日志中。应用程序和系统库函数往往探查的访问多于执行任务所需要的访问。为维护最小权限，不在监听日志中填写 AVC 拒绝消息以进行无害应用程序探查，这种策略可以静默 AVC 拒绝消息，不必使用 dontaudit 规则允许一个权限。这些规则是标准策略中的常见规则。dontaudit 的不利影响是，虽然 SELinux 拒绝了访问，但是拒绝消息并不记入日志，使得故障修复变得困难。

要临时禁用 dontaudit 规则，允许记录所有拒绝消息，以 root 用户身份运行下面的命令：

```
/usr/sbin/semodule -DB
```

-D 选项禁用 dontaudit 规则；-B 选项重建策略。运行 semodule -DB 之后，尝试运行遇到权限问题的应用程序，并检查是否与应用相关的 SELinux 拒绝消息现已记入日志中。当一些拒绝消息被忽略或通过 dontaudit 规则处理时，请谨慎决定应该允许哪些拒绝信息。若有疑问或寻求指导，请联系 SELinux 列表上的其他 SELinux 用户和开发人员，如 *fedora-selinux-list*[4]。

要重建策略并启用 dontaudit 规则，以 root 用户身份运行下面的命令：

```
/usr/sbin/semodule -B
```

这个命令将策略恢复到它的原始状态。对于规则的完整列表，运行 sesearch -- dontaudit 命令，使用-s *domain* 选项和 grep 命令限制搜索范围：

```
$ sesearch --dontaudit -s smbd_t | grep squid
WARNING: This policy contained disabled aliases; they have been removed. dontaudit smbd_t
squid_port_t : tcp_socket name_bind ;
dontaudit smbd_t squid_port_t : udp_socket name_bind ;
```

有关分析拒绝消息的信息，请参阅“原始监听消息”和“sealert 消息”

9.3.3 服务手册页

服务手册页包含了有价值的信息，比如哪种文件类型可用于给定情况，还包含了布尔变量，可用于更改服务的访问权限（如 httpd 访问 NFS 文件系统）。这些信息可能在标准手册页中，也可能在以 selinux 为前缀或后缀的手册页中。

例如，httpd_selinux(8) 手册页提供了哪种文件类型可用于给定情况的相关信息，还

提供了允许脚本、共享文件、访问用户主目录内的子目录等的布尔变量。其他含有 SELinux 服务信息的手册页包括：

- Samba: samba_selinux(8) 手册页描述了通过 Samba 导出的文件和目录必须标记为 samba_share_t 类型，还描述了允许通过 Samba 导出、标记为非 samba_share_t 类型的文件的布尔变量。
- nfs_selinux(8)手册页描述了默认情况下不能通过 NFS 导出文件系统，还描述了要允许导出文件系统，必须将 nfs_export_all_ro 或 nfs_export_all_rw 等布尔变量设置为 on。
- 伯克利因特网命名域 (BIND): named(8) 手册页描述了哪种文件可用于给定情况（请参阅“Red Hat SELinux BIND 安全介绍”章节）。named_selinux(8)描述了默认情况下 named 无法写入主区域文件，还描述了要允许这种访问，必须将 named_write_master_zones 布尔变量设置为 on.

手册页上的信息可以帮助您配置正确的文件类型和布尔值，帮助防止 SELinux 不会拒绝访问。

9.3.4 许可域

当 SELinux 运行于许可模式时， SELinux 不得拒绝访问，但是若运行于强制模式时拒绝了动作，那么拒绝消息将记入日志中。以前无法使某一个域获得许可（切记：进程运行于域中）。在某些情况下，为了修复问题，这会使整个系统都获得许可。

中标麒麟可信操作系统 V6.0 包括了许可域，其中一个管理员可以配置一个单一进程(域)来运行许可，而不是让整个系统都获得许可。对许可域仍执行 SELinux 检查；然而，若 SELinux 拒绝了访问，那么内核允许访问并报告一个 AVC 拒绝消息。

中标麒麟可信操作系统 V6.0 中，*domain*_disable_trans 布尔变量可用于防止应用程序迁移到限制域，因此，进程运行于非限制域，如 initrc_t。将这样的布尔变量设置为 on 可能会引起严重的问题。例如，若 httpd_disable_trans 布尔变量设为 on :

1） httpd 运行于非限制 initrc_t 域。进程运行于 initrc_t 域创建的文件与进程运行于 httpd_t 域创建的文件可能没有相同的标记规则，这就可能允许进程创建误标记文件。因此，以后会引发访问问题。

2） 允许与 httpd_t 通讯的限制域不能与 initrc_t 通讯，可能引发其他故障。

许可域可用于：

1） 让单一进程（域）运行在许可状态来修复问题，而不是使整个系统获得许可，让整个系统面临危险。

2） 为新应用程序创建策略。以前，推荐创建最小策略，然后使整个机器进入许可模

式，这样应用程序可以运行，但 SELinux 拒绝消息仍记入日志。然后，可能使用 audit2allow 帮助编写策略。这使整个系统处于危险之中。使用许可域，可以只将新策略中的域标记为许可，不会使整个系统处于危险之中。

使域获得许可：

要使域获得许可，运行 semanage permissive -a *domain* 命令，其中 *domain* 是想要使之获得许可的域。例如，作为 Liuxroot 用户运行下面命令，使 httpd_t 域（Apache HTTP 服务器运行所在的域）获得许可：

```
/usr/sbin/semanage permissive -a httpd_t
```

要查看获得允许的域列表，以 root 用户身份运行 semodule -l | grep permissive 命令。例如：

```
# /usr/sbin/semodule -l | grep permissive permissive_httpd_t 1.0
```

若您不再想要域获得许可，以 root 用户身份运行 semanage permissive -d domain 命令。例如：

```
/usr/sbin/semanage permissive -d httpd_t
```

许可域的拒绝消息：

SYSCALL 消息对于许可域来说是不同的。以下是来自 Apache HTTP 服务器的 AVC 拒绝消息（相关系统调用）示例。

```
type=AVC msg=audit(1226882736.442:86): avc: denied { getattr } for pid=2427
comm="httpd" path="/var/www/html/file1" dev=dm-0 ino=284133
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

type=SYSCALL msg=audit(1226882736.442:86): arch=40000003 syscall=196 success=no exit=-13
a0=b9a1e198 a1=bfc2921c a2=54dff4 a3=2008171 items=0 ppid=2425 pid=2427 auid=502
uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4 comm="httpd"
exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

默认情况下，httpd_t 域不是许可域，因而拒绝这个操作，并且 SYSCALL 消息包含 success=no。以下示例是同样情况下的 AVC 拒绝消息，只是运行了 semanage permissive -a httpd_t 命令使 httpd_t 域获得了许可：

```
type=AVC msg=audit(1226882925.714:136): avc: denied { read } for pid=2512
comm="httpd" name="file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

type=SYSCALL msg=audit(1226882925.714:136): arch=40000003 syscall=5 success=yes exit=11
```

```
a0=b962a1e8 a1=8000 a2=0 a3=8000 items=0  ppid=2511 pid=2512 auid=502 uid=48 gid=48 euid=48
suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none)  ses=4 comm="httpd" exe="/usr/sbin/
httpd"  subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

在本例中，虽然 AVC 拒绝消息已记入日志中，但是并未拒绝访问，正如 SYSCALL 消息中的 success=yes 所示。

有关许可域的详细信息，请参阅 Dan Walsh 博客中的“许可域”

9.3.5 搜索和查看拒绝消息

本节假设安装了 *setroubleshoot*、*setroubleshoot-server*、*dbus* 和 *audit* 包，且 auditd、rsyslogd 和 setroubleshootd 后台程序正在运行。有关启动这些后台程序的信息，请参阅 *“使用哪个日志文件”*。有很多工具可用于搜索和查看 SELinux 拒绝消息，如 ausearch、aureport 和 sealert。

ausearch

audit 包提供了 ausearch。从 ausearch(8)手册页开始：“ausearch 是基于不同搜索标准查询以事件为基础的审核后台程序日志的一种工具”[6]。ausearch 工具访问 /var/log/audit/audit.log，因而，必须以 root 用户身份运行该工具：

表 2-9 ausearch 选项

搜索	命令
所有拒绝消息	/sbin/ausearch -m avc
今日的拒绝消息	/sbin/ausearch -m avc -ts today
最近 10 分钟的拒绝消息	/sbin/ausearch -m avc -ts recent

要搜索特定服务的 SELinux 拒绝消息，使用-c *comm-name* 选项，其中，*comm- name* 是“可执行文件名”[7]，例如，用于 Apache HTTP 的 httpd，以及用于 Samba 的 smbd。

```
/sbin/ausearch -m  avc -c  httpd
/sbin/ausearch -m  avc -c  smbd
```

详细了解 ausearch 选项，请参阅 ausearch(8)手册页。

aureport

audit 包提供 aureport。aureport(8)手册页中描述：“aureport 是一种为监听系统日志 8. 生成总结报告的工具”。aureport 工具访问 /var/ log/audit/audit.log，因而，必须以 root 用户访问。要查看 SELinux 拒绝信息列表以及每种拒绝信息发生的次数，运行 aureport -a 命令。以下示例是包含两个拒绝消息的输出：

```
#  /sbin/aureport -a
===============================================================
```

```
#  date time comm subj syscall  class  permission obj event
============================================================
05/01/2009 21:41:39 httpd  unconfined_u:system_r:httpd_t:s0 195  file getattr
system_u:object_r:samba_share_t:s0  denied 2
05/03/2009 22:00:25 vsftpd unconfined_u:system_r:ftpd_t:s0 5  file  read
unconfined_u:object_r:cifs_t:s0  denied 4
```

详细了解 aureport 选项，请参阅 aureport(8)手册页。

sealert

setroubleshoot-server 包提供了 sealert，它读取由 *setroubleshoot-server* 转换的拒绝消息。正如/var/log/messages 中所示，为拒绝消息分配了 ID，。以下示例是来自 messages 的拒绝消息：

```
setroubleshoot: SELinux  is  preventing httpd (httpd_t) "getattr" to /var/www/html/
file1 (samba_share_t).  For complete SELinux messages.  run  sealert -l 84e0b04d-
d0ad-4347-8317-22e74f6cd020
```

本例中，拒绝消息的 ID 是 84e0b04d-d0ad-4347-8317-22e74f6cd020。-l 选项把 ID 作为一个参数。运行 sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020 命令将详细分析 SELinux 拒绝访问的原因，并提供允许访问的可能解决方法。

若您正在运行 X Window 系统， setroubleshoot 和 setroubleshoot-server 包已安装，且 setroubleshootd、dbus 和 auditd 后台程序正在运行，当 SELinux 拒绝访问时将显示警告。单击“显示“启动 sealert GUI，并以 HTML 输出形式显示拒绝消息。

图 2-8 SELinux 探测异常行为

1） 运行 sealert -b 命令来启动 sealert GUI.

2） 运行 sealert -l *命令来查看所有拒绝消息的详细分析。

3） 以 root 用户身份运行 sealert -a /var/log/audit/audit.log -H > audit.html 命令来创建一个 HTML 版本的 sealert 分析，正如使用 sealert GUI 时所示

详细了解 sealert 选项的信息，请参阅 sealert(8)手册页。

9.3.6 原始监听消息

原始监听消息记录到/var/log/audit/audit.log 日志。以下示例是 Apache HTTP 服务器(运行在 httpd_t 域)尝试访问/var/www/html/file1 文件（标记为 samba_share_t 类型）时发生的 AVC 拒绝消息（和相关系统调用）：

```
type=AVC msg=audit(1226874073.147:96): avc: denied { getattr } for pid=2465
comm="httpd" path="/var/www/html/file1" dev=dm-0 ino=284133
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

type=SYSCALL msg=audit(1226874073.147:96): arch=40000003 syscall=196 success=no exit=-13
a0=b98df198 a1=bfec85dc a2=54dff4 a3=2008171 items=0 ppid=2463 pid=2465 auid=502 uid=48
gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=6 comm="httpd"
exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)

{ getattr }
```

括号中的项目表明权限已被拒绝。getattr 表明源进程试图读取目标文件的状态信息。这发生在读文件之前。拒绝这个操作是因为访问文件的标记错误。常见权限包括 getattr、read 和 write。

comm="*httpd*"

启动进程的可执行文件。可执行文件的完整路径位于系统调用(SYSCALL)消息的 exe 部分，本例中为 exe="/usr/sbin/httpd"。

path="/var/www/html/file1"

进程试图访问的客体（目标）的路径。

scontext="unconfined_u:system_r:httpd_t:s0"

尝试拒绝操作的进程的 SELinux 上下文。本例中，它是运行于 httpd_t 域的 Apache HTTP 服务器的 SELinux 上下文。

tcontext="unconfined_u:object_r:samba_share_t:s0"

进程试图访问的客体（目标）的 SELinux 上下文。本例中，它是 file1 的 SELinux 上

下文。注意：运行于 httpd_t 域中的进程无法获得 samba_share_t 类型。

某些情况下，例如，进程试图执行一个系统服务时，由于该服务会更改该运行进程的特性，如用户 ID，tcontext 可能会匹配 scontext。另外，若进程试图使用的资源多于正常限制所允许的资源（如内存），会导致进行安全检查，查看是否允许该进程摆脱这些限制，此时 tcontext 可能会匹配 scontext。

在系统调用消息(SYSCALL)中，有两个项目须关注：

- **success=*no*:** 表示拒绝(AVC) 是否强制执行。success=no 表示系统调用不成功（SELinux 拒绝访问）。success=yes 表示系统调用成功，这可以在许可域或非限制域中查看，如 initrc_t 和 kernel_t。
- **exe="*/usr/sbin/httpd*":** 启动进程的可执行文件的完整路径，本例中为 exe="/usr/sbin/httpd"。

错误的文件类型是 SELinux 拒绝访问的常见原因。要开始故障修复，将源上下文(scontext)和目标上下文(tcontext)进行比较。进程(scontext)应该访问这样一个客体(tcontext)吗？例如，除非另有配置，否则 Apache HTTP 服务器(httpd_t) 应该只访问 httpd_selinux(8)手册页中指定的类型，如 httpd_sys_content_t 、public_content_t 等类型。

9.3.7 sealert 消息

如/var/log/messages 所示，为拒绝消息分配了 ID。以下示例是 Apache HTTP 服务器（运行于 httpd_t 域）试图访问/var/www/html/file1 文件（标记为 samba_share_t 类型）时发生的 AVC 拒绝消息（记入 messages 日志中）：

```
hostname setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to /var/www/
html/file1 (samba_share_t). For complete SELinux messages. run sealert -l 84e0b04d-
d0ad-4347-8317-22e74f6cd020
```

按照建议，运行 sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020 命令来查看完整消息。该命令仅在本机上有效，还显示了和 sealert GUI 相同的信息：

```
$ sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020
Summary:
SELinux is preventing httpd (httpd_t) "getattr" to /var/www/html/file1
(samba_share_t).
Detailed Description:
SELinux denied access to /var/www/html/file1 requested by httpd.
/var/www/html/file1 has a context used for sharing by different program. If you
would like to share /var/www/html/file1 from httpd also, you need to change its
file context to public_content_t. If you did not intend to this access, this
```

```
could signal a intrusion attempt.
Allowing Access:
You can alter the file context by executing chcon -t public_content_t
'/var/www/html/file1'
Fix Command:
chcon -t public_content_t '/var/www/html/file1'
Additional Information:
Source Context unconfined_u:system_r:httpd_t:s0
Target Contextunconfined_u:object_r:samba_share_t:s0
Target Objects /var/www/html/file1 [ file ] Sourcehttpd
Source Path /usr/sbin/httpd
Port <Unknown> Host hostname
Source RPM Packages httpd-2.2.10-2
Target RPM Packages
Policy RPM selinux-policy-3.5.13-11.fc12
Selinux Enabled True
Policy Type targeted
MLS Enabled True
Enforcing Mode Enforcing
Plugin Name public_content
Host Name hostname
Platform Linux hostname 2.6.27.4-68.fc12.i686 #1 SMP Thu Oct
30 00:49:42 EDT 2008 i686 i686
Alert Count 4
First Seen Wed Nov 5 18:53:05 2008
Last Seen Wed Nov 5 01:22:58 2008
Local ID 84e0b04d-d0ad-4347-8317-22e74f6cd020
Line Numbers
Raw Audit Messages
node=hostname type=AVC msg=audit(1225812178.788:101): avc: denied { getattr }
for pid=2441 comm="httpd" path="/var/www/html/file1" dev=dm-0 ino=284916
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:samba_share_t:s0
tclass=file
node=hostname type=SYSCALL msg=audit(1225812178.788:101): arch=40000003 syscall=196
success=no exit=-13 a0=b8e97188 a1=bf87aaac a2=54dff4 a3=2008171 items=0 ppid=2439
pid=2441
```

```
auid=502 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none)  ses=3
comm="httpd" exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

总结

拒绝操作的简要总结。这与/var/log/messages 中的拒绝消息相同。本例中，拒绝 httpd 进程访问文件(file1)，因为该文件标记为 samba_share_t 类型。

详细说明

更详细的描述。本例中，file1 文件标记为 samba_share_t 类型。该类型用于想要通过 Samba 导出的文件和目录。描述信息建议将类型更改为需要时可以通过 Apache HTTP 服务器和 Samba 访问的类型。

允许访问

如何允许访问的建议。这可能会重新标记文件，开启布尔值，或建立局域策略模块。在此例中，建议将文件标记为 Apache HTTP 服务器和 Samba 都可访问的类型。

修复命令

建议使用的命令，可允许访问和解决拒绝问题。本例中，它提供了将 file1 类型更改为 Apache HTTP 服务器和 Samba 都可访问的 public_content_t 类型的命令。

其他信息

缺陷报告中的有用信息，如策略包名字和版本信息(selinux- policy-3.5.13-11.fc12)，但是对解决发生拒绝的原因可能没有帮助。

原始监听消息

与拒绝相关的来自/var/log/audit/audit.log 的原始监听消息。更多关于 AVC 拒绝消息中各个项目的信息，请参阅，*“原始监听消息”*。

9.3.8 允许访问: audit2allow

在实际操作中，请勿使用本节中的示例。它仅仅用于描述 audit2allow 的使用方法。

audit2allow(1)手册页中描述："audit2allow - generate SELinux 策略允许拒绝操作日志中的规则"9.根据*章节“sealert 消息”*的内容分析拒绝消息后，并且若没有标记变更或允许的布尔变量访问，请使用 audit2allow 创建局部策略模块。SELinux 拒绝访问后，运行 audit2allow 命令显示类型强制规则，该规则允许以前被拒绝的访问。

下例演示使用 audit2allow 创建一个策略模块：

1） 拒绝消息和相关系统调用记录到/var/log/audit/audit.log。

```
type=AVC  msg=audit(1226270358.848:238): avc:  denied  {  write }
for  pid=13349  comm="certwatch" name="cache" dev=dm-0 ino=218171
```

```
scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0 tclass=dir

type=SYSCALL msg=audit(1226270358.848:238): arch=40000003 syscall=39 success=no
exit=-13
a0=39a2bf a1=3ff a2=3a0354 a3=94703c8 items=0 ppid=13344 pid=13349 auid=4294967295
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295
comm="certwatch" exe="/usr/bin/certwatch" subj=system_u:system_r:certwatch_t:s0
key=(null)
```

本例中，拒绝 certwatch (comm="certwatch")对({ write })标记为 var_t 类型(tcontext=system_u:object_r:var_t:s0)的目录具有写入权限。根据 “sealert 消息”分析拒绝消息。若无标记变更或允许的布尔变量访问，使用 audit2allow 创建一个局部策略模块。

2） 使用记入日志的拒绝消息，如步骤 1 中的 certwatch 拒绝消息，运行 audit2allow -w -a 命令生成人类可读的描述信息，说明拒绝访问的原因。-a 选项是读取所有监听日志。-w 选项生成人类可读的描述信息。

3） audit2allow 工具访问/var/log/audit/audit.log 日志，因而必须以 root 用户身份运行：

```
# audit2allow -w -a
type=AVC msg=audit(1226270358.848:238): avc: denied { write }
for pid=13349 comm="certwatch" name="cache" dev=dm-0 ino=218171
scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0 tclass=dir
Was caused by:
Missing type enforcement (TE) allow rule.
You can use audit2allow to generate a loadable module to allow this access.
```

正如所示，由于类型强制规则丢失，因此拒绝访问。

4） 运行 audit2allow -a 命令来查看允许拒绝访问的类型强制规则。

```
# audit2allow -a
#============= certwatch_t ==============
allow certwatch_t var_t:dir write;
```

要使用 audit2allow -a 显示的规则，作为 root 用户运行 audit2allow -a -M mycertwatch 命令创建自定义模块。-M 选项在当前工作目录中，用-M 指定的名字创建类型强制文件(.te)。

```
# audit2allow -a -M mycertwatch

****************** IMPORTANT **********************
To make this policy package active, execute:
```

```
semodule -i mycertwatch.pp

#  ls
mycertwatch.pp  mycertwatch.te
```

同样，audit2allow 将类型强制规则编译到策略包(.pp)中。要安装此模块，以 Linux root 用户身份运行/usr/sbin/semodule -i mycertwatch.pp 命令。

 重要：用 audit2allow 创建的模块可能允许的访问多于所请求的访问。我们建议，用 audit2allow 创建的策略应加入到 SELinux 列表中进行审查，例如 fedora-selinux-list1。

若多个进程出现多次拒绝，但是只想为单个进程创建一个自定义策略，使用 grep 命令来限制 audit2allow 的输入范围。下例描述了使用 grep 通过 audit2allow 只发送与 certwatch 有关的拒绝消息。

```
#  grep certwatch /var/log/audit/audit.log | audit2allow -M  mycertwatch2
****************** IMPORTANT  **********************
To  make  this policy  package active,  execute:
#  /usr/sbin/semodule -i mycertwatch2.pp
```

第 3 章 用户权能设置

1. 权能概述

本章将主要介绍中标麒麟可信操作系统中提供的用户权能设置支持。通过用户权能的动态设置，安全管理员可以为某位已知用户添加或者删除某一项乃至某些项权能，以达到对用户权限的限制。对于普通用户，不需要给予 root 权限，只需更改该普通用户某项或某几项权能即可实现某些特定的操作。其目标便是消除需要执行某些操作的普通用户对 root 帐户的依赖。因此用户权能的动态设置大大降低了系统面临的风险。下面详细介绍了安全管理员进行用户权能设置的命令集。

2. 用户权能设置命令集

usercapset -v //查看配置文件中用户权能信息

usercapset -u uid -s order //设置 uid 的权能为 order，此设置不写文件

usercapset -u uid -w order //设置 uid 的权能为 order，此设置写入文件

usercapset -c //通过系统调用查看当前用户的权能

usercapset -k //将用户权能从配置文件中加载到内核

usercapset --help //帮助命令

2.1 帮助命令

#usercapset –p

或#usercapset –help

打印输出基本的命令、可供设置的 order 列表和默认的三特权权能信息。

```
[root@localhost ~]# usercapset -p
---------------------------- help -------------------------------------
./usercapset -v               //Capability list view from the file.
./usercapset -u uid -s order  //Set capability, do not write the file.  eg. ./usercapset -u 12 -s CAP_CHOWN+CAP_KILL
./usercapset -u uid -w order  //Set capability and write the file.      eg. ./usercapset -u 12 -w CAP_CHOWN
./usercapset -c               //Get the current user's capability, not read file
./usercapset -k               //Read into the kernel from the file.
./usercapset -d uid           //Delete a row uid informaton in the file.
./usercapset -u uid -a order  //Add some cap.                          eg. ./usercapset -u 50 -a CAP_CHOWN+CAP_KILL
./usercapset -u uid -r order  //Remove some cap.                       eg. ./usercapset -u 50 -r CAP_CHOWN
./usercapset -p               //help.
----------------------------order list---------------------------------
CAP_CHOWN               0     //允许改变文件的所有权
CAP_DAC_OVERRIDE        1     //忽略对文件的所有DAC访问限制
CAP_DAC_READ_SEARCH     2     //忽略所有对读，搜索操作的限制
CAP_FOWNER              3     //如果文件属于进程的UID，就取消对文件读限制
```

图 3-1 帮助命令

2.2 查看当前用户权能

usercapset –c

```
[root@localhost ~]# usercapset -c
-------------- The current user information --------------------
User:   root
uid:    0
cap effective data: 0xffffffff
----------------------------------------------------------------
```

图 3-2 查看当前用户权能命令

2.3 查看配置文件中用户权能

usercapset –v

从配置文件中查看用户权能列表，默认打印输出三特权用户权能。

```
[root@localhost /]# usercapset -v
-------------------------------------------------capability list-------------------------------------------------------------------------------
       username             uid                                    capability
-----------------------------------------------------------------------------------------------------------------------------------------------
          root               0     0x3  0xffffffff
                                   CAP_ALL
-----------------------------------------------------------------------------------------------------------------------------------------------
         secadm             500     0x3  0xb880ffff
                                   CAP_CHOWN              CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH    CAP_FOWNER
                                   CAP_FSETID             CAP_KILL               CAP_SETGID             CAP_SETUID
                                   CAP_SETPCAP            CAP_LINUX_IMMUTABLE    CAP_NET_BIND_SERVICE   CAP_NET_BROADCAST
                                   CAP_NET_ADMIN          CAP_NET_RAW            CAP_IPC_LOCK           CAP_IPC_OWNER
                                   CAP_SYS_NICE           CAP_MKNOD              CAP_LEASE              CAP_AUDIT_WRITE
                                   CAP_SETFCAP            CAP_MAC_OVERRIDE       CAP_MAC_ADMIN
-----------------------------------------------------------------------------------------------------------------------------------------------
        auditadm             501     0x0  0x7880feff
                                   CAP_CHOWN              CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH    CAP_FOWNER
                                   CAP_FSETID             CAP_KILL               CAP_SETGID             CAP_SETUID
                                   CAP_LINUX_IMMUTABLE    CAP_NET_BIND_SERVICE   CAP_NET_BROADCAST      CAP_NET_ADMIN
                                   CAP_NET_RAW            CAP_IPC_LOCK           CAP_IPC_OWNER          CAP_SYS_NICE
                                   CAP_MKNOD              CAP_LEASE              CAP_AUDIT_WRITE        CAP_AUDIT_CONTROL
-----------------------------------------------------------------------------------------------------------------------------------------------
          other              --     0
-----------------------------------------------------------------------------------------------------------------------------------------------
```

图 3-3 查看用户权能命令

2.4 设置用户权能

1、设置 uid 的权能为 order，此设置不写入配置文件。命令如下：

usercapset -u uid -s order

例：将 uid = 501 的用户权能设置为 CAP_CHOWN 的命令为：

#usercapset –u 501 –s CAP_CHOWN

若将 uid = 501 的用户权能设置为 CAP_CHOWN 和 CAP_DAC_OVERRIDE 的命令为：

#usercapset –u 501 –s CAP_CHOWN+CAP_DAC_OVERRIDE

权能 order 可以从 2.1 节中查找。一次设置多个 order，用“+”连接。

```
[root@localhost /]# usercapset -u 501 -s CAP_CHOWN
[root@localhost /]# usercapset -u 501 -s CAP_CHOWN+CAP_DAC_OVERRIDE
[root@localhost /]# usercapset -v
-------------------------------------------capability list-------------------------------------------------------------------------
        username              uid                                    capability
-----------------------------------------------------------------------------------------------------------------------------------
            root                0     0x3  0xffffffff
                                      CAP_ALL
-----------------------------------------------------------------------------------------------------------------------------------
           secadm             500     0x3  0xb880ffff
                                      CAP_CHOWN             CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH    CAP_FOWNER
                                      CAP_FSETID            CAP_KILL               CAP_SETGID             CAP_SETUID
                                      CAP_SETPCAP           CAP_LINUX_IMMUTABLE    CAP_NET_BIND_SERVICE   CAP_NET_BROADCAST
                                      CAP_NET_ADMIN         CAP_NET_RAW            CAP_IPC_LOCK           CAP_IPC_OWNER
                                      CAP_SYS_NICE          CAP_MKNOD              CAP_LEASE              CAP_AUDIT_WRITE
                                      CAP_SETFCAP           CAP_MAC_OVERRIDE       CAP_MAC_ADMIN
-----------------------------------------------------------------------------------------------------------------------------------
         auditadm             501     0x0  0x7880feff
                                      CAP_CHOWN             CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH    CAP_FOWNER
                                      CAP_FSETID            CAP_KILL               CAP_SETGID             CAP_SETUID
                                      CAP_LINUX_IMMUTABLE   CAP_NET_BIND_SERVICE   CAP_NET_BROADCAST      CAP_NET_ADMIN
                                      CAP_NET_RAW           CAP_IPC_LOCK           CAP_IPC_OWNER          CAP_SYS_NICE
                                      CAP_MKNOD             CAP_LEASE              CAP_AUDIT_WRITE        CAP_AUDIT_CONTROL
-----------------------------------------------------------------------------------------------------------------------------------
            other               --     0
-----------------------------------------------------------------------------------------------------------------------------------
```

图 3-4 设置用户权能命令

2、设置 uid 的权能为 order，此设置写入配置文件。命令如下：

usercapset -u uid –w order

例：将 uid = 501 的用户权能设置为 CAP_CHOWN 的命令为：

#usercapset –u 501 –w CAP_CHOWN

若将 uid = 501 的用户权能设置为 CAP_CHOWN 和 CAP_DAC_OVERRIDE 的命令为：

#usercapset –u 501 –w CAP_CHOWN+CAP_DAC_OVERRIDE

权能 order 可以从 2.1 节中查找。一次设置多个 order，用“+”连接。

```
[root@localhost /]# usercapset -u 501 -w CAP_CHOWN+CAP_DAC_OVERRIDE
[root@localhost /]# usercapset -v
-------------------------------------------capability list-------------------------------------------------------------------------
        username              uid                                    capability
-----------------------------------------------------------------------------------------------------------------------------------
            root                0     0x3  0xffffffff
                                      CAP_ALL
-----------------------------------------------------------------------------------------------------------------------------------
           secadm             500     0x3  0xb880ffff
                                      CAP_CHOWN             CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH    CAP_FOWNER
                                      CAP_FSETID            CAP_KILL               CAP_SETGID             CAP_SETUID
                                      CAP_SETPCAP           CAP_LINUX_IMMUTABLE    CAP_NET_BIND_SERVICE   CAP_NET_BROADCAST
                                      CAP_NET_ADMIN         CAP_NET_RAW            CAP_IPC_LOCK           CAP_IPC_OWNER
                                      CAP_SYS_NICE          CAP_MKNOD              CAP_LEASE              CAP_AUDIT_WRITE
                                      CAP_SETFCAP           CAP_MAC_OVERRIDE       CAP_MAC_ADMIN
-----------------------------------------------------------------------------------------------------------------------------------
         auditadm             501     0x0  0x3
                                      CAP_CHOWN             CAP_DAC_OVERRIDE
-----------------------------------------------------------------------------------------------------------------------------------
            other               --     0
-----------------------------------------------------------------------------------------------------------------------------------
```

图 3-5 设置用户权能命令

2.5 修改用户权能

1、为用户添加某项或某几项权能 order。命令如下：

usercapset -u uid –a order

例：为 uid = 501 的用户添加 CAP_CHOWN 权能的命令为：

#usercapset –u 501 –a CAP_CHOWN

若为 uid = 501 的用户添加 CAP_CHOWN 和 CAP_DAC_OVERRIDE 权能的命令为：

#usercapset –u 501 –a CAP_CHOWN+CAP_DAC_OVERRIDE

权能 order 可以从 2.1 节中查找。一次设置多个 order，用“+”连接。

```
[root@localhost ~]# usercapset -v
--------------------------------------------------capability list--------------------------------------------------
       username                uid                        capability
--------------------------------------------------------------------------------------------------------------------
           root                  0      0x3  0xffffffff
                                        CAP_ALL
--------------------------------------------------------------------------------------------------------------------
         secadm                500      0x3  0xb880ffff
                                        CAP_CHOWN              CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH     CAP_FOWNER
                                        CAP_FSETID             CAP_KILL               CAP_SETGID              CAP_SETUID
                                        CAP_SETPCAP            CAP_LINUX_IMMUTABLE    CAP_NET_BIND_SERVICE    CAP_NET_BROADCAST
                                        CAP_NET_ADMIN          CAP_NET_RAW            CAP_IPC_LOCK            CAP_IPC_OWNER
                                        CAP_SYS_NICE           CAP_MKNOD              CAP_LEASE               CAP_AUDIT_WRITE
                                        CAP_SETFCAP            CAP_MAC_OVERRIDE       CAP_MAC_ADMIN
--------------------------------------------------------------------------------------------------------------------
       auditadm                501      0x0  0x3
                                        CAP_CHOWN              CAP_DAC_OVERRIDE
--------------------------------------------------------------------------------------------------------------------
          other                 --      0
--------------------------------------------------------------------------------------------------------------------
[root@localhost ~]# usercapset -u 501 -a CAP_DAC_READ_SEARCH
[root@localhost ~]# usercapset -v
--------------------------------------------------capability list--------------------------------------------------
       username                uid                        capability
--------------------------------------------------------------------------------------------------------------------
           root                  0      0x3  0xffffffff
                                        CAP_ALL
--------------------------------------------------------------------------------------------------------------------
         secadm                500      0x3  0xb880ffff
                                        CAP_CHOWN              CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH     CAP_FOWNER
                                        CAP_FSETID             CAP_KILL               CAP_SETGID              CAP_SETUID
                                        CAP_SETPCAP            CAP_LINUX_IMMUTABLE    CAP_NET_BIND_SERVICE    CAP_NET_BROADCAST
                                        CAP_NET_ADMIN          CAP_NET_RAW            CAP_IPC_LOCK            CAP_IPC_OWNER
                                        CAP_SYS_NICE           CAP_MKNOD              CAP_LEASE               CAP_AUDIT_WRITE
                                        CAP_SETFCAP            CAP_MAC_OVERRIDE       CAP_MAC_ADMIN
--------------------------------------------------------------------------------------------------------------------
       auditadm                501      0x0  0x7
                                        CAP_CHOWN              CAP_DAC_OVERRIDE       CAP_DAC_READ_SEARCH
--------------------------------------------------------------------------------------------------------------------
          other                 --      0
--------------------------------------------------------------------------------------------------------------------
```

图 3-6 添加用户权能命令

如上图所示，uid 为 501 的用户原来的权限是 CAP_CHOWN 和 CAP_DAC_OVERRIDE，运行添加权能命令后，该用户添加了 CAP_DAC_READ_SEARCH 权能。

2、为用户删除某项或某几项权能 order。命令如下：

usercapset -u uid –r order

例：为 uid = 501 的用户删除 CAP_CHOWN 权能的命令为：

#usercapset –u 501 –r CAP_CHOWN

若为 uid = 501 的用户删除 CAP_CHOWN 和 CAP_DAC_OVERRIDE 权能的命令为：

#usercapset –u 501 –r CAP_CHOWN+CAP_DAC_OVERRIDE

权能 order 可以从 2.1 节中查找。一次设置多个 order，用“+”连接。

```
[root@localhost ~]# usercapset -v
-------------------------------------------capability list-------------------------------------------
        username              uid                                 capability
-----------------------------------------------------------------------------------------------------
            root                0     0x3  0xffffffff
                                      CAP_ALL
-----------------------------------------------------------------------------------------------------
           secadm             500     0x3  0xb880ffff
                                      CAP_CHOWN             CAP_DAC_OVERRIDE      CAP_DAC_READ_SEARCH   CAP_FOWNER
                                      CAP_FSETID            CAP_KILL              CAP_SETGID            CAP_SETUID
                                      CAP_SETPCAP           CAP_LINUX_IMMUTABLE   CAP_NET_BIND_SERVICE  CAP_NET_BROADCAST
                                      CAP_NET_ADMIN         CAP_NET_RAW           CAP_IPC_LOCK          CAP_IPC_OWNER
                                      CAP_SYS_NICE          CAP_MKNOD             CAP_LEASE             CAP_AUDIT_WRITE
                                      CAP_SETFCAP           CAP_MAC_OVERRIDE      CAP_MAC_ADMIN
-----------------------------------------------------------------------------------------------------
         auditadm             501     0x0  0x7
                                      CAP_CHOWN             CAP_DAC_OVERRIDE      CAP_DAC_READ_SEARCH
-----------------------------------------------------------------------------------------------------
            other              --     0
-----------------------------------------------------------------------------------------------------
[root@localhost ~]# usercapset -u 501 -r CAP_DAC_READ_SEARCH
[root@localhost ~]# usercapset -v
-------------------------------------------capability list-------------------------------------------
        username              uid                                 capability
-----------------------------------------------------------------------------------------------------
            root                0     0x3  0xffffffff
                                      CAP_ALL
-----------------------------------------------------------------------------------------------------
           secadm             500     0x3  0xb880ffff
                                      CAP_CHOWN             CAP_DAC_OVERRIDE      CAP_DAC_READ_SEARCH   CAP_FOWNER
                                      CAP_FSETID            CAP_KILL              CAP_SETGID            CAP_SETUID
                                      CAP_SETPCAP           CAP_LINUX_IMMUTABLE   CAP_NET_BIND_SERVICE  CAP_NET_BROADCAST
                                      CAP_NET_ADMIN         CAP_NET_RAW           CAP_IPC_LOCK          CAP_IPC_OWNER
                                      CAP_SYS_NICE          CAP_MKNOD             CAP_LEASE             CAP_AUDIT_WRITE
                                      CAP_SETFCAP           CAP_MAC_OVERRIDE      CAP_MAC_ADMIN
-----------------------------------------------------------------------------------------------------
         auditadm             501     0x0  0x3
                                      CAP_CHOWN             CAP_DAC_OVERRIDE
-----------------------------------------------------------------------------------------------------
            other              --     0
-----------------------------------------------------------------------------------------------------
```

图 3-7 删除用户权能命令

如上图所示，uid 为 501 的用户原来的权限是 CAP_CHOWN、CAP_DAC_OVERRIDE 和 CAP_DAC_READ_SEARCH，运行删除权能命令后，该用户删除了 CAP_DAC_READ_SEARCH 权能。

2.6 删除配置文件用户权能

删除配置文件中某用户的权能信息（三特权无法删除）。该命令仅仅是对配置文件的操作，无法删除系统中该用户，更不会删除其权能。命令如下：

usercapset –d uid

2.7 加载配置文件中权能

将配置文件中设置的所有用户的权能加载到内核。命令如下：

usercapset –k

2.8 注意事项

若不清楚某项权能的具体功能，请勿随意查看、删除或添加用户权能，否则可能造成严重后果。

第 4 章 中标麒麟安全控制中心

1. 简介

中标麒麟可信操作系统的安全控制中心基于图形化方式实现系统安全可信功能的集中配置和管理，界面友好，简洁易用；用户可以方便快捷完成系统的安全管理。安全控制中心集成了系统所有安全应用软件，包括可信相关应用软件和系统安全应用软件。可信应用软件包括可信管理中心，安全审计和可信度量；系统安全应用软件包括病毒扫描和弱点扫描。

2. 使用说明

目前中标麒麟安全控制中心共有 6 个应用，弱点扫描、病毒扫描和 Tboot 管理工具是系统管理员使用；可信管理中心是安全管理员使用；安全审计是审计管理员使用；可信保密箱是三个管理员都可以使用。安全管理员可以卸载所有软件。

2.1 启动安全控制管理中心

安装好中标麒麟可信操作系统以后，点击系统的启动项，选择系统工具->中标麒麟安全控制中心，弹出如下图所示界面：

图 4-1 安全控制中心主界面

主界面上包含，导航栏，显示区，托盘四部分。在标题栏中，不同的用户有使用不同应用的权限，高亮的应用即为当前用户可以使用的软件，显示灰色的应用当前用户不能使用，需要切换至具有使用权限的用户才可以使用。

2.2 添加应用程序

应用被卸载后，可以通过点击右上角的绿色“十”按钮来添加相关应用，

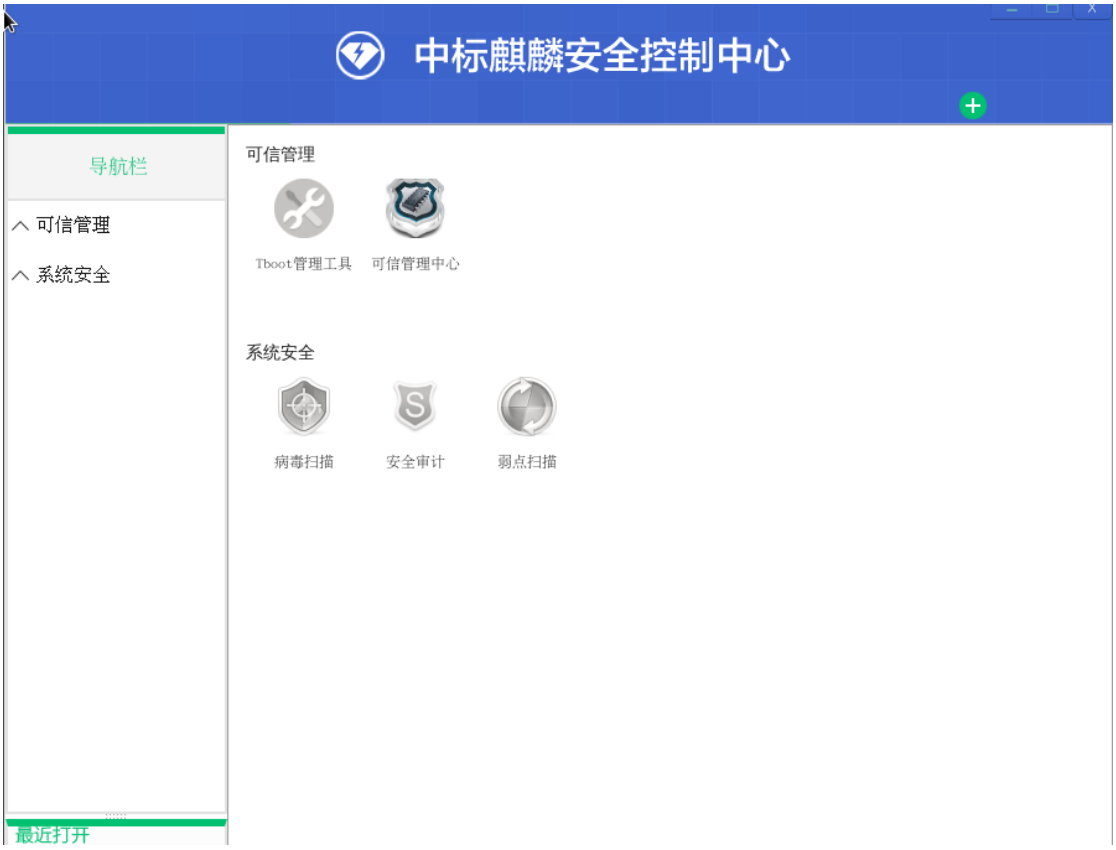


图 4-2 添加应用

上图中可信保密箱模块被卸载，通过点击右上角的绿色“十”按钮弹出软件管理中心如下图：

图 4-3 软件管理中心

点击“安装”即可安装对应的模块。如果要同时安装多个应用，可以勾选“全选”按钮，选择全部模块，点击“一键安装”将所有模块同时安装。

图 4-4 软件管理中心模块安装界面

安装过程如下图：

图 4-5 软件管理中心一键安装安装界面

安装完成后，模块状态显示为“已安装”。在安装过程中，如果想放弃安装，则点击对应软件后面的停止图标。关闭软件管理中心，安全控制中心主界面上会显示所有已经安装的安全应用模块，双击图标可以打开相应的应用模块。

2.3 托盘

启动安全控制管理中心之后，系统托盘上会显示对应的图标，右键图标，会出现“关于”、“退出”、“常用工具”和“最近打开”菜单项，如下图所示：

图 4-6 安全控制管理中心托盘

注：常用工具菜单项会显示经常使用的应用程序，最近打开菜单项会显示最近被使用过的应用程序，应用程序在频繁使用之后才会出现常用工具菜单项。初始情况是没有“常用工具”和“最近打开”。点击“最近打开”下的应用程序也可以启动程序。

点击“退出”，会退出安全控制管理中心；点击托盘菜单项中的“关于”，弹出安全控制管理中心的版本信息，如下图所示：

图 4-7 关于

2.4 主界面操作

点击导航栏中的一项，右侧显示区域会选中对应的图标，单击右侧显示显示区域的选中图标可以取消选中，如下图所示：

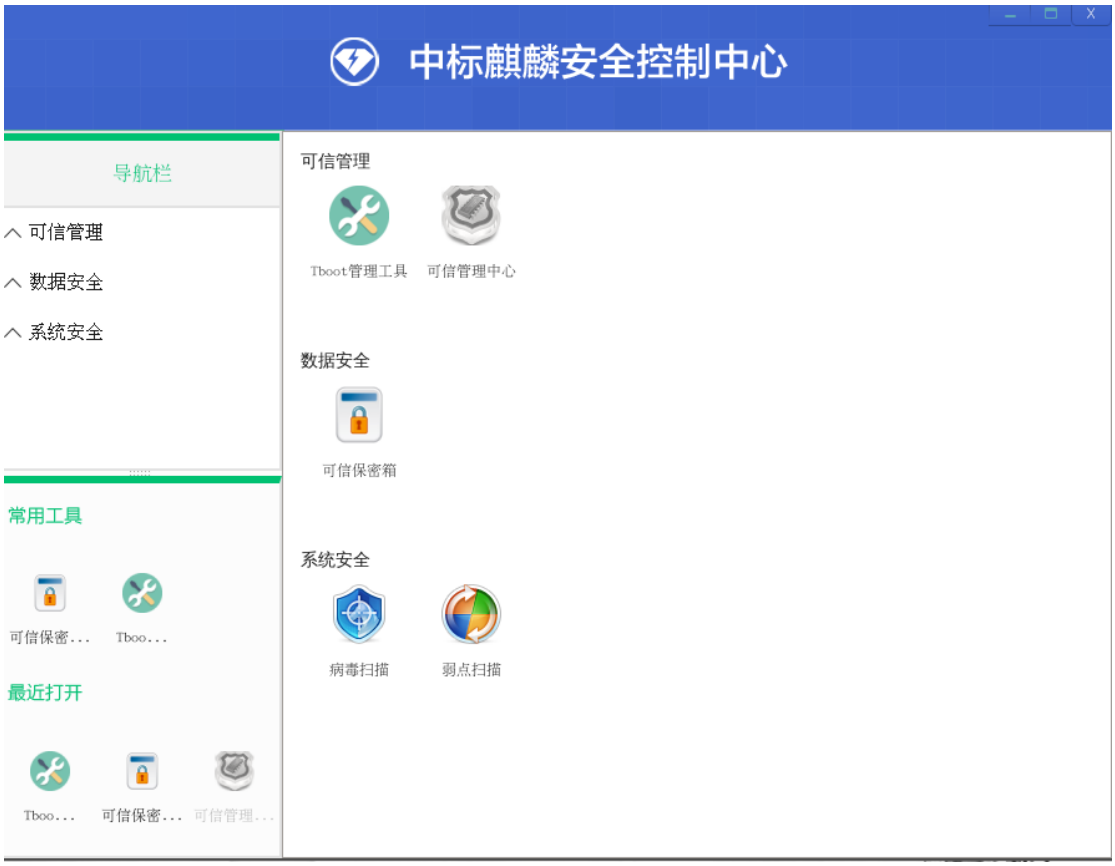


图 4-8 软件安装后的主界面界面

将鼠标放置在软件图标上面，会显示查看详情的图标，如下图所示：

图 4-9 查看详情

点击查看详情图标，显示该软件对应的功能信息，单击详细信息区域，详细信息栏会消失，如下图所示：

图 4-10 软件详细信息

双击软件图标，会启动应用程序，（例如：可信保密箱），如下图所示：

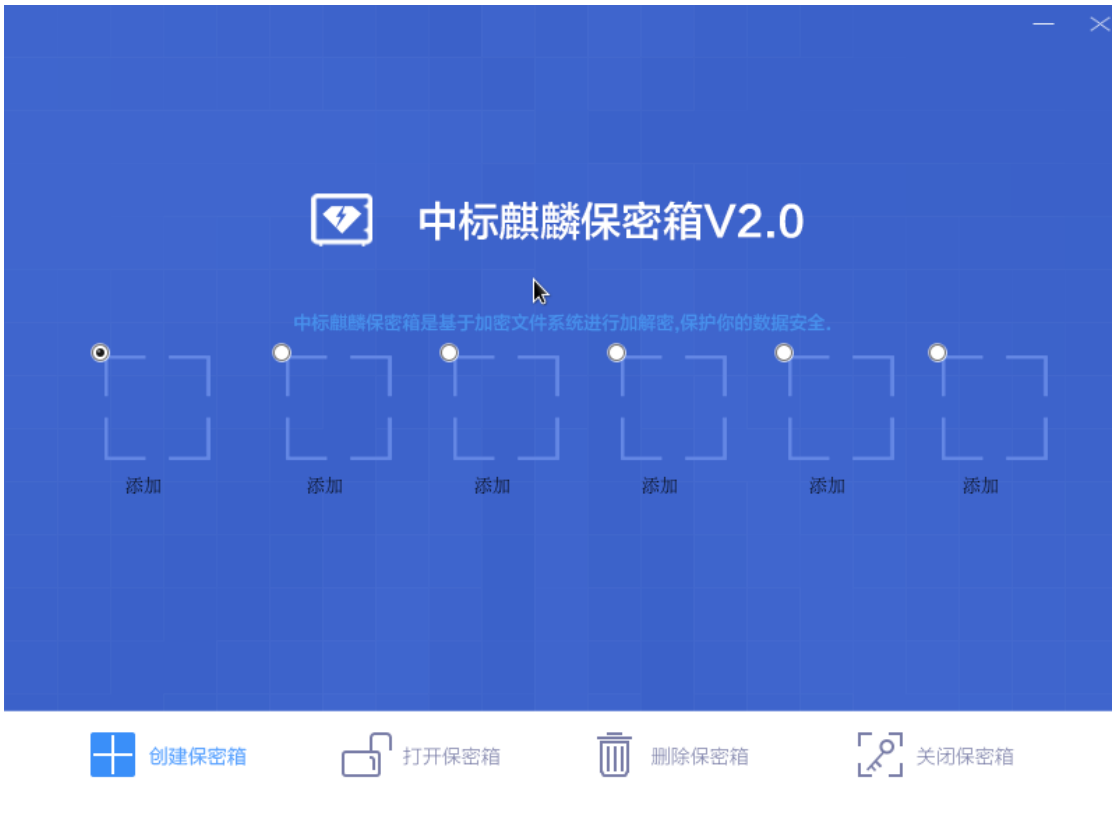


图 4-11 启动应用程序

如果是 root 以外的用户启动应用程序，则会弹出提权的提示框，输入正确的密码则可启动应用程序，如下图所示：

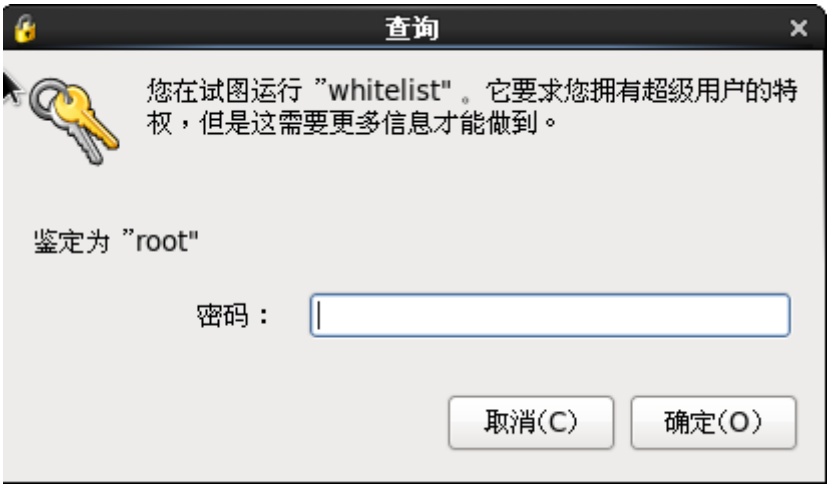


图 4-12 普通用户启动应用程序

右键单击应用程序图标，会显示右键菜单，包括“创建桌面快捷菜单”、“卸载”，选择“创建桌面快捷菜单”会在桌面上创建应用程序对应的快捷方式，用户可以直接双击快捷方式启动应用程序；选择“卸载”，则应用程序被卸载，图标消失，创建的快捷方式也会被删除，托盘中的最近打开也会删除该应用程序。除了安全管理员有卸载功能，其他用户只有创建快捷菜单功能。右键菜单如下图所示：

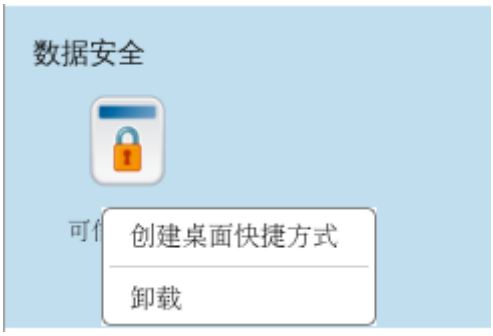


图 4-13 右键菜单

第 5 章 可信管理中心

1.使用说明

在已装好的系统中，用安全管理员登录，点击启动->系统工具->中标麒麟安全控制中心。弹出一界面，如下所示：

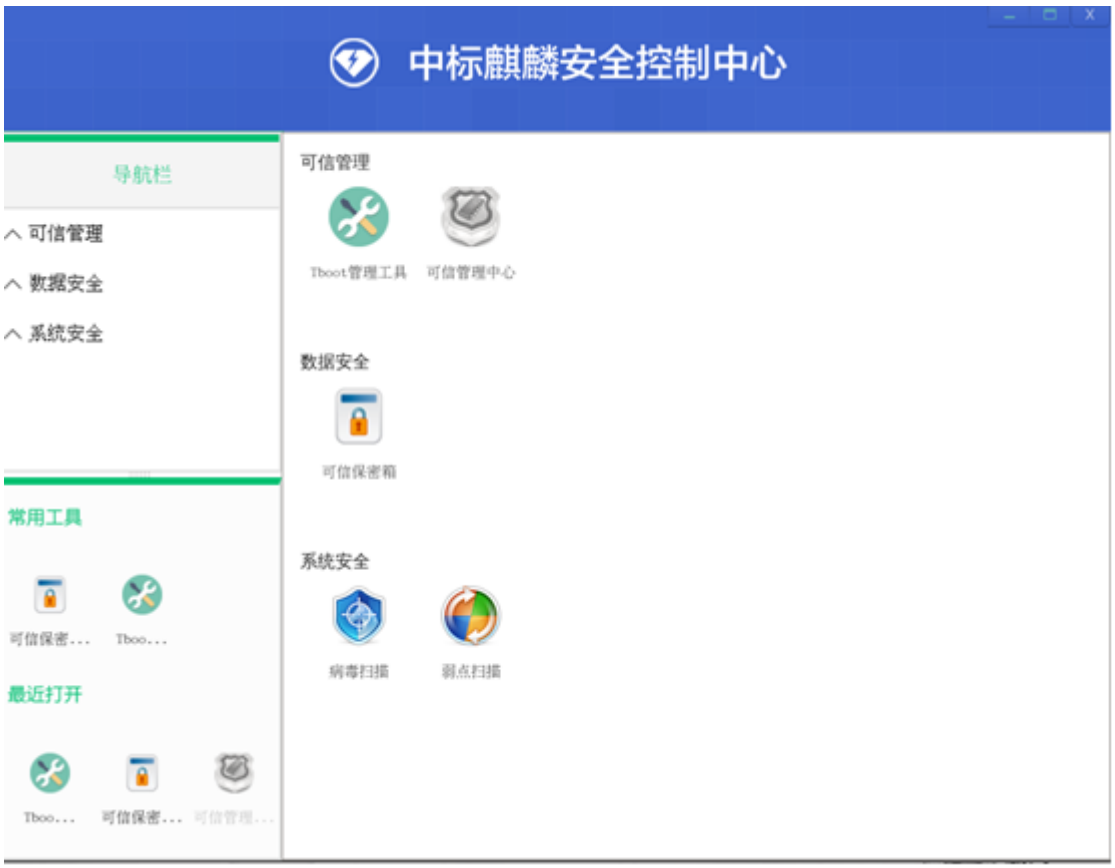


图 5-1 安全控制中心界面

单击可信管理中心图标即可。

2.功能介绍

可信管理中心的界面包含白名单管理，度量报告和设置三个功能模块。

2. 1 白名单管理

白名单管理包括的功能点有白名单扫描和查看白名单，如图 5-2 所示：

图 5-2 白名单管理主界面

1.白名单扫描

白名单扫描界面如图 5-3 所示：

图 5-3 白名单扫描

- 扫描区域

在图 5-2 中的扫描区域列出的是文件夹，可以勾选上复选框；然后点击显示的三角箭头获取该文件夹，则可以显示出该文件夹下存在的子文件夹。

点击“切换软件包”，则将扫描区域切换成软件包列表，如图 5-4 所示：

图 5-4 白名单扫描

- 扫描类型

在该区域中选择需要扫描的文件类型，可以进行多选。

➢ 开始扫描操作

当选择好了扫描区域和扫描类型后，点击“开始扫描”，则会进行扫描操作。如图 5-5 所示：

图 5-5 扫描操作

扫描状态栏： 当正在扫描的文件时，“暂停”操作按钮，“取消”操作按钮；扫描结束后，则有“导入数据库”操作按钮。

扫描过程： 正在扫描的过程会显示出扫描区域、扫描统计、状态。

- 扫描区域：扫描文件的路径。
- 扫描统计：统计扫描多少个文件。
- 状态：可导入白名单的文件个数。

筛选操作：可以按照类型或者状态进行筛选操作。

注 1：当选择的区域是文件夹时，扫描的是该文件夹及子文件夹中的文件；当选中的是软件包是，扫描的是该软件包中包含的文件。

注 2：当状态为已存在的项是无法选中的。

2.查看白名单

查看白名单界面如图 5-6 所示：

图 5-6 查看白名单

白名单信息窗体中以表格的形式显示白名单信息， 表格中包含以下字段：

- 文件名：白名单文件的绝对路径。
- 类型：白名单文件的类型(脚本文件，可执行文件，动态库，KO 模块)
- 状态：包含两种状态：“可信”，“不可信”
- 详细：详细信息中包含白名单的路径，类型，所属软件包，状态，权限，度量值。
- 日期：白名单扫描时间。

➢ 白名单删除

在白名单列表上右键可以删除选中项。或者选中白名单后点击右下角的删除按钮。

➢ 查看详情

在详细列表项中，点击“详情”，会弹出一个白名单详细的信息，如图 5-7 所示：

图 5-7 白名单详情

➢ 添加白名单

在查看白名单界面中（如图 5-6），点击添加白名单，会弹出一个添加白名单框，如图 5-8 所示：

图 5-8 添加白名单

➢ 清空白名单

点击“清空白名单”按钮，弹出“确定要全部删除白名单文件吗？”。点击确定后所有白名单文件被删除。

2.2 度量报告

度量报告窗体包含启动度量和度量统计两部分。

1.启动度量

启动度量主要显示上次度量的白名单的扫描区域，扫描项数统计和上次扫描时间。如图 5-9：

图 5-9 启动度量

2.度量统计

统计本次启动阶段的度量情况，可信的有多少项，不可信的有多少项，并提示用户本次启动是否可信。如图 5-10 所示：

图 5-10 度量统计

2.3 设置

设置功能块用于系统配置的查看和改变操作，包含基本设置、度量设置、审计设置、特权进程设置。

1.基本设置

设置可信模块的开启与关闭状态，如图 5-11 所示：

图 5-11 基本设置

可信模块设置对应的文件是/ctmm/enable，开启与关闭在该文件中的值是 1 和 0。

2.度量设置

设置脚本文件、可执行文件、动态库、KO 模块加载的策略。如图 5-12 所示：

图 5-12 度量设置

设置脚本文件、可执行文件、动态库、KO 模块加载分别对应的文件是/ctmm/hooks/file_hook 、 /ctmm/hooks/exec_hook 、 /ctmm/hooks/lib_hook 、/ctmm/hooks/mod_load_hook；关闭、宽松、严格分别在以上文件中记录的值是 0、1、2。

3.审计设置

对审计策略的设置。如图 5-13 所示：

图 5-13 审计设置

审计设置对应的文件是/ctmm/log，不审计、只审计失败、只审计成功、均审计在该文件中记录的值分别对应的是 0、1、2、3

4.添加特权进程

特权进程设置：此项功能可以单独添加某个进程，此进程不受度量设置的限制。如图 5-14：

图 5-14 特权进程设置

2.4 CTMM 命令

1、ctmm_get_stat

用途：获取 CTMM 模块的状态，包括各控制类型的状态

参数：无

2、ctmm_set_stat

用途：对 CTMM 模块进行开关设置

参数：

-e [0|1] 0:关闭 ctmm 1：开启 ctmm

-l [0|1|2|3] 设置日志记录级别 0：关闭日志 1：只审计拒绝信息 2：只审计允许信息 3：审计全部信息

-h 帮助

3、ctmm_label

用途：可对单个文件进行度量或以目录进行批量度量，当以目录进行批量度

量时，可按可执行文件、动态库、内核模块、脚本分类或组合进行度量，如不指定文件类型即默认全部类型，但不对内核进行更新。

参数：

-f [文件名] 单个文件进行度量指定文件路径

-d [目录] 批量度量指定目录

-D 度量默认目录

-E 批量度量时代表可执行文件

-L 批量度量时代表动态库

-K 批量度量时代表内核模块

-S 批量度量时代表脚本

-h 帮助

4、ctmm_load

用途：将最新的应用层白名单数据或配置数据加载到内核。

参数：

-p 加载白名单数据

-c 加载配置数据

-h 帮助

5、ctmm_add

用途：对单个文件加入白名单，并生效到内核。

参数：

-f [文件名] 所添加文件的路径

-h 帮助

6、ctmm_delete

用途：对单个文件或全部文件从白名单中移除，并生效到内核。

参数：

-f [文件名] 对单个文件移除

-a 对全部文件从白名单移除

-h 帮助

7、ctmm_export

用途：将白名单数据导出到文件

参数：

-f [文件名] 白名单导出的保存文件

-h 帮助

8、ctmm_import

用途：将外部白名单输入导入到系统 CTMM，不及时生效与 ctmm_label 类型，区别为白名单从外部导入的，而非系统扫描出的。

参数：

-f [文件名] 要导入到系统的外部白名单的路径

-h 帮助

9、ctmm_remeasure

用途：对系统中已有的文件进行重新度量，内核及时生效

参数：

-a 对白名单进行重新度量

-f [文件名] 对单个文件进行重新度量

-h 帮助

10、ctmm_lookup

用途：对白名单数据进行分类查找

参数：

-t [exe|lib|ko|script] 分类查找的白名单数据类型：exe 可执行文件，lib 动态库 ko 内核模块 script 脚本

-s [0|1] 分类查找的白名单数据状态：0 不可信 1 可信

-h 帮助

第 6 章 可信保密箱

中标麒麟可信保密箱是基于 ecryptfs 的加密文件系统进行加解密的。主要实现对文件的加密保护功能。保密箱只有在打开的状态下，保密箱中的文件才可以正常查看。保密箱在关闭状态时，保密箱中的文件处于加密状态，直接查看文件的时候会显示乱码。系统管理员，安全管理员和审计管理员都可以使用保密箱功能。

保密箱的功能有创建保密箱、打开保密箱、删除保密箱、关闭保密箱和鼠标右键将文件添加到保密箱，如图 6-1 所示。

图 6-1 保密箱

1.创建保密箱

创建保密箱的个数最多为 6 个，鼠标单击“创建保密箱”按钮即可弹出创建保密箱界面，如图 6-2 所示。

图 6-2 创建保密箱

创建保密箱时，类型有口令加密和 TPM 加密。输入的口令即为密码，最大长度 16 个字符。在高级设置中可以选择加密算法和密钥长度。默认加密算法为 aes，密钥长度为 16 位。输入保密箱信息，点击创建按钮，即可创建保密箱，如图 6-3 所示。对一个已经创建好的保密箱可以执行打开保密箱、删除保密箱和关闭保密箱操作。

图 6-3 输入保密箱信息

2.打开保密箱

选中要打开的保密箱，点击打开保密箱按钮，输入创建该保密箱时的口令，左键单击确认按钮即可打开该保密箱，如图 6-4 所示。在保密箱界面，锁变成打开的。然后将需要加密的文件放到弹出的以该保密箱命名的文件夹中即可。

图 6-4 打开保密箱

3.关闭保密箱

选中打开的保密箱，点击关闭保密箱按钮即可将该保密箱关闭。界面上，该保密箱上的锁变成关闭的锁，如图 6-5 所示。此时，保密箱中的文件已被加密。

图 6-5 关闭保密箱

4.删除保密箱

选中要删除的保密箱，左键单击删除保密箱按钮，在弹出的删除保密箱窗口中输入该保密箱的密码即可删除该保密箱。删除保密箱时保密箱要处于关闭状态，打开状态下保密箱不能被删除。

图 6-6 删除保密箱

5.添加到保密箱

在已经创建过保密箱的条件下，选中要添加到保密箱的文件，鼠标右键时菜单中出现“发送到保密箱”选项。在发送到保密箱选项中会列出已经创建好的保密箱。选择文件如放入的保密箱，在弹出的窗口中输入该保密箱的密码，点确认，如图 6-7 所示。密码正确且添加到该保密箱成功时，弹出添加到保密箱成功窗口，如图 6-8 所示。密码错误则弹出密码错误窗口，如图 6-9 所示。

图 6-7 输入保密箱密码

图 6-8 添加保密箱成功

图 6-9 密码错误

第 7 章 Tboot 管理工具

Tboot（Trusted Boot，可信启动）是一个启动过程度量软件，它利用 Intel 的 TXT 技术和可信平台模块(TPM)对系统内核或者 VMM 执行度量和验证的过程。

Tboot 主要包含以下六个功能：

1）度量启动：如果处理器支持 TXT 技术并且 TXT 功能开启，Tboot 将执行度量启动过程。如果度量启动流程失败，Tboot 将退出并执行非 TXT 启动过程。

2）度量环境的清除：当系统关机时，度量环境将被恰当地清除。Tboot 支持 S3/S4/S5 休眠状态。

3）数据重置保护：当系统重启时没有从内存中清除秘密数据，TXT 将阻止其他程序访问秘密数据。Tboot 通过对内存设置标志位来表明在度量启动过程中，内存是受保护的，而且在度量环境清除前会首先清除标志位。

4）TXT 内存范围的保护：TXT 保留了一定范围的 RAM 内存区域并定义了一些 MMIO 区域。这些内存被保护不能被任何 domains 域（包括 dom0）使用。

5）Intel TXT 启动控制策略(LCP)工具：lcptools 项目包括一组工具（和基础文档），可以用来创建并填充 LCP。LCP 使用 TPM 非易失性存储器（TPM NV）保存启动策略，SINIT AC 模块读取和使用这个策略启动度量启动环境（MLE）。

6）验证启动：Tboot 使用验证启动策略(类似于 LCP 并且也是存储在 TPM NV 空间中)，把验证过程从 MLE 扩展到内核/VMM 和 dom0。这些策略是用 tb_polgen 工具创建和管理，并用 lcptools 工具填充到 TPM NV 空间中。

1.软件启动

首次安装系统启动后，选择 “启动->系统工具->中标麒麟安全控制中心”，可以打开“中标麒麟安全控制中心”，双击可信技术下的“Tboot 管理工具”即可打开， 如下图。

图 7-1 Tboot 管理工具界面

2.创建策略

点击“创建策略”按钮， 如果执行成功，会生成策略并写入到可信芯片的 NV 存储中，并在 grub 启动项中加入 Tboot 的配置，下次启动系统时 Tboot 就会在加载内核之前执行。如下图。

图 7-2 创建策略

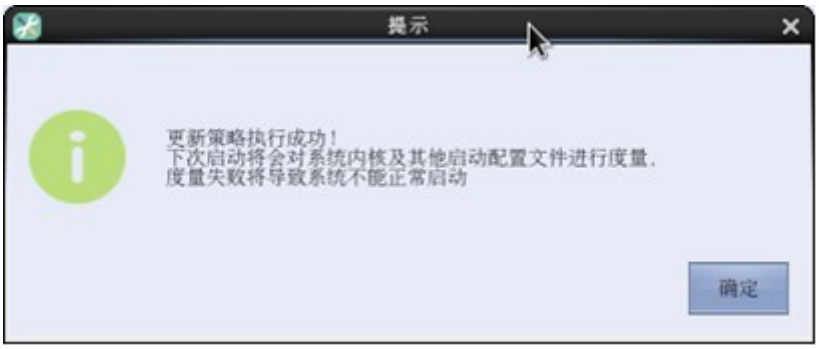


图 7-3 更新策略执行成功

3.更新策略

如果创建过策略之后，更新过内核，或者 Tboot 软件包，或者更改了 grub.conf 文件中的内核启动参数，这时应该更新策略，因为以上改动会导致度量值有变化。点击“更新策略”按钮，如果执行成功，会重新生成策略并写入到可信芯片的 NV 存储中。如下图。

图 7-4 更新策略

图 7-5 更新策略执行成功

4.清除策略

如果要在下次系统启动时去掉对 Tboot 的支持，可以点击按钮“清除”，执行成功后会去除 grub 启动项中的 Tboot 配置，并释放可信芯片中的 NV 空间。如下图。

图 7-6 清除策略

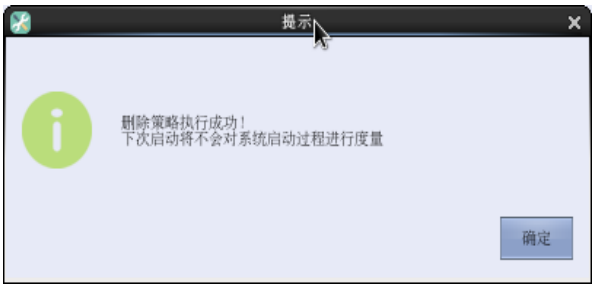


图 7-7 清除策略执行成功

5.查看日志

经过 Tboot 启动系统后，如果想要查看本次 Tboot 启动过程日志，

可以点击选项卡“状态查看”，点“查看”按钮。即可显示本次 Tboot 启动日志。如下图。

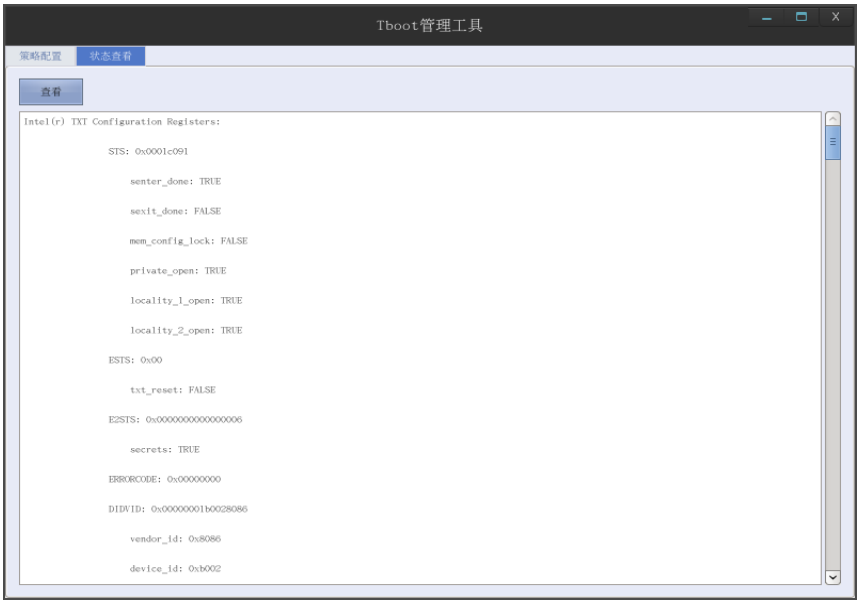


图 7-8 查看日志

6.策略配置

策略的一些默认配置保存在文件/usr/share/tbootlib/tbootconfig。可根据需要进行修改，如下图。

图 7-9 策略配置

启动过程中日志信息每页的显示停顿时间 vga_delay，默认是 1 秒。

策略类型默认是 nonfatal，即 Tboot 启动会忽略所有非致命错误，验证出错信息会显示在日志文件中。

当策略类型改为 continue 时，Tboot 启动过程会忽略验证错误，继续启动过程。

第 8 章 病毒扫描

1. 使用说明

在已装好的系统中，用系统管理员登录，点击启动->系统工具->中标麒麟安全控制中心。弹出一界面，如下所示：

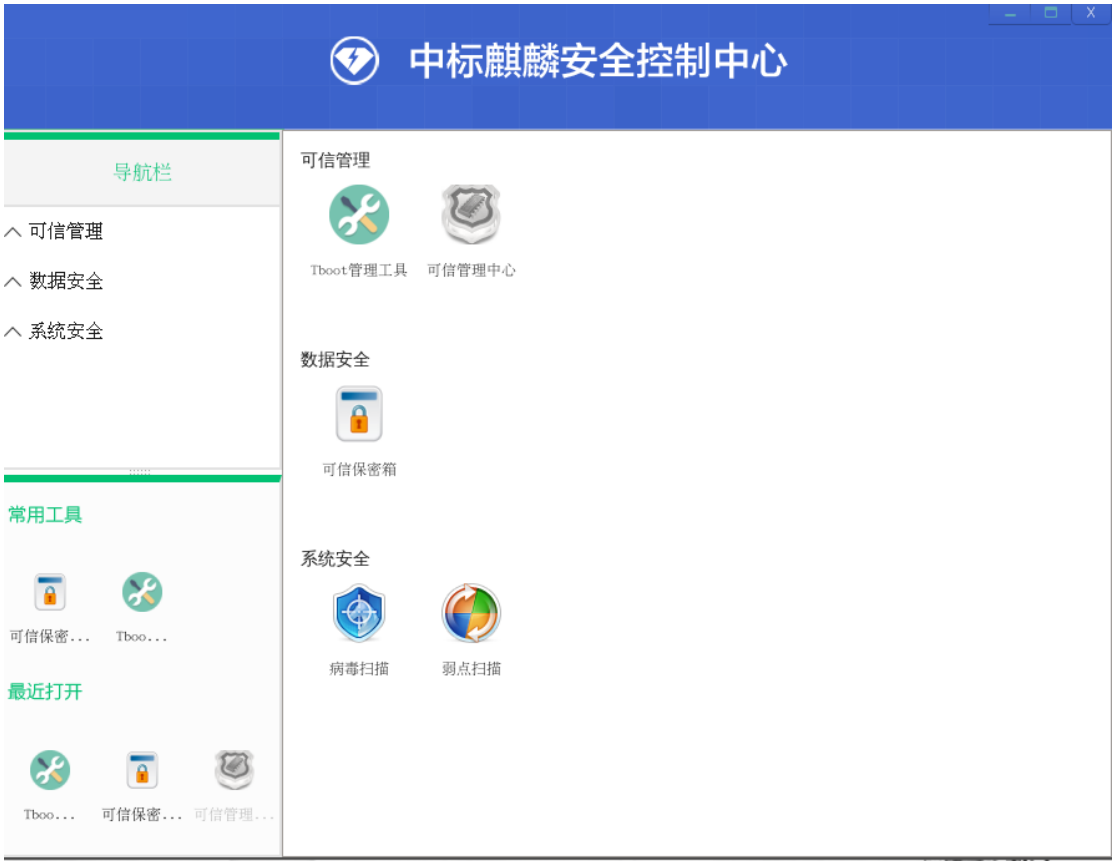


图 8-1 安全控制中心界面

单击病毒扫描图标即可。

图 8-2 病毒扫描主界面

2. 功能介绍

从 8-2 图可以看出病毒扫描首页主要功能有快速扫描、全盘扫描及自定义扫描，其他还有查看详情功能、更新病毒库功能、删除区及隔离区。下面会对这些功能逐一解释

2.1 快速扫描功能（推荐使用）

启动病毒扫描界面，在首页就可以看到“快速扫描“按钮，快速扫描主要针对系统常用文件进行扫描，对扫描过程进行分析及统计，如下图所示：

图 8-3 快速扫描界面

从上图可以看出，在快速扫描的过程中可以看到扫描的类型、开始扫描的时间、扫描已用的时间、扫描文件的数量、危险文件的个数以及扫描文件的内容。当完成扫描时会跳到首页，如果想查看扫描的结果，点击“查看详情“按钮，这个界面主要显示对扫描结果的统计，如果界面表格没有内容，说明没有危险文件存在。下面是一个扫描有危险文件的界面，

2.2 全盘扫描

全盘扫描主要对整个系统的文件进行扫描，占用时间较长，扫描过程跟快速扫描的过程很类似，主要区别在于扫描不同的文件。显示的结果和快速扫描也很相似，如下如所示：

图 8-4 全盘扫界面

2.3 自定义扫描

自定义扫描主要是针对指定的文件进行扫描，扫描过程跟以上的扫描过程是一样的。

总结，对于以上的扫描功能都有“取消“和“暂停/继续“按钮。

- 取消：如果正在对文件进行某种类型扫描，点击“取消“按钮弹出以下界面，如下图所示：

图 8-5 全盘扫描界面

从上图可以看出，可以进行“继续扫描“操作和“终止扫描“操作，终止扫描就是可以取消某种类型的扫描，取消完成后返回首页，或者可以点击”继续扫描“。

- 暂停/继续：如果正在对文件进行某种类型扫描，按钮显示暂停时，点击“暂停”按钮，暂停对文件扫描；按钮显示“继续“，点击“继续”按钮，继续对文件进行扫描。

2.4 查看详情

不管对文件进行某种类型的扫描，扫描的结果都会在查看详情界面展示，下面针对快速扫描进行描述，下面先看一张图，如下所示：

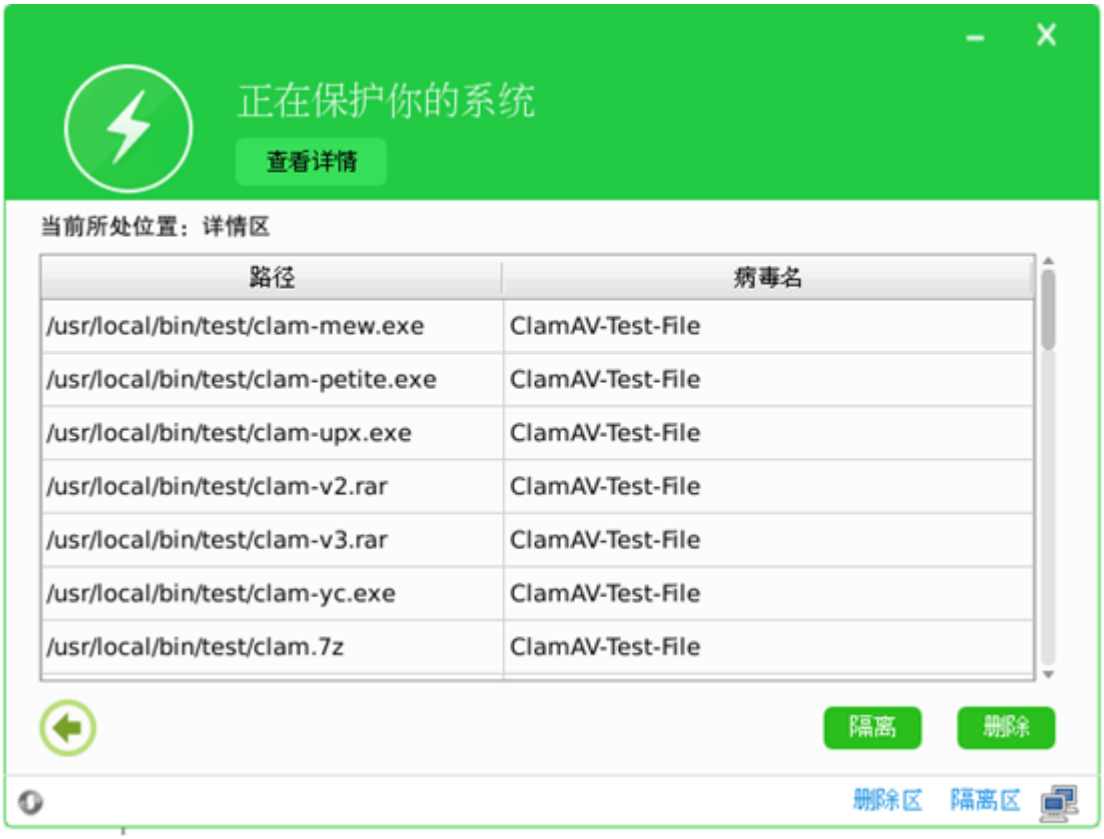


图 8-6 查看详情界面

这是一张快速扫描结果的查看详情界面，界面主要显示危险文件存在的路径及病毒名称，在这个界面当中还有“隔离“和“删除”功能。

- 隔离：对扫描的文件进行隔离处理，只是把文件隔离到某个文件下，方便以后可以进行还原操作。
- 删除： 对扫描的文件进行删除操作。

2.5 隔离区

在任务栏可以看到删除区和隔离区两个链接（蓝色字体），点击隔离区，显

示界面如下：

图 8-7 详情界面

这个界面主要跟查看详情界面有很大关系，隔离区主要显示文件是在查看详情界面隔离的文件，简单说就是对隔离文件的汇总。在隔离区还有两个功能，恢复和删除。

- 恢复：恢复功能主要恢复已经隔离的文件。
- 删除：删除已隔离的文件。（可以进行恢复操作）

2.6 删除区

删除界面主要是汇总查看详情界面和隔离界面的删除文件。简单来说就是汇总删除的文件。

图 8-8 删除区界面

- 恢复：恢复功能主要恢复已经删除的文件。
- 删除：彻底删除文件。

2.7 病毒更新

病毒更新功能是更新病毒库的文件可以更好的检测系统的安全性.如果检测到有新的病毒库版本,在任务栏就显示“病毒库有新版本,请更新”，如下图所示。点击左边的图片按钮即可更新.

图 8-9 任务栏病毒库更新提示

第 9 章 弱点扫描

1. 使用说明

在已装好的系统中，用系统管理员登录，点击：启动->系统工具->中标麒麟安全控制中心。弹出一界面，如下所示：

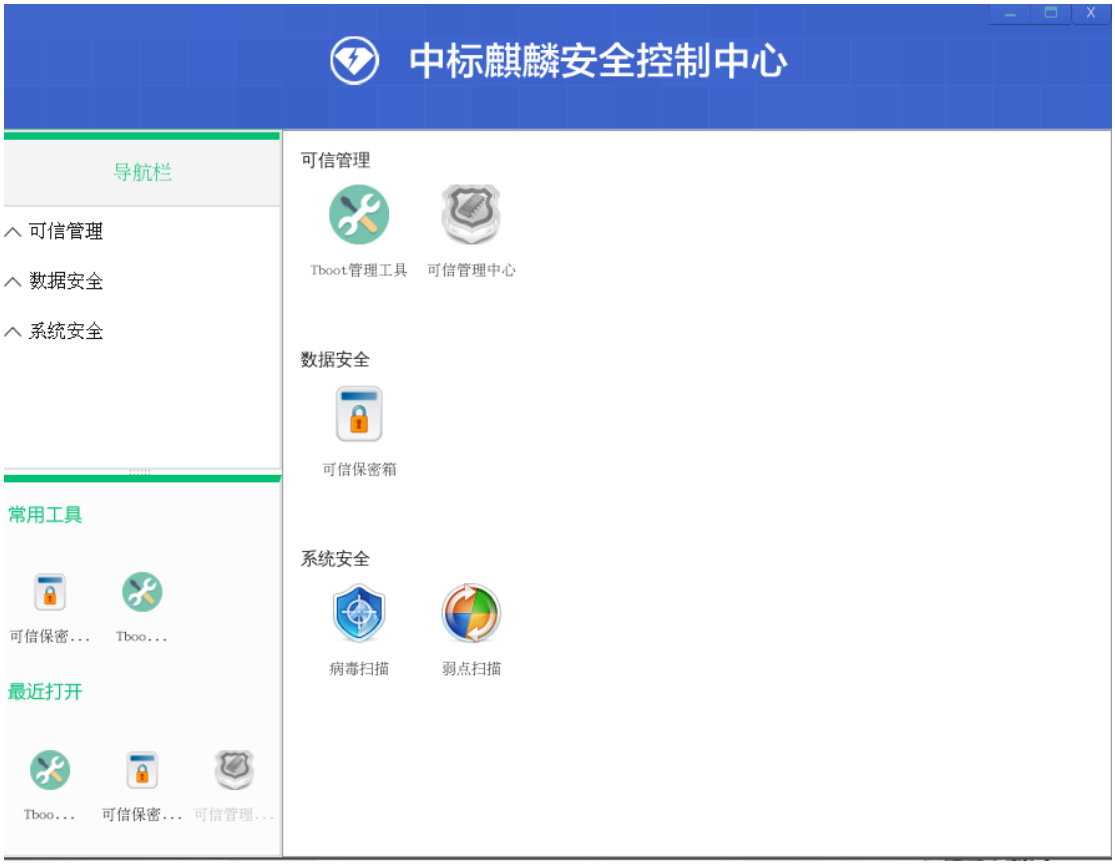


图 9-1 安全控制中心界面

单击弱点扫描图标即可。

图 9-2 弱点扫描主界面

弱点扫描首页只有“立即扫描“按钮，点击此按钮，即可进行弱点扫描。

2. 功能介绍

启动弱点扫描界面，如图 9-2，点击“立即扫描“按钮后，进入扫描界面，扫描界面显示如下：

图 9-3 弱点扫描界面

从上图可以看出,弱点扫描的项目有 grub 密码安全性、Iptable 开启状态、ssh 连接信息等等。扫描结果显示有扫描统计及扫描结果的显示，显示是否安全，相应的结果则在第三列显示。

备注：最后一列，如果按钮显示蓝色，表示有内容输出，如果为灰色，则没有信息显示，即无信息。

第 10 章 常用网络服务安全防护

1. 简介

作为服务器系统，它能够支持多种常用的网络服务，如 WWW 服务、Samba 服务、邮件（SMTP）服务、域名解析（DNS）服务等等。本章节分别对由 Apahce 搭建的 WWW 服务器、由 Sendmail 程序搭建的邮件服务器，以及 Samba 服务器做一些简单介绍。从而说明我们的服务器系统对各种网络服务具有良好的支持。

当系统被用作公共网络的服务器时，它就会成为攻击对象。正因此，对于系统管理员来说，加强系统防御和封闭某些服务就显得至关重要。

在深入研究具体问题之前，请回顾一下以下用来增强服务器安全性的常识：

1） 保持所有服务的更新状态来防御最新出现的威胁。

5） 尽量使用安全协议。

6） 在每台机器上尽量只使用一种网络服务的类型。

7） 密切监视所有服务器上的可疑活动。

2. 使用 TCP Wrappers 和 XINETD 来维护服务安全

TCP wrappers 为多项服务提供访问控制。多数现代的网络服务，如 SSH、Telnet 和 FTP，都使用 TCP Wrappers，它位于进入请求和被请求服务之间。

当与 xinetd 一起使用时，TCP Wrappers 的优越性就更为显著。xinetd 是一种提供附加的访问、记录、关联、重导向和资源利用控制的超级服务。

1） 设置陷阱

xinetd 的一个重要功能是把主机添加到全局 no_access 列表的能力。在这个列表上的主机到被 xinetd 管理的服务的后续连接都会被拒绝一段时间，直到 xinetd 被重新启动为止。这是通过使用 SENSOR 属性来实现的。该技术是阻塞试图扫描服务器端口的主机的简单方法。

设置 SENSOR 的第一个步骤是选择您不打算使用的服务。以下以 Telnet

为例进行说明。

编辑 /etc/xinetd.d/telnet 文件，把含有 flags 的行改成：

```
flags = SENSOR
```

在括号内添加以下行：

```
deny_time = 30
```

这会拒绝试图连接到端口的主机在今后 30 分钟内的所有连接。deny_time 属性还有一个可接受的值是 FOREVER，它会使该禁令在 xinetd 被重新启动前保持有效；NEVER 则会允许连接并且记录它。

最后一行应该是：

```
disable = no
```

虽然使用 SENSOR 是检测和阻止恶意主机的好办法。它有两个缺点：

a） 它对暗中扫描不起作用。

b） 知道 SENSOR 在运行的攻击者可以通过伪造 IP 地址和连接到禁用端口来对某些特定主机发起拒绝服务攻击。

8） 控制服务器资源

xinetd 的另一个重要功能是它能够控制从属服务可以利用的资源量。它通过以下指令来达到这个目的：

cps = <number_of_connections> <wait_period> — 指定每秒钟内被允许到服务的连接数量。该指令只接受整数值。

instances = <number_of_connections> — 指定允许到服务的连接总数。该指令接受整数值或 UNLIMITED。

per_source = <number_of_connections> — 指定每个主机被允许到服务的连接数量。该指令接受整数值或 UNLIMITED。

rlimit_as = <number[K|M]> — 指定服务可以占用的内存地址空间数量，以千字节或兆字节为单位。该指令接受整数值或 UNLIMITED。

rlimit_cpu = <number_of_seconds> — 指定服务占用 CPU 的时间(以秒为单位)。该指令接受整数值或 UNLIMITED。

使用这些指令有助于防止某个 xinetd 服务大量占用系统，从而导致“拒绝服务”情况的出现。

2.1. 使用 TCP WRAPPERS 来强化安全

2.1.1 TCP Wrappers 和连接标语

给连接服务的客户发送一幅警戒性标语是掩盖运行服务的系统的好办法，同时，它也让潜在的攻击者知道系统管理员是相当警惕的。要为某服务实现 TCP Wrappers 的标语，请使用 banner 选项。

这个例子为 vsftpd 实现了一个 banner。首先，创建一个 banner 文件。它可以位于系统上的任何地方，但是它的名称必须和守护进程相同。在这个例子中，该文件叫做 /etc/banners/vsftpd。

该文件的内容如下所示：

```
220-Hello， %c
220-All activity on ftp.example.com is logged.
220-Act up and you will be banned.
```

%c 代符提供了各类客户信息，如用户名和主机名，或用户名和 IP 地址，从而使连接更令人生畏。

要把这个标语展示给每个进入连接，把以下行添加到 /etc/hosts.allow 文件中：

```
vsftpd : ALL : banners /etc/banners/
```

2.1.2 TCP Wrappers 和攻击警告

如果某个主机或网络被发现正在攻击服务器，TCP Wrappers 可以通过 spawn 指令对来自该主机或网络的后续攻击向管理员发出警告。

在这个例子中，假定某个来自 206.182.68.0/24 网络的骇客被发现正在试图攻击服务器。如果把以下行添加到 /etc/hosts.deny 文件中，连接企图就会被拒绝并记录在一个特殊的文件中。

```
ALL : 206.182.68.0 : spawn /bin/ 'date' %c %d >> /var/log/intruder_alert
```

%d 代符提供攻击者其它访问的服务名称。

要运行连接并记录日志，把 spawn 指令放在 /etc/hosts.allow 文件中。

注意：因为 spawn 指令执行任何 shell 命令，您可以创建一个脚本，该脚本会在某个特定客户企图连接服务器的时候通知管理员或执行一系列命令。

2.1.3 TCP Wrappers 和强化记录

如果某类连接比其它连接更值得关注，您可以通过 severity 选项来提高该类服务的记录级别。

在这个例子中，假定每个企图连接 FTP 服务器的端口 23(Telnet 端口)的 客户都是骇客。在日志文件中放置一个 emerg 标记而不是默认的 info 标记来否定连接。

要达到这个目的，把以下行放在 /etc/hosts.deny 文件中：

```
in.telnetd : ALL : severity emerg
```

它使用默认的 authpriv 记录设施，但是把优先级别从默认的 info 提高到 emerg，这会把日志消息直接显示在控制台上。

2.2 使用 XINETD 来增强安全性

xinetd 超级服务器是另一个用来控制到其从属服务访问的有用工具。本节集中讨论如何使用 xinetd 来设置陷阱服务，以及如何使用它来控制任何给定 xinetd 服务可以使用的资源数量，从而阻挠拒绝服务攻击。要阅读更全面的可用选项列表，请参考 xinetd 和 xinetd.conf 的说明书页。

3. 保护 PORTMAP 的安全性

portmap 服务是用于 RPC 服务(如 NIS 和 NFS)的动态端口分配守护进程。它的验证机制比较薄弱，而且具备为它所控制的服务分配大范围端口的能力。由于这些原因，要保护它的安全比较困难。

设置 portmap 只对 NFSv2 和 NFSv3 起作用，NFSv4 不再使用这个服务。如果您想实现一个 NFSv2 或 NFSv3 服务器，您需要使用 portmap。

如果运行 RPC 服务，请遵守以下基本规则。

3.1 使用 TCP Wrappers 来保护 PORTMAP

使用 TCP Wrappers 来限制使用 portmap 服务的网络或主机，这一点很重

要，因为 portmap 没有内建的验证方式。

更进一步，在限制对服务的使用时，只使用 IP 地址。避免使用主机名，因为主机名可以通过 DNS 污染或其它方法被伪造。

3.2 使用 IPTABLES 来保护

要进一步限制对 portmap 服务的使用，在服务器上添加 IPTables 规则来限制到指定网络的进出是一个好办法。

以下是两个 IPTables 命令的例子，允许网络 192.168.0.0/24TCP 和 localhost(Nautilus 程序使用的 sgi_fam 服务所必需的)到 portmap 服务(监听端口 111)的连接。所有其它分组都被放弃。

```
iptables -A INPUT -p tcp -s! 192.168.0.0/24 --dport 111 -j DROP
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 111 -j ACCEPT
```

要以相似的方法限制 UDP 流量，使用以下命令。

```
iptables -A INPUT -p udp -s! 192.168.0.0/24 --dport 111 -j DROP
```

 注意：关于使用 IPTables 命令实现防火墙的详情，请参阅 “0 IPTables 的使用”。

4. 保护 NIS 的安全

NIS 代表网络信息服务(Network Information Service)。它是叫做 ypserv 的 RPC 服务，和 portmap 以及其它相关服务一起使用，被用来在声称属于 NIS 域内的计算机间分发关于用户名、口令、以及其它保密信息的映射表。

NIS 服务器包括几个应用程序。它们是：

/usr/sbin/rpc.yppasswdd — 又称 yppasswdd 服务，该守护进程允许用户改变他们的 NIS 口令。

/usr/sbin/rpc.ypxfrd — 又称 ypxfrd 服务，该守护进程负责在网络上传输 NIS 映射表。

/usr/sbin/yppush — 该应用程序把 NIS 数据库的改变传播给多个 NIS 服务器。

/usr/sbin/ypserv — 这是 NIS 服务器守护进程。

按照今天的标准来看，NIS 不太安全。它没有主机验证机制，在网络上明文传输所有的信息，包括口令散列。结果是，在设置使用 NIS 的网络时您必须格外谨慎。更糟糕的是，默认的 NIS 配置就有其固有的不安全性。

推荐任何打算实现 NIS 服务器的用户首先按照上面“保护 Portmap 的安全性”一节中的步骤来保护 portmap 服务的安全，然后再解决下面的问题(如网络规划)。

4.1 谨慎制定网络计划

因为 NIS 在网络上明文传输保密信息，所以让服务器在防火墙背后的一个分段的安全网络上运行就很重要。无论何时在不安全的网络上传递 NIS 信息都有被截取的危险。从这个角度讲，谨慎制定网络计划就有助于防御重要的安全破坏。

4.2 使用像口令一样的 NIS 域名和主机名

NIS 域内的任何机器都不经验证就可以使用命令从服务器中抽取信息，只要用户知道 NIS 服务器的 DNS 主机名和 NIS 域名即可。

例如：如果某人把便携电脑连接到网络上，或从外部闯入了网络(而且成功地假冒了内部 IP 地址)，以下命令会揭示 /etc/passwd 映射表：

```
ypcat -d <NIS_domain> -h <DNS_hostname> passwd
```

如果攻击者是一个 root 用户，他就可以通过键入以下命令来获得 /etc/shadow 文件：

```
ypcat -d <NIS_domain> -h <DNS_hostname>  shadow
```

 注意：如果使用了 Kerberos，/etc/shadow 文件就不会保存在 NIS 映射表中。

要使攻击者不能够轻易地获取 NIS 映射表，您可以为 DNS 主机名创建一个随机字符串，比如 o7hfawtgmhwg.domain.com。同理，您还可以创建一个不同的随机 NIS 域名。这就令攻击者进入 NIS 服务器比较困难。

4.3 编辑 /VAR/YP/SECURENETS 文件

如果 /var/yp/securenets 文件是空白的或不存在(按默认方式安装后的情形就会如此)，NIS 就会监听所有网络。它做的第一件事是在文件中放置一对子网掩码/网络值，因此 ypserv 只会对来自恰当网络的请求做出答复。以下是 /var/yp/securenets 文件中的示例项目：

```
255.255.255.0 192.168.0.0
```

 注意：在首次使用 NIS 服务器前，决不能没有创建 /var/yp/securenets 文件就启动它。

这种技术并不提供对 IP 假冒攻击的保护，但是它至少限制了 NIS 服务器要为哪些网络提供服务。

4.4 使用 IPTABLES 规则分配静态端口

所有和 NIS 相关的服务器都可以被分配给指定的端口，只有 rpc.yppasswdd 例外 — 该守护进程允许用户改变他们自己的登录口令。给其它两个 NIS 服务器守护进程，rpc.ypxfrd 和 ypserv 分配端口可以允许管理员创建防火墙规则来进一步保护 NIS 服务器守护进程免受入侵者的骚扰。

要达到这个目的，把以下几行添加到 /etc/sysconfig/network 中：

```
YPSERV_ARGS="-p 834"
YPXFRD_ARGS="-p 835"
```

以下 IPTables 规则可以被用来让服务器拒绝那些不信任网络连接服务器指定端口的操作。

```
iptables -A INPUT -p ALL -s! 192.168.0.0/24 --dport 834 -j DROP
iptables -A INPUT -p ALL -s! 192.168.0.0/24 --dport 835 -j DROP
```

4.5 使用 KERBEROS 验证

使用 NIS 验证的一个最明显的固有缺陷是，无论何时用户在机器上登录，/etc/shadow 映射表中的口令散列都会在网络上发送。如果入侵者获得了到 NIS 域的进入权，并且开始嗅探网络流量，他就可以悄悄地收集用户名和口令散列。

只有有足够的时间，口令破译程序就可以猜出薄弱的口令，攻击者就可以获得对网络上有效帐号的使用权。

由于 Kerberos 使用密钥加密术，口令散列就决不会在网络上传送，从而使系统更加安全。关于 Kerberos 的详情，请参阅网络服务用户手册中的“Kerberos”相关章节。

5. 保护 NFS 的安全

网络文件系统或 NFS 是和 portmap 以及其它相关服务一起使用的 RPC 服务，它被用来为客户机器提供可联网存取的文件系统。

注意：NFS 被包括在中标麒麟可信操作系统 V6.0 中，前面“2 保护 Portmap 的安全性”一节指出了 NFSv4 不再需要 portmap 服务。现在所用版本的 NFS 都倾向于使用 TCP 而不使用 UDP (在 NFSv4 中必须使用 TCP)。NFSv4 包括了 Kerberos 用户和组的认证做为 RPCSEC_GSS 内核模块的一部分。由于 NFSv2 和 NFSv3 还在使用 portmap，所以相关的信息仍然包括。

5.1 谨慎制定网络计划

现在 NFSv4 已经在网上传送使用 Kerberos 加密的信息，如果它是在防火墙背后或是在一个分段的网络上，小心地配置这个服务是非常重要的。NFSv2 和 NFSv3 数据在网络上的传输仍是不安全的，在进行网络配置的时候我们需要把它考虑进去。谨慎制定网络计划有助于防御安全破坏。

5.2 注意语法错误

NFS 服务器通过 /etc/exports 文件来决定要导出哪些文件系统，以及把这些目录导出到哪些主机上。编辑这个文件的时候要特别小心不添加额外的空格。如：/etc/exports 文件中的以下行会令主机 bob.example.com 可以读写地共享 /tmp/nfs/ 目录。

```
/tmp/nfs/ bob.example.com(rw)
```

但是 /etc/exports 文件中这一行的情况却不同。它共享同一目录，让主机 bob.example.com 拥有只读权限，却给全局以读写权限。这全是由主机后面的一个空格造成的！

```
/tmp/nfs/ bob.example.com (rw)
```

使用 showmount 命令来校验哪些目录被共享，这样可以检查您的 NFS 共享配置。

5.3 不要使用 NO_ROOT_SQUASH 选项

按照默认设置，NFS 共享把 root 用户改成用户 nfsnobody，它是一个不具备特权的用户帐号。这样，所有 root 用户创建的文件都会被用户 nfsnobody 所有，从而防止了设置 setuid 的程序被上传到系统。

如果使用了 no_root_squash，远程用户就能够改变共享文件系统上的任何文件，把设置了特洛伊木马的程序留给其它用户在无意中执行。

6. 保护 APACHE HTTP 服务器的安全

Apache HTTP 服务器是中标麒麟可信操作系统 V6.0 包括的最稳定和最安全的服务之一。保护 Apache HTTP 服务器安全的方法和技术多得数不胜数，无法逐一详述。在配置 Apache HTTP 服务器时阅读它的文档很重要，以下是管理员应该小心使用的一些配置选项。

6.1 FOLLOWSYMLINKS

该指令被默认启用，在创建到万维网服务器的文档根的符号链接时请小心。例如，提供一个到 / 的符号链接就不是个好主意。

6.2 INDEXES 指令

该指令被默认启用，但它可能不应该被启用。要阻止访问者浏览服务器上的文件，您可以删除该指令。

6.3 USERDIR 指令

UserDir 指令被默认禁用，因为它可以确认某个用户帐号在系统上是否存在。要启用服务器上的用户目录浏览，请使用以下指令：

```
UserDir enabled
UserDir disabled root
```

这些指令为除了 /root/ 以外的所有用户目录激活浏览。要把用户添加到禁用帐号列表中，在 UserDir disabled 行中添加一个用空格隔开的用户列表。

6.4 不要删除 INCLUDESNOEXEC 指令

按照默认设置，服务器端包括(server-side includes)模块不能执行命令。除非在极端必要的情况下，建议您不要改变这个设置，因为它有可能会使攻击者能够在系统上执行命令。

6.5 限制对可执行目录的权限

对于任何包含脚本或 CGI 的目录，请确定只给 root 用户以写权限。这可以通过键入以下命令来达到：

```
chown root <directory_name>
chmod 755 <directory_name>
```

还有，一定要在脚本实际使用之前校验它们在系统上的运行情况是否符合您的设想。

7. 保护 FTP 的安全

文件传输协议(FTP)是用来在网络上传输文件的早期 TCP 协议。因为所有服务器的事务，包括用户验证，都是不经加密的，所以它也被认为是不安全的协议，应该被谨慎配置。中标麒麟可信操作系统 V6.0 提供三种 FTP 服务器：

gssftpd — 使用了 Kerberos 的，基于 xinetd 的 FTP 守护进程。它不在网络中

传递验证信息。

内容加速器(Content Accelerator，tux)— 带有 FTP 能力的内核空间万维网服务器。

vsftpd — 一个独立的面向安全的 FTP 服务。

以下保安原则适用于设置 vsftpd FTP 服务。

7.1 FTP 问候标语

在提供用户名和口令前，所有的用户都会看到一个问候标语。按照默认设置，该标语含有版本信息，这有利于黑客找出系统的弱点。

要改变 vsftpd 的问候标语，把以下指令添加到 /etc/vsftpd/vsftpd.conf 中：

```
ftpd_banner=<insert_greeting_here>
```

把以上指令中的 <insert_greeting_here> 替换成问候消息。

对于多行标语，您最好是使用一个标语文件。要简化对多个标语的管理，把所有标语都放在一个叫做 /etc/banners/ 的新目录中。在这个例子中，FTP 连接的标语文件是 /etc/banners/ftp.msg。以下是这类文件的例子：

```
##########################################
# Hello， all activity on ftp.example.com#
# Is logged. #
##########################################
```

 注意：您不必让文件的每一行都以 220 开头，如前面的 0 一节中所指定。

要为 vsftpd 引用这个问候标语文件，把以下指令添加到 /etc/vsftpd/vsftpd.conf 中：

```
banner_file=/etc/banners/ftp.msg
```

您还可能使用 TCP Wrappers 给进入连接发送附加标语，如前面的 0 中所描述。

7.2 匿名访问

/var/ftp/ 目录的存在会激活匿名帐号。 创建这个目录的最简单方法是安装

vsftpd 软件包。该软件包会为匿名用户设置目录树，把目录的权限为匿名用户配置为只读。 按照默认设置，匿名用户不能写入任何目录。

 注意：如果启用到 FTP 服务器的匿名访问，您需要留意保密信息的存储位置。

匿名上传

要允许匿名用户上传文件，推荐您在 /var/ftp/pub/ 内创建只写目录。要做到它，键入：

```
mkdir /var/ftp/pub/upload
```

下一步，改变权限，因此匿名用户就看不到目录中的内容，键入：

```
chmod 730 /var/ftp/pub/upload
```

长格式的目录列表看起来应该像：

```
drwx-wx--- 2 root ftp 4096 Feb 13 20:05 upload
```

 注意：允许匿名用户在目录中读写的管理员经常发现他们的服务器成为被窃软件的仓库。

此外，在 /etc/vsftpd/vsftpd.conf 文件的 vsftpd 下，添加以下一行：

```
anon_upload_enable=YES
```

7.3 用户帐号

因为 FTP 在不安全的网络中传递明文用户名和口令来验证，拒绝系统用户从他们的用户帐号来进入服务器是一个好办法。

要在 vsftpd 中禁用用户帐号，在 /etc/vsftpd/vsftpd.conf 中添加以下指令：

```
local_enable=NO
```

约束用户帐号

禁用某组特定帐号(如 root 用户帐号，以及有 sudo 特权的用户帐号)的最简单方法是使用 PAM 列表文件，vsftpd 的 PAM 配置文件是 /etc/pam.d/vsftpd。您还可能在每个服务中直接禁用用户帐号。要在 vsftpd 中禁用指定用户帐号，把用户名添加到 /etc/vsftpd/ftpusers 中。

7.4 使用 TCP WRAPPERS 来控制访问

使用 TCP Wrappers 来控制到每种 FTP 守护进程的访问，如 0 中所概述。

8. 保护 SENDMAIL 的安全

Sendmail 是使用简单邮件传输协议(SMTP)的邮件传输代理(MTA)。它在其它 MTA 和电子邮件客户或分发代理之间传送电子消息。虽然一些 MTA 能够加密彼此间的交通，但多数不能够，因此通过公共网络发送邮件被认为是一种带有固有不安全因素的通信方式。

本节假定您已具备如何通过编辑 /etc/mail/sendmail.mc 和运行 m4 命令来生成有效的 /etc/mail/sendmail.cf 的基本知识。推荐任何打算实现 Sendmail 服务器的用户解决以下问题。

8.1 限制“拒绝服务”攻击

由于电子邮件的性质，一个决意要攻击某个服务器的攻击者可以轻易地使用邮件来充斥服务器，从而导致拒绝服务的攻击。通过设置 /etc/mail/sendmail.mc 的以下目录的限度，这类攻击的有效性就会大受限制。

ConnectionRateThrottle — 服务器每秒能够接受的连接数量。按照默认设置，Sendmail 不限制连接数量。如果限度被设置并到达，以后的连接就会被延迟。

MaxDaemonChildren — 服务器能够分出的子进程的最大数量。按照默认设置，Sendmail 不限制子进程的数量。如果限度被设置并达到，以后的连接就会被延迟。

MinFreeBlocks — 必须为服务器保留的用来接受邮件的空闲块的最少数量。默认为 100 块。

MaxHeadersLength — 消息头的可接受大小的最大限度(以字节为单位)。

MaxMessageSize — 单个消息的可接受大小的最大限度(以字节为单位)。

8.2 NFS 和 SENDMAIL

决不要把邮件假脱机目录 /var/spool/mail/ 放在 NFS 共享文件卷上。在 NFSv2 和 NFSv3 中，系统对用户组群 ID 没有控制。因此，几个 UID 相同的用户可以收到和阅读彼此的邮件。NFSv4 使用 Kerberos 而不是使用基于 UID 的用户认证。所以在 NFSv4 中就不会出现这种情况。

8.3 只使用电子邮件程序访问 SENDMAIL 服务器

要防止本地用户利用 Sendmail 服务器上的漏洞，最好是让邮件用户只使用电子邮件程序来访问 Sendmail 服务器。邮件服务器上的 Shell 帐号不应该被允许，/etc/passwd 文件中的所有用户 shell 都应该被设置为 /sbin/nologin(root 用户可能是个例外)。

9. 校验哪些端口正在监听

配置了网络服务之后，关注一下哪些端口在监听系统的网络接口这一点很重要。任何打开的端口都可能是入侵的证明。

要列举正在监听网络的端口，有两种基本方法。一种不太可靠的方法是通过键入 netstat -an 或 lsof -i 之类的命令来查询网络堆栈。这种方法之所以不太可靠是因为这些程序不连接网络上的机器，而是查看系统上在运行什么。因此，它们频繁成为攻击者的替换目标。骇客在打开了未经授权的网络端口后，就以这种方法来企图掩盖他们的踪迹。

更可靠的方法是使用 nmap 之类的端口扫描器来检查哪些端口正在监听网络。

以下从控制台发出的命令会判定哪些端口在监听来自网络上的 TCP 连接：

```
nmap -sT -O localhost
```

该命令的输出和以下相似：

```
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2010-09-24 13:49 EDT
Interesting ports on localhost.localdomain (127.0.0.1):
```

```
(The 1653 ports scanned but not shown below are in state: closed)
22/tcp open ssh
25/tcp open smtp
111/tcp open rpcbind
113/tcp open auth
631/tcp open ipp
834/tcp open unknown
2601/tcp open zebra
32774/tcp open sometimes-rpc11
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.5.25 - 2.6.3 or Gentoo 1.2 Linux 2.4.19 rc1-rc7)
Uptime 12.857 days (since Sat Sep 11 17:16:20 2010)
Nmap run completed -- 1 IP address (1 host up) scanned in 5.190 seconds
```

该输出显示了由于 sunrpc 服务的存在，系统正在运行 portmap。然而，端口 834 上还有一个神秘服务。要查看一下该端口是否和任何已知服务相关，键入：

```
cat /etc/services | grep 834
```

该命令没有返回任何输出。这表明虽然该端口是在保留范围内(即从 0 到 1023 内)，并且需要根权限才能打开，它并没有关联任何已知服务。

下一步，检查使用 netstat 或 lsof 的端口的信息。要使用 netstat 检查端口 834，使用以下命令：

```
netstat -anp | grep 834
```

该命令返回以下输出：

```
tcp 0 0 0.0.0.0:834 0.0.0.0:* LISTEN 653/ypbind
```

这个开放端口在 netstat 中存在，这一点比较令人安慰，因为如果骇客在被攻击的系统上暗中打开一个端口，他们很可能不会让这个端口使用该命令被暴露出来。还有，[p] 选项揭示了打开这个端口的进程 id(PID)。在这个例子中，被打开的端口属于 ypbind(NIS)，这是和 portmap 服务一起进行的 RPC 服务。

lsof 命令揭示了相似的信息，因为它也能够链接开放端口和服务：

```
lsof -i | grep 834
```

以下是这个命令中和讨论有关的输出部分：

```
ypbind 653 0 7u IPv4 1319 TCP *:834 (LISTEN)
ypbind 655 0 7u IPv4 1319 TCP *:834 (LISTEN)
ypbind 656 0 7u IPv4 1319 TCP *:834 (LISTEN)
ypbind 657 0 7u IPv4 1319 TCP *:834 (LISTEN)
```

这些工具揭示了大量关于运行在机器上的服务状态的信息。它们很灵活，能够提供关于网络服务和配置的许多信息。强烈推荐您阅读 lsof、netstat、nmap 和 services 的说明书页。

第 11 章 IPTABLES 的使用

Linux 内核中有一个功能强大的联网子系统 netfilter。netfilter 子系统提供了有状态的或无状态的分组过滤，还提供了 NAT 和 IP 伪装服务。netfilter 还具备为高级选路和连接状态管理而变形(mangle)IP 头信息的能力。netfilter 是通过 IPTables 工具来控制的。

IPTables 作为安装在 linux 上的网络过滤器，替代了 2.4 以前内核中使用的 ipchains，并在网络包的过滤范围和对其控制上具有更强大的功能扩展。

本章内容主要为：一些信息包过滤的基础知识；ipchains 和 iptables 的区别；iptables 命令的参数选项；如何在系统重启过程中保存过滤规则等。

注意：2.6 内核下的默认防火墙机制是 iptables，但 iptables 不能与 ipchains 同时运行。如果系统引导时，当前采用的是 ipchains，系统会报错并无法启动 iptables 。但正在运行的 ipchains 功能不会受到该错误报警的影响。

1. IPTABLES 总览

Netfilter 的强大功能和灵活性是通过 iptables 界面来实现的。这个命令行工具和它的前身 ipchains 的语法很相似；不过，iptables 使用 Netfilter 子系统来增进网络连接、检验、和处理方面的能力；ipchains 使用错综复杂的规则集合来过滤源地和目的地路线以及两者的连接端口。iptables 只在一个命令行界面中就包括了更先进的记录方式；选路前和选路后的行动；网络地址转换；以及端口转发。

2. 使用 IPTABLES

使用 IPTables 的第一步是启动 IPTables 服务。这可以使用以下命令进行：

```
service iptables start
```

注意：您应该使用以下命令关闭 IP6Tables 服务才能使用 IPTables 服务：

```
service ip6tables stop
chkconfig ip6tables off
```

要使 IPTables 在系统引导时默认启动，您必须使用 chkconfig 来改变服务的运行级别状态。

```
chkconfig --level 345 iptables on
```

IPTables 的语法被分成几个层次。主要层次为“链”(chain)。“链”指定处理分组的状态。其用法为：

```
iptables -A chain -j target
```

-A 在现存的规则集合内后补一条规则。chain 是规则所在“链”的名称。IPTables 中有三个内建的链(即影响每一个在网络中经过的分组的链)：INPUT、OUTPUT、和 FORWARD。这些链是永久性的，不能被删除。

-j target 选项指定 iptables 应该“跳”(jump)到规则集中的哪条规则。内建的目标有：ACCEPT、DROP、和 REJECT。

-N 选项可以被用来创建新链(又称用户定义链)。创建新链在定制详尽或复杂规则方面很有用。

2.1 基本防火墙策略

在一开始就建立的某些基本策略为建构更详细的用户定义的规则奠定了基础。IPTables 使用策略(policy， -P)来创建默认规则。对安全敏感的管理员通常想采取放弃所有分组、只逐一允许指定分组的策略。以下规则阻塞网络上所有的出入分组。

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

此外，还推荐您拒绝所有转发分组(forwarded packets) — 要从防火墙被选路发送到它的目标节点的网络交通 — 以便限制内部客户对互联网的无心暴露。要达到这个目的，使用以下规则：

```
iptables -P FORWARD DROP
```

 注意：在处理添加的规则时，REJECT(拒绝)目标和 DROP(放弃)目标这两种行动有

所不同。REJECT 会拒绝目标分组的进入，并给企图连接服务的用户返回一个 connection refused 的错误消息。DROP 会放弃分组，而对 telnet 用户不发出任何警告；不过，为了避免导致用户由于迷惑不解而不停试图连接的情况的发生，推荐您使用 REJECT 目标。

设置了策略链后，为您的特定网络和安全需要创建新规则。以下各节概述了一些您在建构 IPTables 防火墙时可能要实现的规则。

2.2 保存和恢复 IPTABLES 规则

防火墙规则只在计算机处于开启状态时才有效。如果系统被重新引导，这些规则就会自动被清除并重设。要保存规则以便今后载入，请使用以下命令：

```
/sbin/service iptables save
```

保存在 /etc/sysconfig/iptables 文件中的规则会在服务启动或重新启动时(包括机器被重新引导时)被应用。

3. 常用 IPTABLES 过滤

把远程攻击者拒之—LAN 外是网络保安的一个重要方面。LAN 的完好性应该通过使用严格的防火墙规则来抵御蓄意不良的远程用户而被保护。但是，如果默认策略被设置为阻塞所有进入、输出、和转发的分组，防火墙/网关和内部 LAN 用户之间的通信就无法进行。要允许用户执行和网络相关的功能以及使用联网应用程序，管理员必须打开某些端口进行通信。

例如：要允许到防火墙上的端口 80 的通信，添加以下规则：

```
iptables -A INPUT -p tcp -m tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

这会允许用户浏览通过端口 80 通信的网站。要允许到安全网站(如 https://www.example.com/)的访问，您还必须打开端口 443。

```
iptables -A INPUT -p tcp -m tcp --sport 443 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

 注意：在创建 IPTables 规则集合时，记住规则的顺序是至关重要的。例如：如果

某个链指定了来自本地子网 192.168.100.0/24 的任何分组都应放弃，然后一个允许来自 192.168.100.13(在前面要放弃分组的子网范围内)的分组的链被补在这个规则后面(-A)，那么这个后补的规则就会被忽略。您必须首先设置允许 192.168.100.13 的规则，然后再设置放弃规则。要在现存规则链的任意处插入一条规则，使用 -I，随后是您想插入规则的链的名称，然后是您想放置规则的位置号码(1，2，3，...，n)。例如：

```
iptables -I INPUT 1 -i lo -p all -j ACCEPT
```

此规则被插入为 INPUT 链的第一条规则，它允许本地环回设备上的数据传输。

有时候，您可能会需要从 LAN 之外远程地进入 LAN。SSH 和 CIPE 之类的安全服务可以用于到 LAN 服务的加密远程连接。对于拥有基于 PPP 资源(如调制解调器池或批量 ISP 帐号)的管理员来说，拨号进入可以被用来安全地避开防火墙，因为调制解调器连接是直接连接，通常位于防火墙/网关之后。 然而，对于有宽带连接的远程用户来说，您就需要制定些特殊规定。您可以配置 IPTables 接受来自远程 SSH 和 CIPE 客户的连接。例如，要允许远程 SSH 访问，您可以使用以下规则：

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p udp --sport 22 -j ACCEPT
```

来自外部的 CIPE 连接请求可以使用以下命令来接受(把 x 替换成您的设备号码)：

```
iptables -A INPUT -p udp -i cipcbx -j ACCEPT
iptables -A OUTPUT -p udp -o cipcbx -j ACCEPT
```

CIPE 使用它自己的传输数据报(UDP)分组的虚拟设备，因此这条规则允许 cipcb 接口上的进入连接，而不是规定源地端口或目标端口(虽然它们可以被用来代替设备选项)。

这些规则允许个体系统的出入交通，如单个 PC 直接连接到互联网或防火墙/网关；然而，它们并不允许防火墙/网关之后的机器使用这些服务。要允许 LAN 使用这些服务，您可以使用带有 iptables 过滤规则的 NAT。

4. FORWARD 和 NET 规则

多数机构从它们的 ISP 处得到数量有限的可公开选路的 IP 地址。鉴于这种限额，管理员必须创建性地积极寻求分享互联网服务的方法，而又不必把稀有的 IP 地址分配给 LAN 上的每一台机器。使用专用 IP 地址是允许 LAN 上的所有机器正确使用内部和外部网络服务的常用方法。边缘路由器(如防火墙)可以接收来自互联网的报文，并把这些报文选路发送到正确的的 LAN 节点上；同时，防火墙/网关还可以把来自 LAN 节点的输出请求选路发送到远程互联网服务中。这种转发网络交通行为有时会很危险，特别是随着能够假冒内部 IP 地址、使远程攻击者的机器成为您 LAN 上的一个节点的现代攻击工具的出现。为防止此类事件的发生，iptables 提供了选路发送和转发策略，您可以实施它们来防止对网络资源的变相利用。

FORWARD 策略允许管理员控制分组可以被选路发送到 LAN 内的哪些地方。例如：要允许整个 LAN 的转发(假定防火墙/网关在 eth1 上有一个内部 IP 地址)，您可以设置以下规则：

```
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -o eth1 -j ACCEPT
```

该规则令防火墙/网关之后的系统能够进入内部网络。网关把 LAN 节点的分组选路发送到它的目的地，通过 eth1 设备传递所有分组。

注意：按照默认设置，中标麒麟可信操作系统中的 IPv4 策略禁用了对 IP 转发的支持，这会防止运行中标麒麟 Linux 的机器成为专用边缘路由器。

要启用 IP 转发，请运行以下命令：

```
sysctl -w net.ipv4.ip_forward=1
```

如该命令是通过 shell 提示运行，则其设置在重新引导后就不会被保存。可通过编辑 /etc/sysctl.conf 文件来永久性地设置转发。寻找并编辑以下行，把 0 改成 1：

```
net.ipv4.ip_forward = 0
```

执行以下命令来启用 sysctl.conf 文件中的改变：

```
sysctl -p /etc/sysctl.conf
```

这会允许 LAN 节点彼此通信；不过，它们没有被允许和外界(如互联网)通信。要允许带有专用 IP 地址的 LAN 节点和外部的公共网络通信，配置防火墙的 IP 伪装(IP masquerading)，这会把来自 LAN 节点的请求都伪装成防火墙的外部设备(在这个例子中是 eth0)的 IP 地址。

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

该规则使用 NAT 分组匹配表(-t nat)，并在防火墙的外部联网设备(-o eth0)上为 NAT 指定内建的 POSTROUTING 链(-A POSTROUTING)。POSTROUTING 允许分组在离开防火墙的外部设备时被改变。-j MASQUERADE 目标被用来使用防火墙/网关的外部 IP 地址来掩盖节点的内部 IP 地址。

如果您想让内部网络内的某个服务器能够被外部利用，您可以使用 NAT 内 PREROUTING 链的 -j DNAT 目标来指定该向哪个目标 IP 地址以及端口转发请求连接到内部服务的分组。例如，如果您想把进入的 HTTP 请求转发到 172.31.0.23 上的专用 Apache HTTP 服务器服务器系统，运行以下命令：

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
--to 172.31.0.23:80
```

该规则指定 NAT 表使用内建的 PREROUTING 链来仅把进入的 HTTP 请求转发到被列出的 IP 地址 172.31.0.23。

如果您的 FORWARD 链的默认政策是 DROP，您就必须后补一条规则来允许转发进入的 HTTP 请求，因此目标 NAT 选路才有可能。运行以下命令可以达到这个目的：

```
iptables -A FORWARD -i eth0 -p tcp --dport 80 -d 172.31.0.23 -j ACCEPT
```

该规则允许把进入的 HTTP 请求从防火墙转发到防火墙之后的 Apache HTTP 服务器服务器。

4.1 DMZ 和 IPTABLES

您还可以设置一些把报文发送到某些机器(如专用 HTTP 或 FTP 服务器)的选路规则，这些机器最好是位于停火区域(de-militarized zone，DMZ)的和内部

网络分开的机器。要设置一条把所有进入的 HTTP 请求都选路发送到 IP 地址为 10.0.4.2、端口为 80(LAN 192.168.1.0/24 范围之外)的专用 HTTP 服务器的规则，网络地址转换(NAT)会调用 PREROUTING 表来把这些分组转发到恰当的目的地：

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
--to-destination 10.0.4.2:80
```

使用这项命令，所有来自 LAN 以外的到端口 80 的 HTTP 连接都会被选路发送到和内部网络分离的另一个网络上的 HTTP 服务器上。这种网络分段会比允许到内部网络中的机器上的 HTTP 连接更安全。如果 HTTP 服务器被配置接受安全连接，那么端口 443 也必须被转发。

5. IPTABLES 和连接跟踪

iptables 包括一个模块，它允许管理员使用—连接跟踪(connection tracking)方法来检查和限制到内部网络中可用服务的连接。连接跟踪把所有连接都保存在一个表格内，它令管理员能够根据以下连接状态来允许或拒绝连接：

NEW — 请求新连接的分组，如 HTTP 请求。

ESTABLISHED — 属于当前连接的一部分的分组。

RELATED — 请求新连接的分组，但是它也是当前连接的一部分，如被动 FTP 连接，其连接端口是 20，但是其传输端口却是 1024 以上的未使用端口。

INVALID — 不属于连接跟踪表内任何连接的分组。

您可以和任何网络协议一起使用 iptables 连接跟踪的状态功能，即便协议本身可能是无状态的(如 UDP)。下面的例子显示的规则使用连接跟踪来只转发已建立连接的分组：

```
iptables -A FORWARD -m state --state ESTABLISHED，RELATED -j ACCEPT
```

6. IP6TABLES

下一代互联网协议 IPv6 的出现突破了 IPv4(或 IP)的 32 位地址限制。IPv6 支持 128 位地址，因此识别 IPv6 的载体网络就能够制定比 IPv4 更多的可选路地

址。

中标麒麟可信操作系统支持使用 Netfilter 6 子系统和 IP6Tables 命令的 IPv6 防火墙规则。使用 IP6Tables 的第一步是启动 IP6Tables 服务。它可以使用以下命令进行：

```
service ip6tables start
```

您必须使用如下命令关闭 iptables 服务才能专门使用 ip6tables 服务：

```
service iptables stop
chkconfig iptables off
```

要使 ip6tables 在系统引导时默认启动，使用 chkconfig 来改变服务的运行级别状态。

```
chkconfig --level 345 ip6tables on
```

其语法在各方面都和 iptables 相同，只不过 ip6tables 支持 128 位的地址。例如：在识别 IPv6 的网络服务器器上的 SSH 连接可以使用以下规则来启用：

```
ip6tables -A INPUT -i eth0 -p tcp -s 3ffe:ffff:100::1/128 --dport 22 -j ACCEPT
```

关于 IPv6 联网的详情，您还可参阅 IPv6 的信息页：http://www.ipv6.org/。

7. 包过滤(PACKET FILTERING)

Linux 系统内核具有内置的包过滤功能，2.6 内核的网络过滤由以下部分组成：

filter table — 用于处理网络包的缺省表。

nat table — 用于转发包以建立新的连接，或用于网络地址转换 Network Address Translation (NAT)。

mangle table — 用于一些特殊类型包的转换。

注意：除已经设定的内置过滤表外，可自行定制并将其保存在 /lib/modules/<kernel-version>/kernel/net/ipv4/netfilter/ 目录 (其中<kernel-version> 是系统内核的版本号)。

每个表有一组内置的—链，与 netfilter 应用于包的相关动作关联。

filter table 中的内置链如下:

2） INPUT — 应用于定向到主机(host)的网络包。

3） OUTPUT — 应用于本地生成的网络包

4） FORWARD — 应用于经由主机(host)传输的网络包。

nat table 中的内置链如下:

5） PREROUTING — 转换到达的网络包；

6） OUTPUT — 在发出前转换本地生成的网络包；

7） POSTROUTING — 在发出前转换网络包；

mangle table 中的内置链如下:

8） INPUT — 根据目标主机转换网络包；

9） OUTPUT — 在发出前转换本地生成的网络包；

10） FORWARD — 转换路径主机的网络包；

11） PREROUTING — 在发出前转换收到的网络包；

12） POSTROUTING — 在发出前转换网络包；

8. IPTABLES 中的命令选项

用于过滤包的规则是由有序排列的 iptables 命令组成的。在使用 iptables 命令时，有以下几个常规的描述包的称谓：

13） Packet Type — 定义该命令适用的包类型.

14） Packet Source/Destination — 定义适用包的来源或目的地址.

15） Target — 定义当包超越标准要求时所进行的动作.

在 iptables 中，参数选项必须根据整条规则的目的和条件关系按一定的逻辑顺序排列，这样才能成为一条有效的规则。本节将讨论一些常用的 iptables 命令选项。

8.1 IPTABLES 命令的语法结构

大多数 iptables 命令都依照下面的结构:

```
iptables [-t <table-name>] <command> <chain-name> <parameter-1> \ <option-1>
<parameter-n> <option-n>
```

<table-name> — 定义命令涉及的表

<command> — 命令所要执行的特定动作，如：追加、删除等，执行对象为由*<chain-name>* 所定义的规则

<chain-name> — 一组由 parameter 及其 option 构成的对，其中 option 规定当„包"符合相应规则时所执行的选项

根据目的的不同，iptables 命令的长度和复杂度是不同的。从链中删除一条规则的命令可能很短，但用于通过各种参数和选项的组合来从特定的子网中过滤包的命令可能会很长。需要注意的是，一些参数和选项的组合结果可能作为另一对参数和选项的输入需求。为了形成一条有效的命令语句，每一对参数和选项之间的关系和条件都应得到满足。

键入 "iptables –h"， 可列出和深入了解 iptables 命令的语法结构。

8.2 IPTABLES 的命令选项(COMMAND OPTIONS)

命令选项规定 iptables 执行相应的动作，每条命令只允许带一个选项。除 help 命令外，所有命令选项均用大写字母。

列举如下：

-A — 追加一条 iptables 规则在链的队尾，该命令适用于链中对规则顺序没有要求时；

-D — 根据序号删除链中的某条规则(如 5 表示链中的第 5 条规则)；也可键入规则的完整描述，命令将删除链中与之匹配的那条规则；

-E — 重命名一条指定的链，该命令对表的结构无影响；

-F — 清空所指定的链，既删除此链中的所有规则；如没有指定链，该命令会清空所有链中的所有规则；

-h — 列出命令的语法结构清单，同时包括参数和选项的摘要；

-I — 根据指定的序号(整数)在链中插入一条规则，如未说明数字，则插入链顶端；

注意：使用 *-A* 或 *-I* 选项时请注意，每条规则在链中的顺序非常重要，它决定了哪些规则应用于哪些类型的包。

-L — 列出指定链中的所有规则；不指定链和表的名称，则可列出默认表中所有链的所有规则，否则可按下面的格式执行：

```
iptables -L <chain-name> -t <table-name>
```

-N — 根据指定的名称建立一条新的链；

-P — 为指定链设置默认应用策略，以使那些通过的_包‘在没有与之匹配的规则时能够被依照默认的策略进行处理，比如接收或者丢弃；

-R — 替换链中的指定规则，规则的序号必须在链名称后给予指定(链中规则序列的起始编号由 1 开始)；

-X — 删除一条用户定义的链(系统内置的链不允许被删除)；

-Z — 使表中所有链的包计数器数据归零。

8.3 IPTABLES 的参数选项(PARAMETER OPTIONS)

在 iptables 语句中，参数选项同样重要：

-c — 重置一条规则的计数器，通过 PKTS 和 BYTES 两种设定以决定重置哪个计数器；

-d — 设置匹配某条规则的包的目的地，可以是主机名、IP 地址或网络地址；如为网络地址，需遵循以下格式：

N.N.N.N/M.M.M.M — 其中 N.N.N.N 是 IP 地址域、M.M.M.M 为子网掩码；

N.N.N.N/M —其中 N.N.N.N 是 IP 地址域、M 为位掩码；

-f — 将规则用于成碎片的包。

在后面加“!”，表示取反，将匹配那些没有碎片的包。

-i — 设置网络接口类型，如 eth0 或 ppp0； 在 iptables 中，这个参数选项仅应用于 filter 表中的 INPUT 和 FORWARD 链，以及 nat 和 mangle 表中的 PREROUTING 链。

该参数还支持以下的选择：

！— 取反；即将此条规则应用于与此字符串匹配的网络接口以外的其它网络。

+ — 所有与此字符串匹配的通配符；例如，参数 “-i eth+” 是将此条规则应用于所有以太网接口，而排除了其它网络接口(如 ppp0 等).

如果仅使用了

-i 参数而没有指定网络的类型，则该规则将应用于所有类型的接口。

-j — 当包与某一规则匹配时直接进入特定的操作对象。有效的操作对象包括标准选项(接收、丢弃、排队和返回等)，以及系统默认安装的“iptables RPM 包”中提供的扩展选项，如：日志、标记、拒绝等等。 也可以使匹配的包进入当前链以外的一个用户指定的链，以使包被应用其它的规则。

如没有指定相应的操作对象， 包将通过该条规则而不被实施任何动作，但该条规则的计数器将加 1。

-o — 设置网络出口规则，仅应用于 filter table 中的 OUTPUT 和 FORWARD 链，以及 nat table 和 mangle table 中的 POSTROUTING 链。参数选项与网络进口(-i)的相同。

-p — 设置规则的 IP 协议，如 icmp、tcp、udp，以及其它支持的类型。任何列在/etc/protocols 中的协议均可。如果缺省，则默认为所有协议。

-s — 设置包的源地址信息，用法同参数“-d ”。

8.4 IPTABLES 的匹配选项(MATCH OPTIONS)

不同的网络协议提供专门的匹配选项以供设置来完成使用该种协议匹配特定的数据包的功能。首先，在 iptables 命令中应定义协议的类型，如： -p tcp <*protocol-name*> (其中<*protocol-name*> 是目标协议)。

8.5 TCP 协议

以下匹配选项用于 TCP 协议 (-p tcp)：

--dport — 设置包的目的端口；可使用网络服务器名(如 www、smtp)、端口号、或端口号的范围来设置；查阅所用的服务器名或端口号，可到/etc/services file

中查看。注意：“--destination-port ”和“ --dport ”是同义的。

指定端口号范围时，两个数字间用冒号分隔 (:)，如 -p tcp --dport 3000:3200；最大有效范围是 0:65535 。

在 --dport 后使用惊叹号 (!) 用以匹配不使用这一服务器或端口号的所有其它“包”。

--sport — 设置包的源端口；用法同--dport；注意：“--source-port”和“--sport”是同义的。

--syn — 应用于所有起始通信的 TCP 包(通常称为 SYN 包)，而对于数据包除外；在--syn 后使用惊叹号 (!) 用以匹配所有非 SYN 包。

--tcp-flags — 接收含有规则中定义的特定“位(bit)”或“标记(flag)”的 TCP 包。有两个参数：第一个是掩码，设置包中要被检验的标记；第二个是需要匹配的标记。可用的标记有： ACK、FIN、PSH、RST、SYN、URG、ALL、NONE 。

如 -p tcp --tcp-flags ACK，FIN，SYN SYN 表示：匹配那些设置有 SYN ，而无 ACK 和 FIN 标记的 TCP 包。

在 --tcp-flags 后加惊叹号 (!) ，表示取反。

--tcp-option — 用以匹配那些包含已设置的特定选项的 TCP 包；同样可用惊叹号(!)取反。

8.6 UDP 协议

以下匹配选项用于 UDP 协议(-p udp)：

--dport — 设置 UDP 包的目的端口；可使用网络服务器名、端口号、或端口号的范围来设置；注意：“ --destination-port”和“-dport”是同义的。可参考上面“TCP 协议”中的相关使用。

--sport — 设置 UDP 包的源端口；可使用网络服务器名、端口号、或端口号的范围来设置；注意：“--source-port”和“--sport”是同义的。可参考上面“TCP 协议”中的相关使用。

8.7 ICMP 协议

以下选项用于 ICMP 协议(Internet Control Message Protocol) (-p icmp):

--icmp-type — 设置匹配 ICMP 的名称和数量;有效的 ICMP 名可通过 iptables -p icmp -h 命令列出。

9. 如何保存 IPTABLES 规则

我们用 iptables 命令所建立的规则先储存在内存中，如果在未真正保存之前就重启了计算机，规则将被丢失。 由于 netfilter 规则需要在系统重启的过程中一直保留，他们必须被保存，为此可用 root 用户的身份键入如下命令：

```
/sbin/service iptables save
```

该语句执行 iptables init script，既运行/sbin/iptables-save 程序，将当前的 iptables 配置写入 /etc/sysconfig/iptables，则 /etc/sysconfig/iptables 被保存为 /etc/sysconfig/iptables.save。

当系统重新启动时，iptables init script 通过/sbin/iptables-restore 命令即可再次应用保存在/etc/sysconfig/iptables 中的规则。

在将新的 iptables 规则正式加入/etc/sysconfig/iptables 文件以前，对其进行必要的测试是很有用的；还可以将另一个系统版本的该文件中的规则复制到 /etc/sysconfig/iptables 中，这样非常方便利于为提供多重服务的机器很快建立起一整套规则。

注意：如需向其它机器发布/etc/sysconfig/iptables 文件，键入/sbin/service iptables restart 以使新的规则生效。

10. IPTABLES 控制脚本(IPTABLESCONTROL SCRIPTS)

有两种基本的方法来控制 iptables：

安全层配置工具(Security Level Configuration Tool)— 一个为创建、激活和保存基础防火墙规则的图形化接口。

/sbin/service iptables <option> — 一个命令，root 用户可通过它的 initscript

实现激活、停止等一些控制 iptables 的功能。 其中的<option> 可从以下几个指令中选择：

start — 若一个防火墙已经被配置(既/etc/sysconfig/iptables 已经存在)， 所有正在运行的 iptables 会被中止然后执行/sbin/iptables-restore 重启。本 start 指令仅在 ipchains 核心模块未被载入时使用。

stop — 如果一个防火墙正在运行且内存已满，所有 iptables 及其帮助模块将被卸载。

如果/etc/sysconfig/iptables-config 配置文件中的 IPTABLES_SAVE_ON_STOP 被设置为“yes”， 那么当前规则被存入/etc/sysconfig/iptables ，其它存在的规则将被移入/etc/sysconfig/iptables.save。

restart — 如果一个防火墙正在运行且内存已满，若已经配置在/etc/sysconfig/iptables 中，则该防火墙将重启。start 指令仅在 ipchains 核心模块未被载入时使用。

如果/etc/sysconfig/iptables-config 配置文件中的 IPTABLES_SAVE_ON_RESTART 被设置为“yes”， 那么当前规则被存入/etc/sysconfig/iptables ，其它存在的规则将被移入/etc/sysconfig/iptables.save。

status — 在 shell 提示符下列出防火墙状态和所有活动规则的清单；如果没有被配置和装载的防火墙，则该选项将报告这一信息。

除非/etc/sysconfig/iptables-config 配置文件中的 IPTABLES_STATUS_NUMERIC 值为“yes”，该选项将列出规则的域和主机名。

panic — 清空所有的防火墙规则；所有配置表的应用策略被设置为 DROP。

save — 通过 iptables-save 将防火墙规则存入 /etc/sysconfig/iptables；请参阅“如何保存 iptables 规则”以获取更多信息。

附录一：安全管理员用户命令集

setenforce

开启和关闭 selinux 的强制模式

如：　setenforce　1　　开启强制模式（其他管理员都可执行）

　　setenforce　0　　关闭强制模式（只有安全管理员才能执行）

详细使用方法见在线帮助文档。

checkpolicy

将可读策略文件编译生成二进制策略文件，该文件和所在文件夹必须为 policy_src_t 类型。

如：checkpolicy -M policy.conf -o policy.24

详细使用方法见在线帮助文档。

load_policy

如：load_policy -bq(服务器) 更换内核中的安全策略，保持使用当前的 Boolean 值

详细使用方法见在线帮助文档。

chcon

修改文件和文件夹的安全上下文

如：chcon -t bin_t filename

详细使用方法见在线帮助文档。

chcat

修改文件和文件夹的敏感级

如：chcat　s0:c0.c255 filename

详细使用方法见在线帮助文档。

fixfiles

检查修复文件安全上下文

如：fixfiles -F restore filename

详细使用方法见在线帮助文档。

setfiles

初始化文件标记，检查标记正确性

如：setfiles /etc/selinux/mls/contexts/files/file_contexts /

详细使用方法见在线帮助文档。

 # restorecon

恢复默认文件安全上下文

如：restorecon -r filename

详细使用方法见在线帮助文档。

 # semanage

SELinux 管理工具命令

如：semanage user -l

semanage login -l

semanage user -m -R secadm_r secadm_u

详细使用方法见在线帮助文档。

 # semodule

管理 SELinux 策略模块

如：semodule -l

semodule -i modulename

详细使用方法见在线帮助文档。

 # setsebool

修改 SELinux 中的 bool 值来修改策略

如：setsebool xdm_sysadm_login＝1

附录二：系统调用参照表

一、进程控制			
名称	说明	名称	说明
fork	创建一个新进程	prctl	对进程进行特定操作
clone	按指定条件创建子进程	ptrace	进程跟踪
execve	运行可执行文件	sched_get_priority_max	取得静态优先级的上限
exit	中止进程	sched_get_priority_min	取得静态优先级的下限
_exit	立即中止当前进程	sched_getparam	取得进程的调度参数
getdtablesize	进程所能打开的最大文件数	sched_getscheduler	取得指定进程的调度策略
getpGID	获取指定进程组标识号	sched_rr_get_interval	取得按 RR 算法调度的实时进程的时间片长度
setpGID	设置指定进程组标志号	sched_setparam	设置进程的调度参数
getpgrp	获取当前进程组标识号	sched_setscheduler	设置指定进程的调度策略和参数
setpgrp	设置当前进程组标志号	sched_yield	进程主动让出处理器，并将自己等候调度队列队尾
getpid	获取进程标识号	vfork	创建一个子进程，以供执行新程序，常与 execve 等同时使用
getppid	获取父进程标识号	wait	等待子进程终止
getpriority	获取调度优先级	wait3	参见 wait
setpriority	设置调度优先级	waitpid	等待指定子进程终止

modify_ldt	读写进程的本地描述表	wait4	参见 waitpid
nanosleep	使进程睡眠指定的时间	capget	获取进程权限
nice	改变分时进程的优先级	capset	设置进程权限
pause	挂起进程，等待信号	getsid	获取会晤标识号
personality	设置进程运行域	setsid	设置会晤标识号
二、文件系统控制——读写操作			
名称	说明	名称	说明
fcntl	文件控制	lseek	移动文件指针
open	打开文件	_llseek	在 64 位地址空间里移动文件指针
creat	创建新文件	dup	复制已打开的文件描述字
close	关闭文件描述字	dup2	按指定条件复制文件描述字
read	读文件	flock	文件加/解锁
write	写文件	poll	I/O 多路转换
readv	从文件读入数据到缓冲数组中	truncate	截断文件
writev	将缓冲数组里的数据写入文件	ftruncate	参见 truncate
pread	对文件随机读	umask	设置文件权限掩码
pwrite	对文件随机写	fsync	把文件在内存中的部分写回磁盘
三、文件系统控制——系统操作			
access	确定文件的可存取性	getdents	读取目录项
chdir	改变当前工作目录	mkdir	创建目录
fchdir	参见 chdir	mknod	创建索引节点
chmod	改变文件方式	rmdir	删除目录
fchmod	参见 chmod	rename	文件改名
chown	改变文件的属主或用户组	link	创建链接
fchown	参见 chown	symlink	创建符号链接
lchown	参见 chown	unlink	删除链接
chroot	改变根目录	readlink	读符号链接的值

stat	取文件状态信息	mount	安装文件系统
lstat	参见 stat	umount	卸下文件系统
fstat	参见 stat	ustat	取文件系统信息
statfs	取文件系统信息	utime	改变文件的访问修改时间
fstatfs	参见 statfs	utimes	参见 utime
readdir	读取目录项	quotactl	控制磁盘配额
四、系统控制			
名称	说明	名称	说明
ioctl	I/O 总控制函数	alarm	设置进程的闹钟
_sysctl	读/写系统参数	getitimer	获取计时器值
acct	启用或禁止进程记账	setitimer	设置计时器值
getrlimit	获取系统资源上限	gettimeofday	取时间和时区
setrlimit	设置系统资源上限	settimeofday	设置时间和时区
getrusage	获取系统资源使用情况	stime	设置系统日期和时间
uselib	选择要使用的二进制函数库	time	取得系统时间
ioperm	设置端口 I/O 权限	times	取进程运行时间
iopl	改变进程 I/O 权限级别	uname	获取当前 UNIX 系统的名称、版本和主机等信息
outb	低级端口操作	vhangup	挂起当前终端
reboot	重新启动	nfsservctl	对 NFS 守护进程进行控制
swapon	打开交换文件和设备	vm86	进入模拟 8086 模式
swapoff	关闭交换文件和设备	create_module	创建可装载的模块项
bdflush	控制 bdflush 守护进程	delete_module	删除可装载的模块项
sysfs	取核心支持的文件系统类型	init_module	初始化模块
sysinfo	取得系统信息	query_modul	查询模块信息

		e	
adjtimex	调整系统时钟	*get_kernel_syms	取得核心符号，已被 query_module 代替
五、内存管理			
名称	说明	名称	说明
brk	改变数据段空间的分配	munmap	去除内存页映射
sbrk	参见 brk	mremap	重新映射虚拟内存地址
mlock	内存页面加锁	msync	将映射内存中的数据写回磁盘
munlock	内存页面解锁	mprotect	设置内存映像保护
mlockall	调用进程所有内存页面加锁	getpagesize	获取页面大小
munlockall	调用进程所有内存页面解锁	sync	将内存缓冲区数据写回硬盘
mmap	映射虚拟内存页	cacheflush	将指定缓冲区中的内容写回磁盘
六、网络管理			
名称	说明	名称	说明
getdomainname	取域名	sethostid	设置主机标识号
setdomainname	设置域名	gethostname	获取本主机名称
gethostid	获取主机标识号	sethostname	设置主机名称
七、Socket 控制			
名称	说明	名称	说明
socketcall	socket 系统调用	recvmsg	参见 recv
socket	建立 socket	listen	监听 socket 端口
bind	绑定 socket 到端口	select	对多路同步 I/O 进行轮询
connect	连接远程主机	shutdown	关闭 socket 上的连接
accept	响应 socket 连接请求	getsockname	取得本地 socket 名字

send	通过 socket 发送信息	getpeername	获取通信对方的 socket 名字
sendto	发送 UDP 信息	getsockopt	取端口设置
sendmsg	参见 send	setsockopt	设置端口参数
recv	通过 socket 接收信息	sendfile	在文件或端口间传输数据
recvfrom	接收 UDP 信息	socketpair	创建一对已连接的无名 socket
八、用户管理			
名称	说明	名称	说明
getUID	获取用户标识号	setreUID	分别设置真实和有效的用户标识号
setUID	设置用户标志号	getresGID	分别获取真实的，有效的和保存过的组标识号
getGID	获取组标识号	setresGID	分别设置真实的，有效的和保存过的组标识号
setGID	设置组标志号	getresUID	分别获取真实的，有效的和保存过的用户标识号
geteGID	获取有效组标识号	setresUID	分别设置真实的，有效的和保存过的用户标识号
seteGID	设置有效组标识号	setfsGID	设置文件系统检查时使用的组标识号
geteUID	获取有效用户标识号	setfsUID	设置文件系统检查时使用的用户标识号
seteUID	设置有效用户标识号	getgroups	获取后补组标志清单
setreGID	分别设置真实和有效的的组标识号	setgroups	设置后补组标志清单
九、进程间通信			
名称	说明		
ipc	进程间通信总控制调用		
十、进程间通信——信号			
sigaction	设置对指定信号的处理方法		
sigprocmas	根据参数对信号集中的信	*sigsetmask	用给定信号掩码替换现有阻塞信号掩

k	号执行阻塞/解除阻塞等操作		码，已被 sigprocmask 代替
sigpending	为指定的被阻塞信号设置队列	*sigmask	将给定的信号转化为掩码，已被 sigprocmask 代替
sigsuspend	挂起进程等待特定信号	*sigpause	作用同 sigsuspend，已被 sigsuspend 代替
signal	参见 signal	sigvec	为兼容 BSD 而设的信号处理函数，作用类似 sigaction
kill	向进程或进程组发信号	ssetmask	ANSI C 的信号处理函数，作用类似 sigaction
*sigblock	向被阻塞信号掩码中添加信号，已被 sigprocmask 代替		
十一、进程间通信——消息			
msgctl	消息控制操作	msgsnd	发消息
msgget	获取消息队列	msgrcv	取消息
十二、进程间通信——管道			
pipe	创建管道		
十三、进程间通信——信号量			
semctl	信号量控制	semop	信号量操作
semget	获取一组信号量		
十四、进程间通信——共享内存			
shmctl	控制共享内存	shmat	连接共享内存
shmget	获取共享内存	shmdt	拆卸共享内存