



中标软件

中标麒麟高级服务器操作系统 V6.0

系统安全用户手册

中标软件有限公司

2014 年 04 月 20 日

目录

1 安全问题概述.....	1
1.1 什么是计算机安全.....	1
1.2 对网络安全的威胁.....	1
1.2.1 不安全的体系.....	1
1.2.2 广播式网络.....	2
1.2.3 中央化的服务器.....	2
1.3 对服务器安全的威胁.....	2
1.3.1 未用的服务器和打开的端口.....	2
1.3.2 未打补丁的服务.....	3
1.3.3 管理疏忽.....	3
1.3.4 带有固有不安全因素的服务.....	3
1.4 对工作站和家用电脑安全的威胁.....	4
1.4.1 弱口令.....	5
1.4.2 有弱点的客户应用程序.....	5
2 中标麒麟高级服务器操作系统的安全.....	6
2.1 使用 TCP Wrappers 和 XINETD 来维护服务安全	6
2.1.1 使用 TCP Wrappers 来强化安全	7
2.1.2 使用 XINETD 来增强安全性.....	8
2.2 保护 Portmap 的安全性	9
2.2.1 使用 TCP Wrappers 来保护 Portmap	9
2.2.2 使用 IPTables 来保护	9
2.3 保护 NIS 的安全	9
2.3.1 谨慎制定网络计划.....	10
2.3.2 使用像口令一样的 NIS 域名和主机名	10
2.3.3 编辑/var/yp/securenets 文件	11
2.3.4 使用 IPTables 规则分配静态端口.....	11
2.3.5 使用 Kerberos 验证.....	11
2.4 保护 NFS 的安全	12
2.4.1 谨慎制定网络计划.....	12
2.4.2 注意语法错误.....	12
2.4.3 不要使用 no_root_squash 选项	12
2.5 保护 Apache HTTP 服务器的安全	13
2.5.1 FollowSymlinks	13
2.5.2 Indexes 指令	13

2.5.3 UserDir 指令.....	13
2.5.4 不要删除 IncludesNOEXEC 指令.....	13
2.5.5 限制对可执行目录的权限.....	13
2.6 保护 FTP 的安全.....	14
2.6.1 FTP 问候标语.....	14
2.6.2 匿名访问.....	15
2.6.3 用户帐号.....	15
2.6.4 使用 TCP Wrappers 来控制访问	15
2.7 保护 Sendmail 的安全	16
2.7.1 限制“拒绝服务”攻击	16
2.7.2 NFS 和 Sendmail	16
2.7.3 只使用电子邮件程序访问 Sendmail 服务器	17
2.8 校验哪些端口正在监听.....	17
3 SELinux 的使用	19
3.1 概述.....	19
3.1.1 运行 SELinux 的益处	20
3.1.2 示例.....	21
3.1.3 SELinux 的架构	22
3.1.4 其他操作系统上的 SELinux	22
3.2 SELinux 上下文	23
3.2.1 域迁移.....	25
3.2.2 进程的 SELinux 上下文	26
3.2.3 用户的 SELinux 上下文	26
3.3 Targeted 策略.....	27
3.3.1 限制进程.....	27
3.3.2 非限制进程.....	30
3.3.3 限制与非限制用户.....	33
3.4 使用 SELinux	35
3.4.1 SELinux 包	36
3.4.2 使用哪个日志文件.....	37
3.4.3 主配置文件.....	38
3.4.4 启用和禁用 SELinux	39
3.4.5 SELinux 模式	43
3.4.6 布尔变量.....	43
3.4.7 SELinux 上下文——标记文件	46

3.4.8 ile_t 和 default_t 类型	53
3.4.9 安装文件系统.....	54
3.4.10 挂载 SELinux 标记	57
3.4.11 信息收集工具.....	64
3.5 限制用户.....	66
3.5.1 Linux 与 SELinux 用户映射	67
3.5.2 限制新 Linux 用户: useradd.....	67
3.5.3 限制已有 Linux 用户: semanage 登录	68
3.5.4 更改默认映射.....	70
3.5.5 xguest: Kiosk 模式	71
3.5.6 用户执行应用程序的布尔值.....	71
3.6 sVirt.....	72
3.6.1 安全与虚拟化.....	73
3.6.2 sVirt 标记.....	74
3.7 故障排除.....	75
3.7.1 访问拒绝时所发生的情况.....	75
3.7.2 三种引起问题的最常见原因.....	76
3.7.3 修复问题.....	80
4 IPTables 的使用.....	94
4.1 IPTables 总览.....	94
4.2 使用 IPTables.....	94
4.2.1 基本防火墙策略.....	95
4.2.2 保存和恢复 IPTables 规则.....	95
4.3 常用 IPTables 过滤.....	96
4.4 Forward 和 NET 规则	97
4.4.1 DMZ 和 IPTables.....	99
4.5 IPTables 和连接跟踪.....	99
4.6 IP6Tables.....	100
4.7 包过滤(Packet Filtering)	100
4.8 IPTables 中的命令选项.....	101
4.8.1 IPTables 命令的语法结构.....	101
4.8.2 IPTables 的命令选项(command options)	102
4.8.3 IPTables 的参数选项(parameter options)	103
4.8.4 IPTables 的匹配选项(match options)	104
4.8.5 TCP 协议	104

4.8.6 UDP 协议.....	105
4.8.7 ICMP 协议.....	105
4.9 如何保存 IPTables 规则.....	105
4.10 IPTables 控制脚本(IPTables control scripts)	106
5 网络入侵检测系统.....	108
5.1 入侵检测系统.....	108
5.1.1 TripWire	108
5.1.2 RPM 作为一种 IDS.....	108
5.2 基于网络的 IDS	110
6 NMAP 的使用	112
6.1 用 NMAP 扫描主机	112
6.2 NMAP 的使用	112

1 安全问题概述

1.1 什么是计算机安全

近来企业为帮助商业更好的运转和记录个人信息,对功能强大的联网计算机的依赖性日益增强,许多围绕着网络和计算机安全的业务也渐渐形成。企业机构开始聘请安全专家来确切地评审系统以及定制适合其组织的解决方案。由于许多机构属于动态性质,工作人员要么本地要么远程地使用公司的信息技术资源,因此这些机构对安全计算环境的需求已愈来愈强烈。

不幸的是,多数机构(以及个体用户)把安全问题当成“马后炮”。安全问题常常会由于对增强功能、提高生成率以及预算的考虑而被忽略。正确的安全措施经常是根据事后调查—非法的入侵事件出现之后来实施的。安全专家们一致认为,在把网站连接到不信任的网络(如国际互联网)之前预先采取正确的措施是阻止入侵企图的有效方法。

计算机安全这个通用术语所涉及的计算和信息处理领域比较广泛。依赖于计算机系统和网络来处理日常商业事务和存取重要信息的行业把它们的数据当作公司总资产的重要组成部分。一些术语和衡量标准也渐渐开始出现在我们的日常商业用语中。例如:TCO (Total Cost Of Ownership, 总拥有成本)、QOS (Quality Of Service, 服务质量)。在这些衡量标准中,行业把数据完整性和高可用性方面也计算入他们的计划和进程管理费用中。在某些行业(如电子商务)中,数据的可用性和可信性可能会决定成败。

1.2 对网络安全的威胁

在配置网络的以下方面时,某些不良习惯会增加系统被攻击的危险性。

1.2.1 不安全的体系

被错误配置的网络是未经授权用户的主要入口。把一个基于信任的开放型本地网络向极不安全的互联网敞开就如同住在一个罪案重重的街区里却不封门闭户一样—在一段时间内可能什么也不会发生,但是最终总会有人要利用这个机会。

1.2.2 广播式网络

系统管理员经常忽略联网硬件在安全计划里的重要性。简单的硬件，如依赖广播或非转换的原理的集线器和路由器；这就是说，不管什么时候某个节点通过网络来传输数据，集线器或路由器都会广播这些数据分组，直到接收节点收到并处理这些数据为止。这种方法最容易被外界入侵者和本地未经授权的用户施行 ARP(Address Resolution Protocol, 地址解析协议)或 MAC(Media Access Control, 媒体访问控制)的地址假冒攻击。

1.2.3 中央化的服务器

另一种潜在的联网危险是对中央化计算系统的使用。许多企事业单位中最常使用的节省开支措施是把所有的服务都集中于单个功能强大的机器上。这种方法会很省事，因为它比多服务器的配置要更好管理，而且开支也少。然而，中央化的服务器也给网络带来了单一失效点的问题。如果中央服务器泄密了，这就会导致整个网络都变得完全无用，甚至造成数据被篡改或盗窃。在以上这些情况下，中央服务器就成为一个敞开的大门，允许任何人进入整个网络。

1.3 对服务器安全的威胁

服务器安全和网络安全同等重要，这是因为服务器中常常贮存着机构内部的大量重要信息。如果一个服务器泄密了，其中的所有内容都可以被黑客随心所欲地利用。以下各节讨论了一些主要问题。

1.3.1 未用的服务器和打开的端口

中标麒麟高级服务器操作系统的完整安装包括 2200 多个应用程序和库软件包。不过，多数服务器管理员并不打算安装发行版本中的每个软件包，而倾向于进行基本安装，再包括几个服务器程序。

系统管理员管理时常会发生的是，他们安装了操作系统却不关注那些被安装了的程序。这可能会导致问题，因为不需要的服务可能会被安装，并使用默认设置配置，而且可能还被默认启用。这就会导致不需要的服务，如 Telnet、DHCP 或 DNS 在服务器或工作站上运行，而管理员对此却一无所知。由此会带给该服务器不受欢迎的连接，甚至给黑客制造了一条潜在的路径。

1.3.2 未打补丁的服务

多数被包括在默认安装中的服务器程序是稳定的、被全面测试过的软件。在生产环境中使用了多年后，这些软件的源码已经被全面精化，许多软件中存在的错误已被发现并修正。

然而，世界上并不存在完美无缺的软件，万事都有提高的余地。除此之外，由于更新的软件在生产环境中的使用时间不长，或者因为它没有其它服务器软件那么流行，它可能没有像人们所期待的一样被全面测试过。

开发者和系统管理员经常会在服务器程序中发现可被当作漏洞而利用的程序错误，并在错误跟踪和安全相关的网站(如 Bugtraq 邮件列表：<http://www.securityfocus.com>)或计算机紧急响应组(CERT)的网站(<http://www.cert.org>)上公开这些信息。虽然这些机制是向社区发出安全弱点警告的有效方法，但它却要依靠管理员来及时地给各自的系统打补丁。如果骇客也能够获得同样的弱点跟踪服务，他们一有机会就会使用这些信息来攻击未加补丁的系统。优秀的系统管理应该时刻警惕、时刻跟踪错误、并且进行正确的系统维护来保证安全的计算环境。

1.3.3 管理疏忽

忘记给系统打补丁的管理员是威胁服务器安全的最大因素。据系统管理网络和安全学院(System Administration Network and Security Institute, SANS)调查，计算机安全弱点的主要导致原因是指派未经培训的人员来维护安全，并且不提供使其能够胜任工作的培训和时间。

某些管理员忘记了给他们的服务器和工作站打补丁，而另一些则忘记了查看来自系统内核或网络交通的日志消息。另一个常见错误是不改变服务的默认口令或密码。例如，某些数据库有默认的管理口令，因为数据库开发者假定系统管理员在安装后会立即改变这些口令。如果某个数据库管理员忘记了改变口令，甚至一个毫无经验的骇客也能够使用众所周知的默认口令来获得数据库的管理特权。这里仅列举了几种不够严谨的管理给服务器所带来的潜在威胁。

1.3.4 带有固有不安全因素的服务

如果所选的网络服务带固有的不安全因素，即便是警惕性最高的组织也可能

成为受害者。例如，许多服务是在用于可信任网络的假定条件下被开发的，然而，一旦这些服务可通过互联网被使用，这个假定条件就不适用了。互联网本身就带有固有的不可信任性。

还有一类不安全网络服务是那些需要用户名和口令来验证的服务。Telnet 和 FTP 就属于这类服务。如果分组嗅探软件正在监视远程用户和这类服务器间的通信，口令就能够被轻而易举地窃取。

以上提及的服务还会轻易地遭到安全行业称之为“中间人”(man-in-the-middle)的攻击。在这类攻击中，骇客会设计让网络上的一个已攻破的名称服务器指向自己的机器而不是实际的目标服务器来重新导向网络交通。一旦某人打开了一个到该服务器的远程会话，骇客的机器就会充当一个隐型导管，悄悄地在远程服务和毫无疑心的用户间截取信息。骇客可以用这种方法来收集管理性口令和原始数据，服务器和用户对此却一无所知。

另一类不安全服务是网络文件系统和信息服务，如 NFS 或 NIS。它们仅仅是为 LAN 而开发的，但是不幸的是，后来又被扩展来包括 WAN(以方便于远程用户)。按照默认设置，NFS 没有配置任何验证或安全机制来防止骇客挂载 NFS 共享，以便存取其中的数据。同样的，NIS 也有网络上的每个计算机都必须知道的重要信息，包括口令和文件权限。它们都存在于一个纯文本的 ACSII 或 DBM(由 ASCII 推导出的)数据库中。能够进入这个数据库的骇客就能够访问网络上的每个用户帐号，包括管理员的帐号。

按照默认设置，中标麒麟高级服务器操作系统关闭此类服务。然而，管理员经常会发现他们不得不使用这些服务，因此谨慎配置就至关重要。

1.4 对工作站和家用电脑安全的威胁

工作站和家用电脑对攻击者的吸引力可能不会像网络或服务器那么大，但是由于其上经常包含保密信息，如信用卡信息，这些机器也被系统骇客看好。工作站还能够被骇客在协调的攻击中被用作一从属机器，而其主人对此却一无所知。由于这些原因，了解工作站的弱点也能够帮助用户避免重新安装操作系统的麻烦。

1.4.1 弱口令

弱口令是攻击者获得对系统的访问权最简单的方法之一。

1.4.2 有弱点的客户应用程序

虽然管理员可能会有一个完全安全的、全面打过补丁的服务器，这并不意味着远程用户在进入它时就是安全的。例如，如果服务器提供经由公共网络的 Telnet 或 FTP 服务，攻击者可能会将纯文本用户名和口令在网络经过时劫获，然后使用这个帐号信息来进入远程用户的工作站。

甚至在使用安全协议如 SSH 的时候，如果远程用户没有时时更新他们的客户程序，他们也可能遭受某些攻击。例如，使用 SSH 第一版本的用户就有可能遭受来自蓄意不良的 SSH 服务器的 X 转发攻击。一旦连接到服务器上，攻击者就能够不动声色地截取客户通过网络进行的击键和鼠标点击。这个问题在 SSH 协议的第二个版本中被修正了，但是它却要依赖于用户自身的自觉性，看用户是否关注哪些应用程序有哪些弱点，并在必要时更新这些程序。

2 中标麒麟高级服务器操作系统的安全

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。
- 2) 尽量使用安全协议。
- 3) 在每台机器上尽量只使用一种网络服务的类型。
- 4) 密切监视所有服务器上的可疑活动。

2.1 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的服务之间。

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
```

这会拒绝在今后 30 分钟内试图连接到端口的主机的所有连接。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。但它有两个缺点：

- 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。

2) 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>` — 指定每秒钟内允许连接服务的数量。该指令只接受整数值。

`instances = <number_of_connections>` — 指定允许连接服务的总数。该指令接受整数值或 UNLIMITED。

`per_source = <number_of_connections>` — 指定每个主机允许连接服务的数量。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。

2.1.1 使用 TCP Wrappers 来强化安全

2.1.1.1 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保护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/

2.1.1.2 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 指令可以执行任何 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。

2.1.1.3 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。

2.1.2 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。

2.2 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。

2.2.1 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。

2.2.2 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 192.168.0.0/24 的 TCP 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，请参阅章节“4 iptables 的使用”。

2.3 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)。

2.3.1 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。

2.3.2 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。

2.3.3 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。

2.3.4 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
```

2.3.5 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。只要有足够的时间, 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。

2.4 保护 NFS 的安全

网络文件系统 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 被包括在中标麒麟高级服务器操作系统中，前面“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。

2.4.1 谨慎制定网络计划

现在 NFSv4 已经在上网传送使用 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。

2.4.2 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。

2.4.3 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。

2.5 保护 Apache HTTP 服务器的安全

Apache HTTP 服务器是中标麒麟高级服务器操作系统包括的最稳定和 safest 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。

2.5.1 FollowSymlinks

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

2.5.2 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。

2.5.3 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。

2.5.4 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。

2.5.5 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。

2.6 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。中标麒麟高级服务器操作系统提供三种 FTP:

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。

2.6.1 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 2.1.1.1 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 2.1.1.1 中所描述。

2.6.2 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
```

2.6.3 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，需要把用户名添加到/etc/vsftpd/ftpusers 中。

2.6.4 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，如错误!未找到引用

源。中所概述。

2.7 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。推荐打算实现 Sendmail 服务器的用户解决以下问题。

2.7.1 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。

`confCONNECTION_RATE_THROTTLE` — 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。

`confMAX_DAEMON_CHILDREN` — 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。

`confMIN_FREE_BLOCKS` — 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。

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

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

2.7.2 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。

2.7.3 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。不应该允许邮件服务器上的 Shell 帐号，/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)。

2.8 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，就企图以这种方法来掩盖他们的踪迹。

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 5.21 ( 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。

3 SELinux 的使用

3.1 概述

安全增强 Linux(SELinux)是 Linux 内核中强制访问控制机制的实现，对标准的自主访问控制下允许的操作进行检查。SELinux 是由美国国家安全局创建的，能够基于预定义的策略实施 Linux 系统中文件、进程及其动作的规则。

当使用 SELinux 时，文件（包括目录和设备等在内）被称为客体。而进程，如用户运行一个命令或 Mozilla Firefox 应用程序，都被定义为主体。多数操作系统使用自主访问控制(DAC)系统来控制主体如何与客体进行交互以及主体之间如何相互交互。在使用 DAC 的操作系统上，用户控制他们所拥有的文件（客体）的权限。例如：在 Linux 操作系统上，用户可以让其家目录在任意范围内都是可读的，使所有用户和进程（主体）可以访问其潜在的敏感信息，而对非法动作没有任何进一步的限制。

只依靠 DAC 机制是不能满足更高的系统安全要求的。DAC 访问决策只基于用于身份和所有权，忽略其他诸如用户角色、程序的功能和可信度、数据的敏感性和完整性等与安全相关的信息。每个用户对于他们的文件具有完整的处理权，这很难加强系统内的安全策略。而且，一个用户运行的所有程序都继承了该用户的所有权限，并可随意改变对用户文件的访问。所以提供了最小保护来防止那些恶意软件的攻击。很多系统服务和特许程序运行时具有远远超过了他们需求的粗粒度特权，以至于这些程序中的任何缺陷都可以被利用来获取进一步的系统访问权限。

下面是一个用于 Linux 操作系统没有运行安全增强 Linux(SELinux)上权限的示例。这些示例中的权限与输出可能与您的系统略有不同。请使用 `ls -l` 命令来查看文件权限：

```
$ls -l file1
-rwxrw-r--. 1 user1 group1 0 2009-08-30 11:03 file1
```

本例中，前三个权限位（`rwX`）控制着用户 `user1`（这种情况下，就是该拥有者）对 `file1` 的访问。后面的三个权限位（`rw-`）控制着 Linux 组 `group1` 对 `file1` 的访问。最后三位权限位（`r--`）控制着其他用户对 `file1` 的访问，包括所有用户和进程。

安全增强 Linux (SELinux)向 Linux 内核增加了强制访问控制(MAC), MAC 在中标麒麟高级服务器操作系统中默认情况下是启动的。通用目标下的 MAC 架构需要实施系统中所有进程和文件管理上的安全策略,基于包含多种安全相关的信息的标记进行决策。当正确实施时,它使得一个系统能充分保护本身并通过防止干预和回避安全应用程序,对应用程序安全提供关键支持。MAC 提供了很强的应用程序分离,该措施允许不信任应用程序的安全执行。MAC 具有限定相关进程特权的能力,能限制应用程序和系统服务中的漏洞可能引起的潜在危害的范围。MAC 使得信息不会受到具有有限权限的合法用户和无意下执行了恶意应用程序的授权用户的伤害,从而得到保护。

下面是包含安全相关信息标记的示例,在运行 SELinux 的 Linux 操作系统上,这些与安全相关的信息用于进程、Linux 用户和文件。这种信息叫做 SELinux 上下文,可用 `ls -Z` 命令查看。

```
$ ls -Z file1  
-rwxrw-r--. user1 group1 unconfined_u:object_r:user_home_t:s0file1
```

本例中,SELinux 提供一个用户(`unconfined_u`)、一个角色(`object_r`)、一种类型(`user_home_t`)和一个层次(`s0`)。这个信息可用做访问控制决策。使用 DAC,访问只基于 Linux 用户和用户组 ID 进行控制。非常重要的一点是,记住 SELinux 安全策略规则是在 DAC 规则后检查。若 DAC 规则首先拒绝访问,则 SELinux 策略规则不会被使用。

Linux 与 SELinux 用户

在运行 SELinux 的 Linux 操作系统上,有 Linux 用户也有 SELinux 用户。SELinux 用户是 SELinux 策略的一部分。Linux 用户被映射为 SELinux 用户。为避免混淆,本指南使用“Linux 用户”和“SELinux 用户”来区分二者。

3.1.1 运行 SELinux 的益处

1) 所有进程和文件都标记有一种类型。

类型定义了进程和文件的域。不同进程独立运行在自己的域中,SELinux 策略规则定义进程如何与文件交互以及进程间如何交互,只有 selinux 安全策略存在时访问才被允许。

2) 细粒度访问控制。

超越了传统 UNIX 由用户判断、基于 Linux 用户和组 ID 的控制权限,SELinux

访问决策基于全部可以获得的信息，如 SELinux 用户、角色、类型和可选的层次。

3) SELinux 策略是面向管理定义的，在系统范围内实施，而不是由用户决定的。

4) 加强了防御特权提升防攻击的能力。

一个例子是：既然进程运行在域中，因此相互是分离的。因为 SELinux 策略规则定义了进程如何访问文件和其他进程，因此，若一个进程受到危害，攻击者只能访问那个进程的一般常规功能和该进程被配置的有权限访问的文件。例如，若 Apache HTTP 服务器受到攻击，攻击者不能使用那个进程来读取用户主目录下的文件，除非增加或配置了允许这样访问的特殊 SELinux 策略规则。

5) SELinux 可用于加强数据保密性和完整性，同时保护进程，拒绝不被信任的输入。

然而，SELinux 并不是：

- 1) 防病毒软件。
- 2) 密码、防火墙或其他安全系统的替代品。
- 3) 一个一体化的安全解决方案。

SELinux 的设计可以增强现有安全解决方案，而不是取代它们。即使在运行 SELinux 时，遵守好的安全做法也是非常重要的，例如保持软件的更新，使用很难猜到的密码、防火墙等等。

3.1.2 示例

下面的示例描述了 SELinux 如何增强安全性：

1) 默认的动作是拒绝。

若一个 SELinux 策略规则不允许诸如进程打开文件此类访问，则访问被拒绝。

2) SELinux 能限制 Linux 用户。

很多受限的 SELinux 用户存在于 SELinux 策略中。Linux 用户可被映射为受限的 SELinux 用户，使他们可利用适用于它们的安全规则和机制。例如，映射一个 Linux 用户为 SELinux 的 user_u 用户，将导致 Linux 用户无法运行（除非另外配置）设置用户 ID(setuid)的应用程序，如 sudo 和 su,也可防止用户执行他们主目录下（如果配置了）的文件和应用程序，这将防止用户从他们的主目录执行恶

意文件。

3) 使用进程分离。

进程运行于它们自己的域，防止进程访问其他进程使用的文件，防止进程访问其他进程。例如，运行 SELinux 时，除非另外配置，攻击者不能危害 Samba 服务器，更不能使用 Samba 服务器作为攻击媒介来读和写其他进程用到的文件，如 MySQL 使用的数据库。

4) SELinux 有助于限制由于配置错误引起的危害。

域名系统(DNS)服务器之间经常相互复制信息，称为区域传送 (Zone Transfer)。攻击者可使用区域传送以错误信息更新 DNS 服务器。当运行伯克利因特网域名(BIND)作为中标麒麟高级服务器操作系统中的 DNS 服务器时，即使管理员忘记限制哪个服务器可以执行区域传送，默认的 SELinux 策略也能防止区域文件通过区域传送被名为 BIND 的后台程序本身和其他进程更新。

5) 关于 SELinux 的背景信息和 SELinux 防止的 Exploit 漏洞的各种信息，请参阅 LinuxWorld.com 和《服务器软件的安全带：SELinux 阻止 Real-worldExploit 漏洞》。

在 Tresys Technology 网站有一个 SELinux 缓解攻击新闻章节(在右手边上)，该章节列出了被 SELinux 缓解或阻止的 Exploit 漏洞攻击的最新信息。

3.1.3 SELinux 的架构

SELinux 是一个被嵌入 Linux 内核的 Linux 安全模块。SELinux 是由可加载策略规则驱动的。当发生与安全相关的访问时，例如当一个进程试图打开一个文件，该操作在内核中将被 SELinux 阻止。若 SELinux 策略规则允许此操作，它将继续，否则，该操作被停止，并且该进程会接收到一个错误。

诸如允许或禁止访问等的 SELinux 决策将被缓冲。该缓冲称为访问向量缓冲区(AVC)。缓冲决策降低了 SELinux 策略规则需要被检查的频率，增强了性能。请记住，如果 DAC 规则首先拒绝访问，则 SELinux 策略规则将不生效。

3.1.4 其他操作系统上的 SELinux

请参阅下面，查看关于在其他 Linux 发行版本上运行 SELinux 的信息。

•Hardened Gentoo:

<http://www.gentoo.org/proj/en/hardened/selinux/selinux-handbook.xml>.

• Debian: <http://wiki.debian.org/SELinux>.

•Ubuntu: <https://wiki.ubuntu.com/SELinux> 和 <https://help.ubuntu.com/community/SELinux>.

• Fedora: <http://fedoraproject.org/wiki/SELinux> 和 [Fedora SELinux FAQ16](#).

3.2 SELinux 上下文

进程与文件都被标记上包含额外信息的 SELinux 上下文，如 SELinux 用户、角色、类型和可选的层次信息。当运行 SELinux 时，所有这些信息都用于做访问控制决策。中标麒麟高级服务器操作系统中，SELinux 提供了基于角色的访问控制(RBAC)、“类型增强”(TE)和可选的多层安全的组合(MLS)。

下面是表明 SELinux 上下文的一个示例。SELinux 上下文用于运行 SELinux 的 Linux 操作系统中的进程、Linux 用户和文件，使用 `ls -Z` 命令来查看文件和目录的安全上下文：

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

SELinux 上下文遵守 SELinux 用户:角色:类型:层次 (SELinux user:role:type:level) 的语法：

SELinux 用户 (SELinux user)

SELinux 用户身份是对特定角色和特定 MLS 范围授权的策略已知的一种身份。每个 Linux 用户通过 SELinux 策略都映射到一个 SELinux 用户。这允许 Linux 用户继承来自 SELinux 用户的约束。被映射的 SELinux 用户身份用于该会话中进程的 SELinux 上下文，从而定义它们能具有什么角色和层次。以 root 用户运行 `semanage login -l` 命令来查看 SELinux 与 Linux 用户账户之间的映射列表：

```
# /usr/sbin/semanage login -l

Login  Name SELinux User MLS/MCS Range

__default__  unconfined_u  s0-s0:c0.c1023
root unconfined_u  s0-s0:c0.c1023
system_u system_u s0-s0:c0.c1023
```

不同系统的输出可能稍有不同。Login Name 列列出了 Linux 用户，SELinux User 列列出了 Linux 用户映射到了哪个 SELinux 用户。对于进程，SELinux 用

户限制了哪些角色和层次是可访问的。最后一列 **MLS/MCS Range** 是多层安全 (MLS) 和多类别安全 (MCS) 使用的层次。稍后会给出层次的简要描述。

角色 (Role)

部分 SELinux 是基于角色的访问控制(RBAC)安全模型。角色是 RBAC 的一个属性。SELinux 用户是角色的授权，角色是域的授权。角色作为域和 SELinux 用户间的媒介。所具有的角色决定了可进入哪些域，基本上，这控制着哪些客体类型能被访问。这有助于增强对特权提升攻击能力的防御。

类型 (type)

类型是“类型增强”的一个属性。类型定义了进程的域、文件的类型。SELinux 策略规则定义了类型间如何相互访问以及类型是否是一个可访问某种类型或可访问其他域的域。只有当存在允许访问的 SELinux 策略规则时，才允许访问。

层次 (level)

层次是 MLS 和多类别安全(MCS)的属性。MLS 范围是一对层次，若两个层次不同，则记为 **lowlevel-highlevel**，若两个层次相同(**s0-s0** 与 **s0** 相同)，则记为 **lowlevel**。每个层次是一个敏感性类别对，类别是可选的。若存在类别，层次记为 **sensitivity:category-set**。若没有类别，则记为 **sensitivity**。

若类别是连续序列，则可被省略。例如，**c0.c3** 与 **c0,c1,c2,c3** 相同。

/etc/selinux/targeted/setrans.conf 文件将层次(**s0:c0**)映射到人可读的形式 (即 **CompanyConfidential**)。不要用文本编辑器编辑 **setrans.conf**，请使用 **semanage** 来进行改动。在中标麒麟高级服务器操作系统中，**targeted** 策略增强了 MCS，在 MCS 中，只有一种敏感性 **s0**。中标麒麟高级服务器操作系统中的 MCS 支持 1024 种不同的类别：从 **c0** 到 **c1023**。**s0-s0:c0.c1023** 是 **s0**，并且授权给所有类别。

MLS 增强了 Bell-LaPadula Mandatory Access Model，用于被标记的安全保护文件(LSPP)环境。要使用 MLS 约束，请安装 **selinux-policy-mls** 包，并配置 MLS 为默认的 SELinux 策略。与中标麒麟高级服务器操作系统同时发布的 MLS 策略省略了一些程序域，这些程序域不是被评估配置的一部分，因此，桌面工作站上的 MLS 是不可用的 (不支持 X Window 系统)，然而，**upstream SELinux Reference Policy** 中的一个 MLS 策略可被构建，包含所有程序域。

3.2.1 域迁移

通过对新域执行具有 `entrypoint` 类型的应用程序，可将一个域中的进程迁移到另一个域中。`Entrypoint` 权限用于 SELinux 策略，并控制哪些应用程序可进入一个域。下面的示例描述了域迁移：

若一个用户要变更其的密码，运行 `passwd` 应用程序。`/usr/bin/passwd` 可执行程序被标记为 `passwd_exec_t` 类型：

```
$ ls -Z /usr/bin/passwd
-rwsr-xr-x. root root system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd
```

应用程序 `passwd` 访问 `/etc/shadow`，`/etc/shadow` 被标记为 `shadow_t` 类型：

```
$ ls -Z /etc/shadow
-r------. root root system_u:object_r:shadow_t:s0 /etc/shadow
```

SELinux 策略声明运行于 `passwd_t` 域的进程可以读和写标记为 `shadow_t` 类型的文件。`shadow_t` 类型只适用于那些变更密码时所需的文件。这包括 `/etc/gshadow`，`/etc/shadow` 及其备份文件。

SELinux 策略规则说明 `passwd_t` 域具有 `passwd_exec_t` 类型的权限。

当用户运行 `/usr/bin/passwd` 应用程序时，用户的 Shell 进程迁移到 `passwd_t` 域。使用 SELinux，只要存在一条规则允许运行于 `passwd_t` 域的应用程序访问标记为 `shadow_t` 类型的文件，即使默认动作是拒绝，`passwd` 应用程序仍允许访问 `/etc/shadow`，并更新用户的密码。

本示例并不详尽，只是用于解释域迁移的基本例子。虽然有规则允许运行在 `passwd` 域的主体访问标记为 `shadow_t` 文件类型的客体，但在主体迁移到新域前一定要满足其他 SELinux 策略规则。本例中，“类型增强”保证了：

- 只有执行标记为 `passwd_exec_t` 类型的应用程序才能进入 `passwd_t` 域；只能从授权共享的库中执行标记了诸如 `lib_t` 类型的应用程序，不能执行其他应用程序。
- 只有如 `passwd_t` 授权的域能向标记为 `shadow_t` 类型文件写入。即使其他进程以超级用户特权运行，那些进程也不能写入到标记为 `shadow_t` 类型文件，因为它们不在 `passwd_t` 域中运行。
- 只有授权的域能迁移到域 `passwd_t`。例如，运行在 `sendmail_t` 域的 `sendmail` 进程不具备执行 `passwd` 的合法理由，因此，它不能迁移到 `passwd_t` 域。

- 运行在 `passwd_t` 域的进程只能读和写授权类型的文件，如标记为 `etc_t` 或 `shadow_t` 类型的文件。这可防止 `passwd` 应用程序被骗读或骗写任意文件。

3.2.2 进程的 SELinux 上下文

请使用 `ps -eZ` 命令查看进程的 SELinux 上下文。例如：

- 1) 打开一个终端，如点击【应用程序】=>【系统工具】=>【终端】。
- 2) 运行 `/usr/bin/passwd` 命令。不要键入新的密码。
- 3) 打开一个选项卡或另一终端，运行 `ps -eZ | grep passwd` 命令。输出与如下结果类似：

```
unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 13212 pts/1 00:00:00 passwd
```

- 4) 在第一个选项卡/终端中，按 `Ctrl+C` 来取消 `passwd` 应用程序。

本例中，当执行 `/usr/bin/passwd` 应用程序（被标记为 `passwd_exec_t` 类型）时，用户的 Shell 进程迁移到 `passwd_t` 域。请记住：类型定义了进程的域和文件的类型。

使用 `ps -eZ` 命令查看运行进程的 SELinux 上下文。下面是输出的部分示例，可能与您系统的输出不同。

```
system_u:system_r:sshd_t:s0-s0:c0.c1023 1882 ? 00:00:00 sshd
system_u:system_r:gpm_t:s0 1964 ? 00:00:00 gpm
system_u:system_r:crond_t:s0-s0:c0.c1023 1973 ? 00:00:00 crond
system_u:system_r:kerneld_t:s0 1983 ? 00:00:05 kerneld
system_u:system_r:crond_t:s0-s0:c0.c1023 1991 ? 00:00:00 atd
```

`system_r` 角色是用于系统进程的，如 `Daemon`。“类型增强”划分每个域。

3.2.3 用户的 SELinux 上下文

使用 `id -Z` 命令来查看您 Linux 用户的 SELinux 上下文：

```
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

在中标麒麟高级服务器操作系统中，Linux 用户默认运行为非限制用户。这种 SELinux 上下文表明 Linux 用户被映射到 SELinux 的 `unconfined_u` 用户，以 `unconfined_r` 角色运行，并运行在 `unconfined_t` 域。`s0-s0` 是一个 MLS 范围，此情况下，MLS 范围就是 `s0`。用户访问的类别被 `c0.c1023` 拒绝，`c0.c1023` 是所有类别（从 `c0` 到 `c1023`）。

3.3 Targeted 策略

在中标麒麟高级服务器操作系统中，Targeted 策略是默认的 SELinux 策略。当使用 targeted 策略时，targeted 进程运行在限制域内，非 targeted 进程运行在非限制域内。例如，默认情况下，登陆用户运行于 unconfined_t 域，由初始化启动的系统进程运行于 initrc_t 域——这两个域都是限制域。

非限制域（和限制域）都要受到可执行和可写内存检查。默认情况下，运行于非限制域的主体不能分配可写内存并执行写操作。这减少了易受缓冲区溢出的攻击漏洞。通过设置布尔变量，可以禁止这些内存检查，从而可在运行时修改 SELinux 策略。布尔变量的配置稍后讨论。

3.3.1 限制进程

几乎监听网络的所有服务在中标麒麟高级服务器操作系统中都是受限制的，如 sshd 或 httpd。同样，作为 Linux root 用户并为用户执行任务的多数进程都是受限制的，如 passwd 应用程序。限制进程运行于它自己的域，如 httpd 进程运行于 httpd_t 域。若一个限制进程是由于受到攻击而处于危险中，根据 SELinux 策略配置，攻击者对资源的访问和它们可能造成的损害都是有限的。

下面的示例描述了 SELinux 如何防止 Apache HTTP 服务器 (httpd) 免于受到没有被正确标记的读文件操作的伤害，如 Samba 使用的文件。这是一个示例，请不要在实际中使用。它假设已安装了 httpd 和 wget 包，使用了 SELinux targeted 策略，并且 SELinux 运行于强制模式：

- 1) 运行 sestatus 命令来确定 SELinux 被启用，并运行于强制模式，使用 targeted 策略。

```
$ /usr/sbin/sestatus
SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: enforcing
Mode from config file: enforcing
Policy version: 24
Policy from config file: targeted
```

SELinux 状态: 当 SELinux 被启用时，返回 enabled。当前模式: 当 SELinux 运行于强制模式时，返回 enforcing。config 文件中的策略: 当 SELinux 使用 targeted

策略时，返回 targeted 。

2) 作为 root 用户，运行 touch /var/www/html/testfile 命令创建一个文件。

3) 运行 ls -Z /var/www/html/testfile 命令来浏览 SELinux 上下文：

```
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/testfile
```

默认情况下，中标麒麟高级服务器操作系统中，Linux 用户以非限制状态运行，这就是为什么 testfile 文件标记为 SELinux unconfined_u 用户。RBAC 用于进程，而不是文件。

角色对于文件来说没有意义，object_r 是用于文件（在持久存储和网络文件系统中）的普通角色。在 /proc/ 目录下，与进程有关的文件可以使用 system_r 角色。httpd_sys_content_t 类型允许 httpd 进程访问此文件。

作为 root 用户，运行 service httpd start 命令来启动 httpd 进程，若启动成功，输出如下所示：

```
# /sbin/service httpd start
Starting httpd: [ OK ]
```

转换到您的 Linux 用户具有写权限的一个目录，运行 wget http://localhost/testfile 命令。若您没有改变默认配置，命令会执行成功：

```
--2009-11-06 17:43:01-- http://localhost/testfile
Resolving localhost... 127.0.0.1
Connecting to localhost[127.0.0.1]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `testfile'

[ <=> ] 0 --.-K/s in 0s

2009-11-06 17:43:01 (0.00 B/s) - `testfile' saved [0/0]
```

chcon 命令重新标记文件；然而，当文件系统被重新标记时，这样的标记变更不会生效。为了使文件系统被重新标记时，文件的标记变更能持久，请使用 semanage 命令，稍后讨论该命令。作为 root 用户，请运行下面的命令来改变到 Samba 使用的一个类型：

```
chcon -t samba_share_t /var/www/html/testfile
```

运行 `ls -Z /var/www/html/testfile` 命令来查看变更:

```
-rw-r--r--.  root  root  unconfined_u:object_r:samba_share_t:s0 /var/www/html/testfile
```



注意: 当前的 DAC 权限允许 httpd 进程访问 testfile。请变更到您的 Linux 用户具有写权限的一个目录, 并运行 `wget http://localhost/testfile` 命令。若您没有改变默认配置, 命令会执行失败:

```
--2009-11-06 14:11:23-- http://localhost/testfile
Resolving localhost... 127.0.0.1
Connecting to localhost[127.0.0.1]:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2009-11-06 14:11:23 ERROR 403: Forbidden.
```

作为 root 用户, 请运行 `rm -i /var/www/html/testfile` 命令来删除 testfile。

如果您不需要运行 httpd 进程, 作为 root 用户, 请运行 `service httpd stop` 命令来停止 httpd 进程:

```
# /sbin/service httpd stop
Stopping httpd: [ OK ]
```

本示例描述了 SELinux 增加的额外安全性。虽然 DAC 规则允许 httpd 进程访问步骤 7 中的 testfile, 但是因为文件已被标记为 httpd 进程无权访问, 所以 SELinux 拒绝此访问。

与下面类似的一个错误被记录到 `/var/log/audit/audit.log`:

```
type=AVC msg=audit(1220706212.937:70): avc: denied { getattr } for
pid=1904 comm="httpd" path="/var/www/html/testfile" dev=sda5 ino=247576
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:samba_share_t:s0
tclass=file

type=SYSCALL msg=audit(1220706212.937:70): arch=400000003 syscall=196
success=no exit=-13 a0=b9e21da0 a1=bf9581dc a2=555ff4 a3=2008171 items=0
ppid=1902 pid=1904 auid=500 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48
fsgid=48 tty=(none) ses=1 comm="httpd" exe="/usr/sbin/httpd"
subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

同样, 与下面类似的一个错误被记录到 `/var/log/httpd/error_log`:

```
[Wed May 06 23:00:54 2009] [error] [client 127.0.0.1] (13)Permission denied:
access to /
```

testfile denied

3.3.2 非限制进程

非限制进程运行于非限制域，例如，初始化程序运行于 `initrc_t` 非限制域，非限制内核进程运行于 `kernel_t` 域，非限制 Linux 用户运行于 `unconfined_t` 域。对于非限制进程，SELinux 策略规则将被应用，但是策略规则允许运行于非限制域的进程几乎所有的权限。运行于非限制域的进程转而专用 DAC 规则。若一个非限制进程受到攻击，SELinux 不阻止攻击者访问系统资源与数据，但是当然，DAC 规则仍被使用。SELinux 是在 DAC 规则基础上的安全增强，它并没有取代 DAC 规则。

下面的示例描述了运行于非限制域的 Apache HTTP 服务器 (`httpd`) 如何访问 Samba 使用的数据。注意：在中标麒麟高级服务器操作系统中，`httpd` 进程默认运行于限制域 `httpd_t`。这只是一个示例，请不要在实际中使用。它假设已安装了 `httpd`、`wget`、`dbus` 和 `audit` 包，使用了 `targeted` 策略，SELinux 运行于强制模式。

- 1) 运行 `sestatus` 命令来确定 SELinux 被启用，并运行于强制模式，使用 `targeted` 策略。

```
$ /usr/sbin/sestatus
SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: enforcing
Mode from config file: enforcing
Policy version: 24
Policy from config file: targeted
```

SELinux 状态：当 SELinux 启用时，返回 `enabled`。当前模式：当 SELinux 运行于强制模式时，返回 `enforcing`。config 文件中的策略：SELinux 使用 `targeted` 策略时，返回 `targeted`。

- 2) 作为 Linux root 用户，运行 `touch /var/www/html/test2file` 命令来创建一个文件。
- 3) 运行 `ls -Z /var/www/html/test2file` 命令来查看 SELinux 上下文。

```
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
test2file
```

默认情况下，中标麒麟高级服务器操作系统中，Linux 用户以非限制状态运行，这就是为什么 test2file 文件标记为 SELinux unconfined_u 用户。RBAC 用于进程，而不是文件。角色对于文件来说没有意义，object_r 是用于文件（在持久存储和网络文件系统中）的普通角色。在 /proc/ 目录下，与进程有关的文件可以使用 system_r 角色。httpd_sys_content_t 类型允许 httpd 进程访问此文件。

4) chcon 命令重新标记文件；

然而，当文件系统被重新标记时，这样的标记变更不会生效。为了使文件系统被重新标记时，文件的标记变更能持久，请使用 semanage 命令，稍后讨论该命令。作为 Linux root 用户，请运行下面的命令来改变到 Samba 使用的一个类型。

```
chcon -t samba_share_t /var/www/html/test2file
```

5) 运行 ls -Z /var/www/html/test2file 命令来查看变更。

```
-rw-r--r--. root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/test2file
```

6) 运行 service httpd status 命令来确定 httpd 进程没有运行。

```
$ /sbin/service httpd status httpd
stopped
```

如果输出跟上面的结果不同，请作为 root 用户运行 service httpd stop 命令来停止 httpd 进程。

```
# /sbin/service httpd stop
Stopping httpd: [ OK ]
```

要令 httpd 进程运行于非限制域，请作为 root 用户运行下面的命令来从 /usr/sbin/httpd 类型变更到不需要转换到限制域的另一类型。

```
chcon -t unconfined_exec_t /usr/sbin/httpd
```

7) 运行命令来确定 /usr/sbin/httpd 是标记为 unconfined_exec_t 类型。

```
-rwxr-xr-x. root root system_u:object_r:unconfined_exec_t /usr/sbin/httpd
```

8) 作为 root 用户，运行 service httpd start 命令来启动 httpd 进程。若 httpd 成功启动，则输出如下内容：

```
# /sbin/service httpd start
Starting httpd: [ OK ]
```

9) 运行 ps -eZ | grep httpd 命令来查看运行于 unconfined_t 域的 httpd 进程：

```
$ ps -eZ | grep httpd
```

```
unconfined_u:system_r:unconfined_t 7721 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7723 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7724 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7725 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7726 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7727 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7728 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7729 ? 00:00:00 httpd
unconfined_u:system_r:unconfined_t 7730 ? 00:00:00 httpd
```

10) 转换到您的 Linux 用户具有写权限的一个目录，运行 `wget http://localhost/test2file` 命令。若您没有改变默认配置，命令会执行成功。

```
--2009-05-07 01:41:10-- http://localhost/test2file
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `test2file.1'

[ <=> ]--.-K/s in 0s

2009-05-07 01:41:10 (0.00 B/s) - `test2file.1' saved [0/0]
```

虽然 httpd 进程没有权限访问标记为 `samba_share_t` 类型的文件，但 httpd 进程运行于 `unconfined_t` 非限制域，转而使用 DAC 规则，`wget` 命令将成功。若进程运行在限制域 `httpd_t`，`wget` 将失败。

`restorecon` 命令还原文件的默认 SELinux 上下文。作为 root 用户，请运行 `restorecon -v /usr/sbin/httpd` 命令来还原 `/usr/sbin/httpd` 默认的 SELinux 上下文。

```
# /sbin/restorecon -v /usr/sbin/httpd
restorecon reset /usr/sbin/httpd context system_u:object_r:unconfined_notrans_exec_t:s0-
>system_u:object_r:httpd_exec_t:s0
```

11) 运行 `ls -Z /usr/sbin/httpd` 命令来确定 `/usr/sbin/httpd` 被标记为 `httpd_exec_t` 类型：

```
$ ls -Z /usr/sbin/httpd
-rwxr-xr-x. root root system_u:object_r:httpd_exec_t /usr/sbin/httpd
```

- 12) 作为 root 用户, 请运行 `/sbin/service httpd restart` 命令来重启 httpd 进程。在重启后, 运行 `ps -eZ | grep httpd` 命令来确认 httpd 运行于 httpd_t 限制域。

```
# /sbin/service httpd restart
Stopping httpd:  [ OK ]
Starting httpd: [ OK ]
```

```
# ps -eZ | grep httpd
unconfined_u:system_r:httpd_t 8880 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8882 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8883 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8884 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8885 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8886 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8887 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8888 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t 8889 ? 00:00:00 httpd
```

- 13) 作为 Linux root 用户, 请运行 `rm -i /var/www/html/test2file` 命令来删除 test2file 文件。

- 14) 若您不需要 httpd 运行, 作为 root 用户, 请运行 `service httpd stop` 命令来停止 httpd。

```
# /sbin/service httpd stop
Stopping httpd:  [ OK ]
```

3.3.3 限制与非限制用户

通过 SELinux 策略, 每个 Linux 用户都映射到一个 SELinux 用户。这使得用户可以继承对 SELinux 关于用户的限制条件。Linux 用户映射可以通过作为 root 用户运行 `semanage login-l` 命令来查看:

```
# /usr/sbin/semanage login -l

Login Name  SELinux User    MLS/MCS Range
__default__ unconfined_u    s0-s0:c0.c1023
root unconfined_u s0-s0:c0.c1023
```

```
system_u system_u s0-s0:c0.c1023
```

在中标麒麟高级服务器操作系统中，Linux 用户被默认映射到 SELinux default login（Linux 用户被映射到 SELinux unconfined_u 用户）。下面定义了这种默认映射：

default	unconfined_u	s0-s0:c0.c1023
---------	--------------	----------------

下面的示例描述了增加一个新的 Linux 用户，此新用户被映射到 SELinux unconfined_u 用户。它假设 root 用户运行于非限制域，就如同它在中标麒麟高级服务器操作系统 V6.4 中的默认设置一样。

- 1) 作为 root 用户，运行/usr/sbin/useradd newuser 命令来创建一个新的 Linux 用户，命名为 newuser。
- 2) 作为 root 用户，运行 passwd newuser 命令来为 Linux 用户 newuser 设置一个密码。

```
# passwd newuser
Changing password for user newuser.
New UNIX password: Enter a password
Retype new UNIX password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

- 3) 注销当前会话，并以用户 newuser 的身份登陆。当您登录时，pam_selinux 将一个 Linux 用户映射为一个 SELinux 用户(本例中为 unconfined_u)，并建立 SELinux 上下文。Linux 用户的 Shell 以这个上下文启动。运行 id -Z 命令来查看 Linux 用户的上下文。

```
[newuser@localhost ~]$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 4) 注销 Linux 用户 newuser 的会话，并登陆您的账户。若您不想要 newuser 用户，作为 root 用户运行/usr/sbin/userdel -r newuser 命令删除它，同时删除 Linux 用户 newuser 的主目录。

限制与非限制用户都受到可执行与可写内存的检查，并被 MCS（和 MLS，若使用 MLS 策略）约束。若非限制 Linux 用户执行了由 SELinux 策略定义的可从 unconfined_t 域转换到它本身限制域的一个应用程序，则非限制 Linux 用户都受到限制域的约束。这样做在安全方面的益处在于，即使 Linux 用户是非限制的，

应用程序仍是限制的，因此，应用程序错误的传播可通过策略得到控制。注意：这并不保护系统免受用户的攻击。事实上，用户和系统都会受到保护，使它们免于受到应用程序错误引起的可能危害，下面受限制的 SELinux 用户在中标麒麟高级服务器操作系统中都可以找到。

表 3-1 SELinux 用户权限

用户	域	X Window 系统	Su 和 sudo	在主目录和/tmp/ 执行	联网
guest_u	guest_t	否	否	可选	否
xguest_u	xguest_t	是	否	可选	仅 Firefox
user_u	user_t	是	否	可选	是
staff_u	staff_t	是	仅 sudo	可选	是

- 如果 SELinux 策略允许某些应用程序(例如 passwd)，则 guest_t, xguest_t 和 user_t 域的 Linux 用户只可以运行固定用户 ID(setuid)的应用程序。它们不能够运行 su 和/usr/bin/ sudo setuid 应用程序，因此，不能使用这些应用程序而变成 root 用户。

- guest_t 域的 Linux 用户没有网络权限，只能通过一个终端（包括 ssh；他们能通过 ssh 登录，但不能使用 ssh 连接到其他系统）登录。
- xguest_t 域的 Linux 用户具有仅有的网络访问，使用 Firefox 连接到网页。
- xguest_t, user_t 和 staff_t 域的 Linux 用户可以通过 X Window 系统和一个终端登录。
- 默认情况下，staff_t 域的 Linux 用户不具备执行需要使用/usr/bin/sudo 的应用程序的权限。这些权限必须由管理员进行配置。

默认情况下，guest_t 和 xguest_t Linux 域的用户不能执行他们主目录或/tmp/目录下的应用程序，防止他们执行他们有写权限的目录中的应用程序（此应用程序继承用户的权限）。这有助于防止错误的或恶意应用程序修改用户自己的文件。

默认情况下，user_t 和 staff_t 域的 Linux 用户可以执行他们主目录或/tmp/目录下的应用程序。

3.4 使用 SELinux

下面的章节给出了中标麒麟高级服务器操作系统中主要 SELinux 包的简短

概述；安装和更新包；使用哪个日志文件；主 SELinux 配置文件；启用和禁用 SELinux；SELinux 模式；配置布尔变量；临时和永久变更文件与目录的标记；用 mount 命令重载文件系统标记；安装 NFS 文件系统；复制与归档文件目录时，如何保存 SELinux 上下文。

3.4.1 SELinux 包

中标麒麟高级服务器操作系统中，SELinux 包默认为全部安装，除非在安装时手工去除其中某些包。若执行文本模式下的最小安装，默认情况下，不安装 policycoreutils-python 包。同样，默认情况下，使用 SELinux targeted 策略，SELinux 运行在强制模式下。下面是主要 SELinux 包的简述：

policycoreutils-python 包：提供如 semanage, audit2allow, audit2why 和 chcat 等的实用工具，用于操作和管理 SELinux.

Policycoreutils 包：提供如 restorecon、secon、setfiles、semodule、load_policy 和 setsebool 等的实用工具，用于操作和管理 SELinux.

policycoreutils-gui 包：提供用于管理 SELinux 的图形工具 system-config-selinux。

selinux-policy 包：提供 SELinux Reference 策略。SELinux Reference 策略是完备的，是其他策略的基础，如 targeted 策略。更多信息请参阅 Tresys 技术 SELinux Reference Policy 策略页。selinux-policy-devel 包提供了开发工具，如 /usr/share/selinux/devel/policygentool 和 /usr/share/selinux/devel/policyhelp，还有示例策略文件。

selinux-policy-policy 包：提供 SELinux 策略。为 targeted 策略安装 selinux-policy-targeted。为 MLS 安装 selinux-policy-mls。

setroubleshoot-server 包：将 SELinux 拒绝访问时产生的拒绝消息翻译为可用 sealert（包含在本包中）查看的详细描述。

setools-console 包：本包提供 Tresys Technology SETools distribution，包含分析和查询策略、日志监听和报告，以及文件上下文处理等一系列工具和库。Setools 包是 SETools 的元包。SETools 包提供了 apol、seaudit 和 sediffx 工具。setools-console 包提供了 seaudit-report、sechecker、sediff、seinfo、sesearch、findcon、replcon 和 indexcon 命令行工具。

libselinux-utils 包: 提供了 avcstat、getenforce、getsebool、matchpathcon、selinuxconlist、selinuxdefcon、selinuxenabled、setenforce、togglesebool 工具。

Mcstrans 包: 将诸如 s0-s0:c0.c1023 的层次翻译为更容易阅读的形式, 如 SystemLow-SystemHigh。默认情况下不安装此包。

要安装某个包, 请作为 root 用户运行 `yum install package-name` 命令。例如, 要安装 mcstrans 包, 请运行 `yum install mcstrans` 命令。要更新中标麒麟高级服务器操作系统中所有已安装的包, 请运行 `yum update` 命令。

3.4.2 使用哪个日志文件

中标麒麟高级服务器操作系统中, 如果没有从默认的包选择中删除, 则 dbus、setroubleshoot-server 和 audit 包将被安装。

SELinux 的拒绝消息, 如下所示, 默认写入 `/var/log/audit/audit.log`:

```
type=AVC msg=audit(1223024155.684:49): avc: denied { getattr } for
pid=2000 comm="httpd" path="/var/www/html/file1" dev=dm-0 ino=399185
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=system_u:object_r:samba_share_t:s0
tclass=file
```

```
May 7 18:55:56 localhost setroubleshoot: SELinux is preventing httpd (httpd_t)
"getattr"
to /var/www/html/file1 (samba_share_t). For complete SELinux messages. run
sealert -l de7e30d6-5488-466d-a606-92c9f40d316d
```

中标麒麟高级服务器操作系统中, setroubleshootd 不再作为一个服务不断运行, 然而, 它仍被用来分析 AVC 消息。当需要时, 两个新程序 edispatch 和 seapplet 可以启动 setroubleshoot。sedispatch 作为监听子系统的一部分运行, 当 AVC 拒绝发生时, 通过 dbus 发送消息, 若 setroubleshootd 正在运行, 则直接跳转到 setroubleshootd, 否则, 启动 setroubleshootd。seapplet 是在系统工具条运行的一个工具, 等待 setroubleshootd 中的 dbus 消息, 并利用提醒泡, 允许用户查看拒绝信息。

自动启动 后台程序

要配置 auditd 和 rsyslogd 后台程序在启动时自动运行, 请作为 root 用户运行下面的命令:

```
/sbin/chkconfig --levels 2345 auditd on
```

```
/sbin/chkconfig --levels 2345 rsyslog on
```

请使用 `service service-name status` 命令检查这些服务是否正在运行，例如：

```
$ /sbin/service auditd status  
auditd (pid 1318) is running...
```

若上面的服务没有运行（`service-name is stopped`），请作为 Linux root 用户运行 `service service-name start` 命令来启动它们。例如：

```
# /sbin/service auditd start  
Starting auditd: [ OK ]
```

3.4.3 主配置文件

`/etc/selinux/config` 文件是主 SELinux 配置文件。它控制所使用的 SELinux 模式和 SELinux 策略：

```
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
# enforcing - SELinux security policy is enforced.  
# permissive - SELinux prints warnings instead of enforcing.  
# disabled - No SELinux policy is loaded.  
SELINUX=enforcing  
# SELINUXTYPE= can take one of these two values:  
# targeted - Targeted processes are protected,  
# mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

`SELINUX=enforcing`

`SELINUX` 选项 设置 SELinux 运行的模式。SELinux 有三种模式：强制、允许和禁用。当使用强制模式时，SELinux 策略是强制的，SELinux 基于 SELinux 策略规则来拒绝访问。拒绝消息被记在日志中。当使用允许模式时，SELinux 策略不是强制的。SELinux 不拒绝访问，但是拒绝信息被记录在日志中，用于那些如果 SELinux 运行在强制模式下时将被拒绝的操作。当使用禁用模式时，SELinux 是禁用的（SELinux 模块没有在 Linux 内核中注册），仅使用 DAC 规则。

`SELINUXTYPE=targeted`

`SELINUXTYPE` 选项设置 SELinux 使用的策略。Targeted 策略是默认策略。若您要使用 MLS 策略，请改变此选项。要使用 MLS 策略，请在安装

selinux-policy-mls 包，/etc/selinux/config 中配置 SELINUXTYPE=mls，并重启您的系统。



重要：当系统运行于 SELinux 的允许或禁用模式时，用户被允许不正确的标记文件。同样，禁用 SELinux 时创建的文件不会被标记。当变更到强制模式时，这将引起问题。为了防止不正确的标记和无标记文件引起问题，当从禁用模式变更到允许或强制模式时，文件系统将自动重新标记它们。

3.4.4 启用和禁用 SELinux

请使用/usr/sbin/getenforce 或 /usr/sbin/sestatus 命令来检查 SELinux 的状态。getenforce 命令返回 Enforcing、Permissive 或 Disabled。当 SELinux 启用时 (SELinux 策略规则被强制)，getenforce 命令返回 Enforcing:

```
$ /usr/sbin/getenforce
Enforcing
```

当启用 SELinux 时，getenforce 命令返回 Permissive，但 SELinux 策略规则并没有被强制使用，只使用了 DAC 规则。若 SELinux 被禁用时，getenforce 命令返回 Disabled。

sestatus 命令返回所使用的 SELinux 状态和 SELinux 策略。

```
$ /usr/sbin/sestatus
SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: enforcing
Mode from config file: enforcing
Policy version: 24
Policy from config file: targeted
```

SELinux 状态：启用 SELinux 时返回 enabled。当前模式：SELinux 运行于强制模式时，返回 enforcing。config 文件中的策略：SELinux 使用 targeted 策略时，返回 targeted。

3.4.4.1 启用 SELinux

在 SELinux 被禁用的系统上，/etc/selinux/ config 中包含 SELINUX=disabled 选项的配置。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
```

```
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

同样，`getenforce` 命令返回 Disabled。

```
$ /usr/sbin/getenforce
Disabled
```

要启用 SELinux：

1) 运行如下命令确认已安装 SELinux 包

```
rpm -qa | grep selinux; rpm -q policycoreutils
rpm -qa | grep setroubleshoot
```

本指南假设已安装下面的包：`elinux-policy-targeted`、`selinux-policy`、`libselinux`、`libselinux-python`、`libselinux-utils`、`policycoreutils`、`policycoreutils-python`、`setroubleshoot`、`setroubleshoot-server` 和 `setroubleshoot-plugins`。若没有作为 Linux root 用户安装这些包，请通过 `yum install package-name` 命令安装。下面的包是可选的：`policycoreutils-gui`、`setroubleshoot`、`selinux-policy-devel` 以及 `mcstrans`。

2) 在 SELinux 被启用前，文件系统上的每个文件都必须标记有一个 SELinux 上下文。在此之前，限制域可能会拒绝访问，阻止您的系统正确启动。为防止这种情况，请在 `/etc/selinux/config` 中配置 `SELINUX=permissive`：

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
```

```
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- 3) 作为 root 用户，请运行 reboot 命令重启您的系统。在下次启动前，文件系统将被标记。标记进程用 SELinux 上下文标记所有文件。

```
*** Warning -- SELinux targeted policy relabel is required.
*** Relabeling could take a very long time, depending on file
*** system size and speed of hard drives.
****
```

最后一行中的每个字符*代表 1000 个已被标记的文件。上例中，4 个*代表 4000 个文件已被标记。标记所有文件所需的时间取决于系统中文件的数量，以及硬盘驱动器的速度。在现代系统中，这个进程只有 10 分钟。

- 4) 在许可模式下，SELinux 策略不是强制的，但运行于强制模式时被拒绝的动作信息仍被记入日志。在变更到强制模式前，作为 root 用户，运行 `grep "SELinux is preventing" /var/log/messages` 命令确认 SELinux 在上次启动时没有拒绝任何动作。若 SELinux 在上次启动时没有拒绝任何动作，此命令不返回任何输出。
- 5) 若 /var/log/messages 中无拒绝消息，请在 /etc/selinux/config 中配置 SELINUX=enforcing:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- 6) 重启您的系统。重启后，请确认 getenforce 命令返回的是 Enforcing:

```
$ /usr/sbin/getenforce
Enforcing
```

7) 作为 root 用户，运行 `/usr/sbin/semanage login -l` 命令查看 SELinux 与 Linux 用户之间的映射。输出应如下所示：

```
Login Name  SELinux User MLS/MCS Range
default    unconfined_u s0-s0:c0.c1023
root       unconfined_u s0-s0:c0.c1023
system_u   system_u s0-s0:c0.c1023
```

如果输出不是这种情况，请作为 root 用户运行下面的命令来修复用户映射。如果发生 SELinux-user username is already defined 警告，您可以忽略它，这种做法很安全，其中 username 是 unconfined_u、guest_u 或 xguest_u：

```
/usr/sbin/semanage user -a -S targeted -P user -R "unconfined_r system_r" -r
s0-s0:c0.c1023 unconfined_u
/usr/sbin/semanage login -m -S targeted -s "unconfined_u" -r s0- s0:c0.c1023 ____
default____
/usr/sbin/semanage login -m -S targeted -s "unconfined_u" -r s0- s0:c0.c1023 root
/usr/sbin/semanage user -a -S targeted -P user -R guest_r guest_u
/usr/sbin/semanage user -a -S targeted -P user -R xguest_r xguest_u
```



重要：当系统运行于允许或禁用模式的 SELinux 时，用户标记文件。同样，当 SELinux 禁用时创建的文件不被标记。当变更到强制模式时将引起问题。为了防止不正确的标记和无标记文件引起问题，当从禁用模式变更到允许或强制模式时，将自动重新标记文件系统。

3.4.4.2 禁用 SELinux

要禁用 SELinux，请在 `/etc/selinux/config` 中配置 `SELINUX=disabled`：

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

重启您的系统。重启后，请确认 `getenforce` 命令返回的是 Disabled：

```
$ /usr/sbin/getenforce
```

```
Disabled
```

3.4.5 SELinux 模式

SELinux 有三种模式：

- 1) 强制 (Enforcing)：SELinux 策略是强制的。SELinux 基于 SELinux 策略规则拒绝访问。
- 2) 允许 (Permissive)：SELinux 策略不是强制的。SELinux 不拒绝访问，但一些动作的拒绝信息将被记录在日志中，这些动作运行于强制模式下时会被拒绝。
- 3) 禁用 (Disabled)：SELinux 是关闭的。只使用 DAC 规则。

使用 `/usr/sbin/setenforce` 命令实现强制模式与允许模式间的切换。`/usr/sbin/setenforce` 所做的变更不能保持到重启之后。要变更为强制模式，作为 root 用户，运行 `/usr/sbin/setenforce 1` 命令。要变更到许可模式，运行 `/usr/sbin/setenforce 0` 命令。使用 `/usr/sbin/getenforce` 命令查看当前 SELinux 模式。

更多关于许可模式变更的信息，请参阅“3.4.4 启用与禁用 SELinux”。

3.4.6 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 策略即可访问 NFS 文件系统。

3.4.6.1 布尔变量列表

对于布尔变量列表中每一个变量的解释以及是 on 或是 off, 请作为 root 用户运行命令 `semanage boolean -l`。下例列出部分布尔变量：

```
# /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
```

以上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，但没有给出每一个变量的详细描述。下例列出部分布尔变量：

```
$ /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
```

3.4.6.2 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 HTTP 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。

3.4.6.3 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` 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
```



注意：这些布尔变量存在于中标麒麟高级服务器操作系统发布时的 SELinux 策略中。

它们并不存在于中标麒麟高级服务器操作系统其他版本或其他操作系统的策略中。

3.4.7 SELinux 上下文——标记文件

在运行 SELinux 的系统上，所有进程和文件都用与安全相关的信息标记。这种信息叫做 SELinux 上下文，对于文件，可以用 `ls -Z` 命令查看它的上下文。

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

本例中，SELinux 提供了一个用户(`unconfined_u`)，一个角色(`object_r`)，一种类型(`user_home_t`)和一个层次(`s0`)。这种信息用于做访问控制决策。在 DAC 系统中，访问是由 Linux 用户和组 ID 进行控制的。检查完 DAC 规则后，才检查 SELinux 策略规则。若 DAC 规则在一开始就拒绝了访问，那么就不必使用 SELinux 策略规则了。

Linux 有很多管理文件的 SELinux 上下文的命令，如 `chcon`、`semanage fcontext` 和 `restorecon` 等。

3.4.7.1 临时变更：chcon

`chcon` 命令变更文件的 SELinux 上下文。然而，文件系统重新标记或执行 `/sbin/restorecon` 命令时，用 `chcon` 命令所作的变更不能保存下来。SELinux 策略控制用户是否可以修改任何给定文件的 SELinux 上下文。当使用 `chcon` 命令时，用户提供要变更的所有或部分的 SELinux 上下文。不正确的文件类型是 SELinux 拒绝访问的常见原因。

快速参考

- 运行 `chcon -t type file-name` 命令变更文件类型，其中 `type` 是诸如

httpd_sys_content_t 的类型，file-name 是文件或目录名。

- 运行 `chcon -R -t type directory-name` 命令变更目录及其内容的类型，其中 `type` 是诸如 `httpd_sys_content_t` 的类型，`directory-name` 是目录名。

变更文件或目录的类型

下例描述了如何变更 SELinux 上下文的类型和其他属性：

- 1) 运行不带参数的 `cd` 命令变更到您的主目录。
- 2) 运行命令 `touch file1` 创建一个新文件。

使用 `file1` 命令查看 `file1` 文件的 SELinux 上下文。

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

本例中，`file1` 的 SELinux 上下文包括 SELinux 的 `unconfined_u` 用户、`object_r` 角色、`user_home_t` 类型和 `s0` 层次。关于 SELinux 上下文每个部分的详细描述，请参阅“3.2 SELinux 上下文”。

- 3) 运行 `chcon -t samba_share_t file1` 命令将类型变更为 `samba_share_t`。
`-t` 选项只改变类型。使用 `ls -Z file1` 命令查看变更。

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

- 4) 使用命令恢复文件 `file1` 的 SELinux 上下文。

使用 `-v` 选项查看发生变化的内容：

```
$ /sbin/restorecon -v file1
restorecon reset file1 context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:user_home_t:s0
```

本例中，前面的 `samba_share_t` 类型被恢复到正确的 `user_home_t` 类型。当使用 `targeted` 策略时（中标麒麟高级服务器操作系统中默认的 SELinux 策略），`/sbin/restorecon` 命令读取 `/etc/selinux/targeted/contexts/files/` 目录的文件，检查文件应具有哪种 SELinux 上下文。

本节中的例子同样适用于目录，例如，`file1` 可以是一个目录。

变更目录及其内容类型

下例描述了如何创建新目录并改变目录的文件类型（及其内容）为 Apache HTTP 服务器所使用的类型。若您需要 Apache HTTP 服务器使用不同的文档根

目录（而不是/var/www/html/），请使用本例中的配置：

- 1) 作为 root 用户，运行 `mkdir /web` 命令创建新目录，然后运行 `touch /web/file{1,2,3}` 命令创建 3 个空文件(file1、 file2 和 file3)。`/web/`目录和其中的文件都被标记为 `default_t` 类型。

```
# ls -dZ /web
drwxr-xr-x root root unconfined_u:object_r:default_t:s0 /web
# ls -lZ /web
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 file3
```

- 2) 作为 root 用户，运行 `chcon -R -t httpd_sys_content_t /web/` 命令将 `/web/` 目录（及其内容）的类型变更为 `httpd_sys_content_t`：

```
# chcon -R -t httpd_sys_content_t /web/
# ls -dZ /web/
drwxr-xr-x root root unconfined_u:object_r:httpd_sys_content_t:s0 /web/
# ls -lZ /web/
-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
```

- 3) 作为 root 用户，运行 `/sbin/restorecon -R -v /web/` 命令恢复默认的 SELinux 上下文：

```
# /sbin/restorecon -R -v /web/
restorecon reset /web context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file2 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file3 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file1 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
```

更多关于 `chcon` 的信息，请参考 `chcon(1)` 手册页。



注意：类型强制是 SELinux targeted 策略使用的主要权限控制。多数情况下，SELinux

用户和角色可被忽略。

3.4.7.2 永久变更：semanage fcontext

`/usr/sbin/semanage fcontext` 命令变更文件的 SELinux 上下文。当使用 `targeted` 策略时，若对 `file_contexts` 中存在的文件做变更，使用该命令所作的变更被添加到 `/etc/selinux/targeted/contexts/files/file_contexts` 文件中，否则对新文件和目录所作的变更将被添加到 `file_contexts.local` 中，如创建一个 `/web/` 目录。文件系统被重新标记时用到的 `setfiles` 命令以及恢复默认 SELinux 上下文的 `/sbin/restorecon` 命令都读取这些文件。这意味着由 `/usr/sbin/semanage fcontext` 所作的变更是永久的，即使文件系统被重新标记。SELinux 策略控制用户是否能修改任何给定文件的 SELinux 上下文。

快速参考

使 SELinux 上下文改变不受文件系统重新标记的影响：

- 1) 运行 `/usr/sbin/semanage fcontext -a options file-name|directory-name` 命令，请记住要使用文件或目录的完整路径。
- 2) 运行 `/sbin/restorecon -v file-name|directory-name` 命令来应用上下文变更。

变更文件类型

下例描述了如何变更文件的类型以及 SELinux 上下文的其他属性。

- 1) 作为 `root` 用户，运行 `touch /etc/file1` 命令创建一个新文件。默认情况下，`/etc/` 目录中新创建的文件被标记为类型 `etc_t`：

```
# ls -Z /etc/file1
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0 /etc/file1
```

- 2) 作为 `root` 用户，运行 `/usr/sbin/semanage fcontext -a -t samba_share_t /etc/file1` 命令将文件 `file1` 的类型改变为 `samba_share_t`。 `-a` 选项增加了一个记录， `-t` 选项定义了一个类型(`samba_share_t`)。注意：运行此命令并不能直接改变类型——`file1` 文件仍被标记为 `etc_t` 类型：

```
# /usr/sbin/semanage fcontext -a -t samba_share_t /etc/file1
# ls -Z /etc/file1
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0 /etc/file1
```

如下， `/usr/sbin/semanage fcontext -a -t samba_share_t /etc/file1` 命令增加到 `/etc/selinux/targeted/contexts/files/file_contexts.local` 的入口：

```
/etc/file1      unconfined_u:object_r:samba_share_t:s0
```

- 3) 作为 root 用户，运行 `/sbin/restorecon -v /etc/file1` 命令来改变类型。因为 `semanage` 命令为 `/etc/file1` 增加了一个到 `file.contexts.local` 的入口，所以 `/sbin/restorecon` 命令将类型改变为 `samba_share_t`：

```
# /sbin/restorecon -v /etc/file1
restorecon reset /etc/file1 context unconfined_u:object_r:etc_t:s0-
>system_u:object_r:samba_share_t:s0
```

- 4) 作为 root 用户，运行 `rm -i /etc/file1` 命令删除 `file1` 文件。
- 5) 作为 root 用户，运行 `/usr/sbin/semanage fcontext -d /etc/file1` 命令删除为 `/etc/file1` 增加的上下文。当删除上下文时，运行 `restorecon` 将类型改变为 `etc_t`，而不是 `samba_share_t`：

改变目录的类型

下例描述了如何创建新目录和改变目录的文件类型为 Apache HTTP 服务器使用的文件类型。

- 1) 作为 Linux root 用户，运行 `mkdir /web` 命令来创建一个新目录。此目录标记为 `default_t` 类型。

```
# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0  /web
```

`ls -d` 选项使 `ls` 列出关于一个目录而不是关于其内容的信息，`-Z` 选项使 `ls` 显示 SELinux 上下文（本例为 `unconfined_u:object_r:default_t:s0`）。

- 2) 作为 Linux root 用户，运行 `/usr/sbin/semanage fcontext -a -t httpd_sys_content_t /web` 命令从 `/web` 类型变更到 `httpd_sys_content_t`。 `-a` 选项增加一个新记录，`-t` 选项定义一个类型(`httpd_sys_content_t`)。注意，运行该命令并不直接改变目录类型——`/web` 仍标记为 `default_t` 类型：

```
# /usr/sbin/semanage fcontext -a -t httpd_sys_content_t      /web
# ls -dZ /web
drwxr-xr-x.  root root unconfined_u:object_r:default_t:s0      /web
```

如下，`/usr/sbin/semanage fcontext -a -t httpd_sys_content_t /web` 命令增加到 `/etc/selinux/targeted/contexts/files/ file_contexts.local` 的入口：

```
/web unconfined_u:object_r:httpd_sys_content_t:s0
```

- 3) 作为 Linux root 用户，运行 `/sbin/restorecon -v /web` 命令来变更类型。因

为 `semanage` 命令为 `/web` 增加了一个到 `file.contexts.local` 的入口，所以 `/sbin/restorecon` 命令将类型变更为 `httpd_sys_content_t`：

```
# /sbin/restorecon -v /web
restorecon reset /web context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

默认情况下，新创建的文件和目录继承其父文件夹的 SELinux 类型。当使用该示例时，在删除 `web`/增加的 SELinux 上下文前，在 `/web/` 目录中创建的文件和目录均标记为 `httpd_sys_content_t` 类型。

- 4) 作为 Linux root 用户，运行 `/usr/sbin/semanage fcontext -d /web` 命令来删除为 `/web`/增加的上下文。
- 5) 作为 Linux root 用户，运行 `/sbin/restorecon -v /web` 命令来恢复默认的 SELinux 上下文。

变更目录及其内容类型

下例描述了如何创建新目录，并将目录内文件类型（和它的内容）变更为 Apache HTTP 服务器所使用的类型。若您想要 Apache HTTP 服务器使用不同的文档根目录（而不是 `/var/www/html/`），请使用本例中配置：

- 1) 作为 root 用户，运行 `mkdir /web` 命令创建新目录，然后运行 `touch /web/file{1,2,3}` 命令创建 3 个空文件(file1、 file2 和 file3)。`/web/` 目录和其中的文件都被标记为 `default_t` 类型：

```
# ls -dZ /web
drwxr-xr-x root root unconfined_u:object_r:default_t:s0 /web
# ls -lZ /web
-rw-r--r-- root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r-- root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r-- root root unconfined_u:object_r:default_t:s0 file3
```

- 2) 作为 root 用户，运行 `/usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"` 命令来变更 `/web/` 目录和其中的文件的类型。`-a` 选项增加一个记录，`-t` 选项定义一个类型(`httpd_sys_content_t`)。 `"/web(/.*)?"` 正则表达式使 `semanage` 命令将变更应用到 `/web/` 目录和其中的文件。注意：运行此命令并不能直接改变类型——`/web/` 和其中的文件仍被标记为 `default_t` 类型：

```
# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web

# ls -lZ /web
-rw-r--r--.  root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--.  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--.  root root unconfined_u:object_r:default_t:s0 file3
```

/usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"命令增加了如下到/etc/selinux/targeted/contexts/files/ file_contexts.local 的入口:

```
/web(/.*)?    system_u:object_r:httpd_sys_content_t:s0
```

- 3) 作为 root 用户, 运行/sbin/restorecon -R -v /web 命令来变更/web/目录及其文件的类型。-R 是递归的意思, 表明/web/目录下的所有文件和目录都被标记为 httpd_sys_content_t 类型。因为 semanage 命令为 /web(/.*)? 增加一个到的 file.contexts.local ,的入口, /sbin/restorecon 命令将类型变更为 httpd_sys_content_t:

```
# /sbin/restorecon -R -v /web
restorecon reset /web context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file2 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file3 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file1 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

默认情况下, 新创建的文件和目录继承其父目录的 SELinux 类型。本例中, /web/目录中创建的文件和目录都将被标记为 httpd_sys_content_t 类型。

- 4) 作为 root 用户, 运行/usr/sbin/semanage fcontext -d "/web(/.*)?"命令删除为"/web(/.*)?"增加的上下文。
- 5) 作为 root 用户, 运行/usr/sbin/semanage fcontext -d "/web(/.*)?"命令恢复默认的 SELinux 上下文。

删除增加的上下文

下例展示了如何增加和删除 SELinux 上下文:

- 1) 作为 root 用户, 运行/usr/sbin/semanage fcontext -a -t httpd_sys_content_t

/test 命令。/test/目录不要求必须存在。

该命令向/etc/selinux/targeted/contexts/files/file_contexts.local 增加如下上下文。

```
/test    system_u:object_r:httpd_sys_content_t:s0
```

- 2) 要删除上下文，作为 root 用户，运行/usr/sbin/semanage fcontext -d file-name|directory-name 命令，其中 file-name|directory-name 是 file_contexts.local 的第一部分。下面是 file_contexts.local 中上下文的示例：

```
/test    system_u:object_r:httpd_sys_content_t:s0
```

第一部分是/test。防止目录/test/在运行/sbin/restorecon 之后或文件系统重新标记后被标记为 httpd_sys_content_t，作为 root 用户，运行下面命令从 file_contexts.local 中删除上下文：

```
/usr/sbin/semanage fcontext -d /test
```

若上下文是正则表达式的一部分，例如，/web(/.)*?，正则表达式两边使用引号：

```
/usr/sbin/semanage fcontext -d "/web(/.)*?"
```

关于/usr/sbin/semanage 的更多信息，请参阅 semanage(8)手册页。



重要：使用/usr/sbin/semanage fcontext -a 改变 SELinux 上下文时，请使用文件或目录的完整路径来避免在文件系统被重新标记或执行/sbin/restorecon 命令后文件被误标记。

3.4.8 ile_t 和 default_t 类型

对于支持扩展属性的文件系统，当访问磁盘上一个缺少 SELinux 上下文的文件时，认为它有一个 SELinux 策略定义的默认上下文。在通用策略中，这种默认上下文使用 file_t 类型。这是唯一使用这种情况，所以磁盘上没有上下文的文件在策略内被区分，并通常不可访问限制域。因为运行 SELinux 的系统的文件都应有一个 SELinux 上下文，而且 file_t 类型从不用于文件上下文配置，所以 file_t 类型不应存在于正确标记的文件系统中。

default_t 类型用于与文件上下文配置中任何模式都不匹配的文件，从而使这样的文件与在磁盘上没有上下文的文件有所区别，它们通常不可访问限制域。若您创建一个顶层目录，如/mydirectory/，这个目录可被标记为 default_t 类

型。若服务需要访问这样一个目录，请更新文件上下文配置的位置信息。

3.4.9 安装文件系统

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。

3.4.9.1 上下文安装

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 SELinux `targeted` 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。

3.4.9.2 变更默认上下文

如章节“3.4.8 `file_t` 与 `default_t` 类型”中所述，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，则认为其有 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”。
- 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/` 下创建的文件将保持它们的标记。

3.4.9.3 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取和输出。

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` 选项挂载文件系统的可替代方法，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，请参阅章节“3.4.6.3 NFS 与 CIFS 布尔变量”。

3.4.9.4 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:mysql_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:mysql_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。

3.4.9.5 使上下文挂载持久

若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
```

3.4.10 挂载 SELinux 标记

这些章节描述了复制、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。

3.4.10.1 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/目录创建一个新文件。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 上下文可以阻止进程访问这样的文件和目录。

3.4.10.2 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，阻止如 Apache HTTP 服务器和 Samba 的进程访问这样的文件和目录。

3.4.10.3 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
```

3.4.10.4 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。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。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 不能被用到：

```
$ 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 /test/目录，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。

3.4.10.5 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。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` 不能被用到：

```
$ 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/test/目录, 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。

3.4.11 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, 它以 policy.conf 文件或二进制策略文件作为输入。

seinfo 的输出会因二进制文件和源文件的不同而不同。例如, 策略源文件使用{ }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 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，并且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。若要查询非限制域的数量，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 ~]$ sestatus --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;
```

sestatus 命令能提供 allow 规则的数量:

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

dontaudit 规则的数量是:

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

3.5 限制用户

在中标麒麟高级服务器操作系统中可以使用大量限制的 SELinux 用户。每个 Linux 用户都通过 SELinux 策略映射为一个 SELinux 用户，允许 Linux 用户继承对 SELinux 用户设置的限制。例如，根据不同用户，不能执行以下操作：运行 X

Window 系统、使用网路、运行 `setuid` 应用程序（除非 SELinux 策略允许）或者运行 `su` 和 `sudo` 命令。这有助于保护系统不会被用户破坏。有关限制用户的更多信息，请参阅“3.3.3 限制与非限制用户”。

3.5.1 Linux 与 SELinux 用户映射

以 root 用户身份运行 `semanage login -l` 命令来查看 Linux 用户和 SELinux 用户之间的映射：

```
# /usr/sbin/semanage login -l

Login Name  SELinux User    MLS/MCS Range
default     unconfined_u    s0-s0:c0.c1023
root        unconfined_u    s0-s0:c0.c1023
system_u    system_u         s0-s0:c0.c1023
```

在中标麒麟高级服务器操作系统中，Linux 用户默认映射为 SELinux `default_login`，依次映射为 SELinux `unconfined_u` 用户。当用 `useradd` 命令创建一个 Linux 用户时，如果没有指定选项，则此用户映射为 SELinux `unconfined_u` 用户。下面定义默认映射：

default	unconfined_u	s0-s0:c0.c1023
---------	--------------	----------------

3.5.2 限制新 Linux 用户：useradd

映射为 SELinux `unconfined_u` 用户的 Linux 用户运行在 `unconfined_t` 域。这可以通过以映射为 `unconfined_u` 的 Linux 用户身份登录时运行 `id -Z` 命令来查看：

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Linux 用户运行在 `unconfined_t` 域时，应用 SELinux 策略规则，但策略规则允许运行在 `unconfined_t` 域的 Linux 用户具有几乎所有访问权限。如果非限制 Linux 用户执行由 SELinux 策略定义的可从 `unconfined_t` 域迁移到自己限制域的应用程序，非限制 Linux 用户仍受到该限制域的约束。这样做在安全方面的好处在于，即使一个 Linux 用户是非限制运行的，应用程序仍会受到限制，因此可以通过策略限制应用程序中的缺陷被利用。注意：这并不会使系统免于受到用户攻击。而是保护用户和系统免于受到可能由应用程序中的缺陷而造成的破坏。

使用 `useradd` 创建 Linux 用户时，使用 `-Z` 选项来指定用户映射的 SELinux 用户。下例创建一个新的 Linux 用户 `useruuser`，并将此用户映射为 SELinux `user_u`

用户。映射为 SELinux user_u 用户的 Linux 用户运行在 user_t 域。在此域中，除非 SELinux 策略允许（如 passwd），否则 Linux 用户不能运行 setuid 应用程序，也不能运行 su 或 sudo，从而防止 Linux 用户使用这些命令成为 root 用户。

- 1) 以 root 用户身份运行 `/usr/sbin/useradd -Z user_u useruser` 命令来创建一个新的 Linux 用户(useruser)，其映射为 SELinux user_u 用户。
- 2) 以 root 用户身份运行 `semanage login -l` 命令来查看 Linux useruser 用户和 user_u 之间的映射。

```
# /usr/sbin/semanage login -l
Login Name SELinux User MLS/MCS Range
default      unconfined_u s0-s0:c0.c1023
root unconfined_u s0-s0:c0.c1023
system_u system_u s0-s0:c0.c1023
useruser user_u s0
```

- 3) 以 root 用户身份运行 `passwd useruser` 命令来为 Linux useruser 用户分配一个密码。

```
# passwd useruser
Changing password for user useruser.
New UNIX password: Enter a password
Retype new UNIX password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

- 4) 注销当前会话，并以 Linux useruser 用户身份登录。登录时，pam_selinux 将 Linux 用户映射为一个 SELinux 用户（本例中为 user_u），并建立相应的 SELinux 上下文。然后通过此上下文启动 Linux 用户的 shell。运行 `id -Z` 命令来查看 Linux 用户的上下文。

```
[useruser@localhost ~]$ id -Z
user_u:user_r:user_t:s0
```

- 5) 注销 Linux useruser 的会话，并使用您的帐户重新登录。若您不需要 Linux useruser 用户，以 root 用户身份运行 `/usr/sbin/userdel -r useruser` 命令将其删除，并删除其主目录。

3.5.3 限制已有 Linux 用户：semanage 登录

若 Linux 用户映射为 SELinux unconfined_u 用户（默认行为），且您希望更

改它们映射的 SELinux 用户，使用 `semanage login` 命令。下例创建一个新 Linux 用户 `newuser`，然后将此 Linux 用户映射为 SELinux `user_u` 用户：

以 `root` 用户身份运行 `/usr/sbin/useradd newuser` 命令来创建一个新 Linux 用户 (`newuser`)。因为用户使用默认映射，所以它不出现在 `/usr/sbin/semanage login -l` 输出中：

```
# /usr/sbin/semanage login -l

Login Name  SELinux User    MLS/MCS Range
default     unconfined_u    s0-s0:c0.c1023
root unconfined_u s0-s0:c0.c1023
system_u system_u s0-s0:c0.c1023
```

要将 Linux `newuser` 用户映射为 SELinux `user_u` 用户，以 `root` 用户身份运行下面的命令：

```
/usr/sbin/semanage login -a -s user_u newuser
```

`-a` 选项增加一个新记录，`-s` 选项指定 Linux 用户要映射的 SELinux 用户。最后一个参数 `newuser`，是您想要映射为指定 SELinux 用户的 Linux 用户。

要查看 Linux `newuser` 用户和 `user_u` 之间的映射，以 `root` 用户身份运行命令 `semanage login -l`：

```
# /usr/sbin/semanage login -l

Login Name  SELinux User    MLS/MCS Range
default     unconfined_u    s0-s0:c0.c1023
newuser user_u    s0
root unconfined_u s0-s0:c0.c1023
system_u system_u s0-s0:c0.c1023
```

以 `root` 用户身份运行 `passwd newuser` 命令来为 Linux `newuser` 用户分配一个密码：

```
# passwd newuser
Changing password for user newuser.
New UNIX password: Enter a password
Retype new UNIX password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

注销您的当前会话，并以 Linux `newuser` 用户身份登录。运行 `id -Z` 命令来查看 `newuser` 的 SELinux 上下文：

```
[newuser@localhost ~]$ id -Z
user_u:user_r:user_t:s0
```

注销 Linux newuser 的会话，并使用您帐户重新登录。若您不想要 Linux newuser 用户，以 root 用户身份运行 `userdel -r newuser` 命令来将其删除，并删除其主目录。另外还会删除 Linux newuser 用户和 user_u 之间的映射：

```
# /usr/sbin/userdel -r newuser
# /usr/sbin/semanage login -l

Login Name SELinux User      MLS/MCS Range
default      unconfined_u s0-s0:c0.c1023
root         unconfined_u s0-s0:c0.c1023
system_u     system_u s0-s0:c0.c1023
```

3.5.4 更改默认映射

中标麒麟高级服务器操作系统中，Linux 用户默认映射为 SELinux ---default--- login，依次映射为 SELinux unconfined_u 用户)。若您想要用新的 Linux 用户，并且希望 Linux 用户特定不映射为默认限制的 SELinux 用户，使用 `semanage login` 命令更改默认映射。

例如，作为 root 用户运行下面的命令来更改从 unconfined_u 到 user_u 的默认映射：

```
/usr/sbin/semanage login -m -S targeted -s "user_u" -r s0 default
```

以 root 用户身份运行 `semanage login -l` 命令来确认 default login 已映射为 user_u：

```
# /usr/sbin/semanage login -l

Login Name SELinux User      MLS/MCS Range
default      user_u s0
root         unconfined_u s0-s0:c0.c1023
system_u     system_u s0-s0:c0.c1023
```

若创建了新的 Linux 用户，而没有指定 SELinux 用户，或者以一个已有的 Linux 用户登录，且没有从 `semanage login -l` 输出中匹配一个特定的入口，根据 default login，它们映射为 user_u。

要重新更改为默认行为，以 root 用户身份运行下面的命令来将 default login 映射为 SELinux unconfined_u 用户：

```
/usr/sbin/semanage login -m -S targeted -s "unconfined_u" -r\  
s0-s0:c0.c1023 default
```

3.5.5 xguest: Kiosk 模式

xguest 包提供了 kiosk 用户帐户。此帐户用于保护人们走近和使用的机器，如图书馆、银行、机场、信息亭和咖啡厅里的机器。Kiosk 用户帐户受到极大限制：实际上，它只允许用户登录和使用 Firefox 浏览网络。使用该帐户登录时所做的任何更改，如创建文件或更改设置，注销后都将丢失。

要设置 kiosk 帐户：

- 1) 以 root 用户身份运行 `yum install xguest` 命令来安装 xguest 包。按需要安装依赖项。
- 2) 为允许不同人群使用 kiosk 帐户，此帐户没有设置密码保护，因而，此帐户只能在 SELinux 运行在强制模式下时受到保护。使用该帐户登录前，使用 `getenforce` 命令来确认 SELinux 正运行在强制模式下：

```
$ /usr/sbin/getenforce  
Enforcing
```

如果情况不是这样，请参阅“3.4.5 SELinux 模式”，了解更改为强制模式的详细信息，如果 SELinux 处于允许模式或禁用模式，则无法以此帐户登录。

- 3) 您只可以通过 GNOME 显示管理器 (GDM) 登录到此帐户。一旦安装了 xguest 包，Guest 帐户就添加到 GDM 登录显示器。

3.5.6 用户执行应用程序的布尔值

不允许 Linux 用户在其有写入权限的主目录和/tmp/中执行应用程序(继承了用户的权限)，这有助于防止缺陷或恶意应用程序修改用户拥有的文件。在中标麒麟高级服务器操作系统中，默认情况下，guest_t 和 xguest_t 域中的 Linux 用户不能执行其主目录或/tmp/目录的应用程序，然而，默认情况下，user_t 和 staff_t 域的 Linux 用户则可以这样操作。

布尔值可用于更改这种行为，布尔值可用 `setsebool` 命令进行配置。必须以 root 用户身份运行 `setsebool` 命令。`setsebool -P` 命令能够进行永久性更改。若在重启期间您不想进行永久性更改，请勿使用 -P 选项。

guest_t

允许 guest_t 域中的 Linux 用户执行他们主目录和/tmp/目录中的应用程序：

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

xguest_t

要允许 xguest_t 域中的 Linux 用户在其主目录和/tmp/目录中执行应用程序：

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

user_t

要防止 user_t 域中的 Linux 用户在其主目录和/tmp/目录中执行应用程序：

```
/usr/sbin/setsebool -P allow_user_exec_content off
```

staff_t

要防止 staff_t 域中的 Linux 用户在其主目录和/tmp/目录中执行应用程序：

```
/usr/sbin/setsebool -P allow_staff_exec_content off
```

3.6 sVirt

sVirt 是内置在中标麒麟高级服务器操作系统中集成了 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成为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高级服务器操作系统中的。

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 服务器等服务器组成。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，直接向网络提供服务。图 3-1 描绘一个非虚拟化的环境：

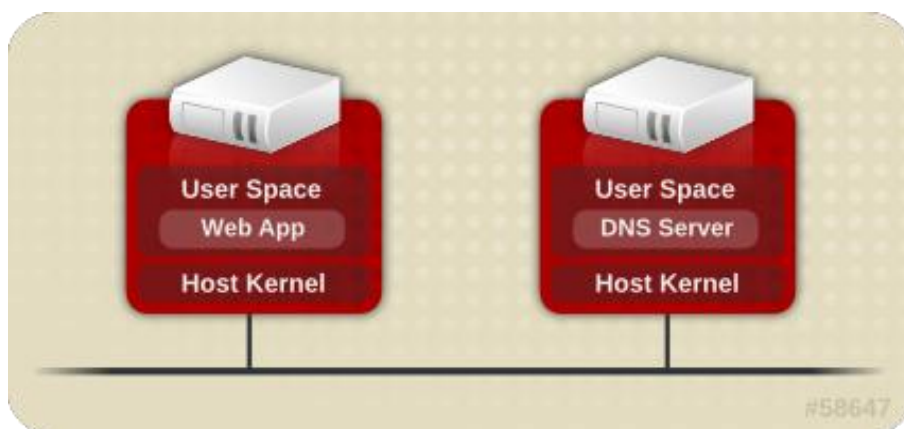


图 3-1 非虚拟化环境

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 3-2 描绘一个虚拟化环境：

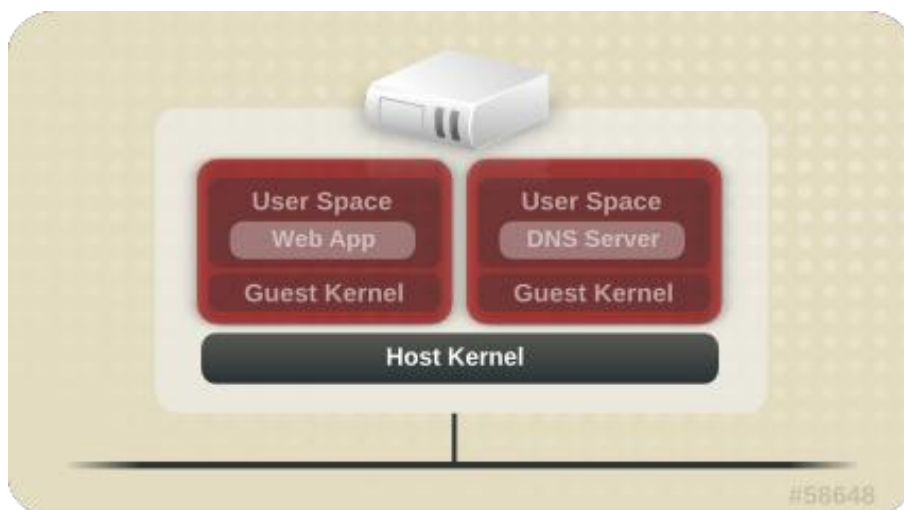


图 3-2 虚拟化环境

3.6.1 安全与虚拟化

服务未虚拟化时，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，限制他们启动进一步攻击。图 3-3 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。

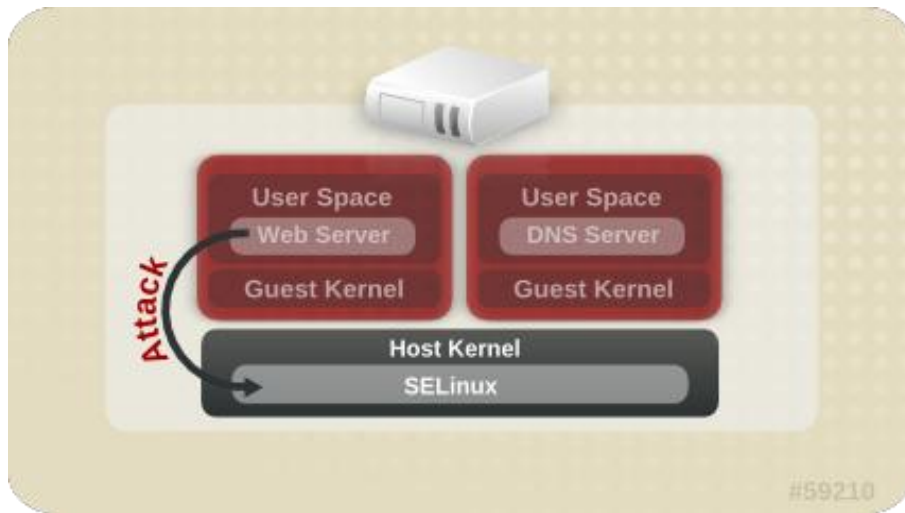


图 3-3 虚拟机攻击

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。

3.6.2 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 3-2 列出了使用 sVirt 时可分配的不同标记：

表 3-2 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	t: Asl0l svirt_t 进程可以读带有此标记的文件/设备。
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。

3.7 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。

3.7.1 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 3-3 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 拒绝访问时会显示一个警告信息：



图 3-4 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

3.7.2 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。

3.7.2.1 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/。中标麒麟高级服务器操作系统 V6.4 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“3.4.7.2 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“3.4.10.3 检查默认的 SELinux 上下文”。

3.7.2.2 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是 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。

3.7.2.3 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 可能没有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目前还未发现的，这可能造成即使应该允许访问，SELinux 也会拒绝访问。

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“3.7.3.8 允许访问: audit2allow”。

3.7.3 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。

3.7.3.1 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。

有关管理 Linux 权限的信息，请参阅《中标麒麟高级服务器操作系统文档项目“权限”》草案。

3.7.3.2 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。

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 sestatus -- dontaudit 命令，使用 -s domain 选项和 grep 命令限制搜索范围：

```
$ sestatus --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“3.7.3.6 原始监听消息”和“3.7.3.7 sealert 消息”。

3.7.3.3 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。`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。

3.7.3.4 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。

中标麒麟高级服务器操作系统包括了许可域，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。

中标麒麟高级服务器操作系统 V4 和 V5 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
```

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
aid=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 aid=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”。

3.7.3.5 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“3.4.2 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”。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 3-4 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”，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 3-5 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。

3.7.3.6 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(SYSSCALL)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。

3.7.3.7 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 Context `unconfined_u:object_r:samba_share_t:s0`

Target Objects `/var/www/html/file1 [ file ] Source httpd`

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。

3.7.3.8 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”。根据章节“3.7.3.7 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=400000003 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-list11](#)。

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
```

4 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 and 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。

4.1 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。

4.2 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。

4.2.1 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。

4.2.2 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。

4.3 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。

4.4 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 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，中标麒麟高级服务器操作系统中的 IPv4 策略禁用了对 IP 转发的支持，这会防止运行中标麒麟高级服务器操作系统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。

4.4.1 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。

4.5 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
```

4.6 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。

中标麒麟高级服务器操作系统支持使用 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/>。

4.7 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一链，与 Netfilter 应用于包的相关动作关联。

filter table 中的内置链如下：

- 1) INPUT — 应用于定向到主机(host)的网络包。
- 2) OUTPUT — 应用于本地生成的网络包。
- 3) FORWARD — 应用于经由主机(host)传输的网络包。

nat table 中的内置链如下：

- 1) PREROUTING — 转换到达的网络包。
- 2) OUTPUT — 在发出前转换本地生成的网络包。
- 3) POSTROUTING — 在发出前转换网络包。

mangle table 中的内置链如下：

- 1) INPUT — 根据目标主机转换网络包。
- 2) OUTPUT — 在发出前转换本地生成的网络包。
- 3) FORWARD — 转换路径主机的网络包。
- 4) PREROUTING — 在发出前转换收到的网络包。
- 5) POSTROUTING — 在发出前转换网络包。

4.8 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：

- 1) Packet Type — 定义该命令适用的包类型。
- 2) Packet Source/Destination — 定义适用包的来源或目的地址。
- 3) 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。

4.8.1 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。

4.8.2 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。

4.8.3 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”。

4.8.4 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)。

4.8.5 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。

4.8.6 UDP 协议

以下匹配选项用于 UDP 协议(-p udp):

--dport — 设置 UDP 包的端口；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。可参阅“4.8.5 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。可参阅“4.8.5 TCP 协议”中的相关使用。

4.8.7 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。

4.9 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。

4.10 IPTables 控制脚本(IPTables control 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_NUMERI`

C 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“4.9 如何保存 iptables 规则”以获取更多信息。

5 网络入侵检测系统

网络受到入侵或网络资源被滥用是不可避免的，管理员可以采取一些积极措施来防止安全入侵的发生。例如组织一个能够快速有效地对安全问题做出响应的紧急情况响应组。

计算机系统和数据需要有相当的安全保证。互联网推进了各类信息的流动，其中包括个人信息和经济信息。与此同时，它也孕育了多种危险。企图不良的用户和骇客会试图寻找较脆弱的目标，例如：未应用补丁的系统、被特洛伊木马病毒感染的系统、以及运行不安全服务的网络。管理员和安全组的成员在系统被攻击时应该收到警报，这样，他们才能够立刻对此类威胁做出相应的反应。入侵检测系统(intrusion detection systems, IDS)就被设计成这样的警报系统。

5.1 入侵检测系统

入侵检测系统(IDS)是一种分析系统和网络上未经授权的进入和(或)有不良企图的活动的积极进程或设备。IDS 检测异常情况的方法可能会大相径庭；但是它们的最终目的都是在攻击者还未对您的系统造成损害前当场捕捉他们。

IDS 帮助系统免受攻击、滥用和弱化。它还能够监视网络活动、审查网络和系统配置中的弱点、分析数据完好性等等。根据您的选择要使用的检测方法而定，使用 IDS 有好几种直接和间接的优越性。

5.1.1 TripWire

Tripwire 是最流行的用于 Linux 的基于主机的 IDS。Tripwire, Inc. 是 Tripwire 的开发商，它们新近为 Linux 版本按照 GNU 通用公共许可证的条款开放了源码。Tripwire 在 <http://www.tripwire.org/> 上可以获得。

5.1.2 RPM 作为一种 IDS

RPM 软件包管理器(RPM)是另一个可以被用作基于主机的 IDS 的程序。RPM 包含各类用于查询软件包及其内容的选项。对于那些怀疑重要的系统文件和可执行文件已被修改的管理员来说，这些校验选项具有很高的价值。

以下列表详细地描述了一些可以用来校验您的中标麒麟高级服务器操作系统上的文件完好性的 RPM 选项。

```
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-neokylin-release
```

`rpm -V` 软件包名称

`-V` 选项校验已安装了的叫做“`package_name`”的软件包中的文件。如果该命令没有显示任何输出而退出，这就意味着其中所有文件自从最后一次 RPM 数据库更新以来都没有被修改。如果该命令给出了错误，例如：

S.5....T c /bin/ps

那么，这就说明该文件已被修改了。您需要决定是应该保留它(如果被修改的文件是 `/etc` 中的配置文件)还是删除它再重新安装包含这个文件的软件包。以下列表解释了表示校验失败的这八个字符(上面例子中是 S.5....T)的含义。

- . — 已通过这一阶段的校验测试。
- ? — 测试中发现了一份无法读取的文件，这极可能是文件权限问题。
- S — 测试中发现了一份比最初安装在系统上的文件稍大或稍小的文件。
- 5 — 测试中发现了一份和最初安装时的 md5 校验和不匹配的文件。
- M — 测试中检测到文件权限或文件类型错误。
- D — 测试中检测到主/次号码不匹配的设备文件。
- L — 测试中找到一个被改变到另一个文件路径的符号链接。
- U — 测试中找到一份用户所有者被改变的文件。
- G — 测试中找到一份组群所有者被改变的文件。
- T — 测试中遇到文件的 mtime 校验错误。

`rpm -Va`

`-Va` 选项校验所有安装了的软件包，并找出校验测试中的失败之处(和 `-V` 选项相似)，但是由于它校验每个安装了的软件包，其输出要比 `-V` 选项更详细。

`rpm -Vf /bin/ls`

`-Vf` 选项校验某个安装了的软件包中的单个文件。如果您打算只快速地校验一个有疑点的文件，这个选项就很有用。

`rpm -K application-1.0.i386.rpm`

`-K` 选项检查 RPM 软件包文件的 md5 校验和与 GPG 签名。

application-1.0.i386.rpm (SHA1) DSA sha1 md5 (GPG) NOT OK
(MISSING KEYS: GPG#897da07a)

PRM 是一个强大的工具，它所提供的诸多对安装了的软件包和 RPM 软件包文件的校验工具就足以证明这一点。我们强烈建议您在安装了中标麒麟高级服

务器操作系统后,把 RPM 数据库目录(/var/lib/rpm/)备份到只读介质(如光盘)上。这么做会使您能够根据只读数据库而不是您的系统上的数据库来校验文件和软件包,因为有不良企图的用户可能会破坏您的数据库来歪曲校验结果。

5.2 基于网络的 IDS

基于网络的入侵检测系统和基于主机的入侵检测系统的工作方式不同。基于网络的 IDS 的设计哲学是在路由器或主机级别扫描网络分组、审查分组信息、并在一个特殊文件中详细记录可疑分组。根据这些可疑分组,基于网络的 IDS 可以扫描它自己的已知网络攻击特征数据库,并为每个分组指定严重级别。如果严重级别够高,它就会给安全组的成员发送电子邮件或通知传呼机,因此安全组的成员就可以进一步调查这些异常特点。

随着互联网在其范围和交通流量方面的增大,基于网络的 IDS 已经越来越受欢迎。在安全行业中,能够扫描大量网络活动并成功地标记可疑传输的 IDS 很受好评。鉴于 TCP/IP 协议所固有的安全问题,开发扫描器、嗅探器、以及其它网络评审和检测工具来防止由蓄意不良的网络活动所带来的安全破坏已显得极端重要。这些活动包括:

- 1) IP 伪装(Spoofing)。
- 2) “拒绝服务”攻击。
- 3) arp 缓存污染。
- 4) DNS 名称破坏。
- 5) 中间人攻击。

多数基于网络的 IDS 需要把主机系统网络设备设置为混杂(promiscuous)模式。该模式允许设备捕捉每个经过网络的分组。混杂模式可以通过 ifconfig 命令来设置,例如:

```
ifconfig eth0 promisc
```

运行不带选项的 ifconfig 会显示出 eth0 现在处于混杂(PROMISC)模式。

```
eth0 Link encap:Ethernet HWaddr 00:00:D0:0D:00:01
inet addr:192.168.1.50 Bcast:192.168.1.255 Mask:255.255.252.0
UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
RX packets:6222015 errors:0 dropped:0 overruns:138 frame:0
TX packets:5370458 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:100
RX bytes:2505498554 (2389.4 Mb) TX bytes:1521375170 (1450.8 Mb)
Interrupt:9 Base address:0xec80
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:21621 errors:0 dropped:0 overruns:0 frame:0
TX packets:21621 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1070918 (1.0 Mb) TX bytes:1070918 (1.0 Mb)
```

使用一个类似于 `tcpdump`(包括在中标麒麟高级服务器操作系统中)的工具, 我们能够看到网络中的大量数据或通信:

```
tcpdump: listening on eth0
02:05:53.702142 pinky.example.com.ha-cluster > \
heavenly.example.com.860: udp 92 (DF)
02:05:53.702294 heavenly.example.com.860 > \
pinky.example.com.ha-cluster: udp 32 (DF)
02:05:53.702360 pinky.example.com.55828 > dns1.example.com.domain: \
PTR? 192.35.168.192.in-addr.arpa. (45) (DF)
02:05:53.702706 ns1.example.com.domain > pinky.example.com.55828: \
6077 NXDomain* 0/1/0 (103) (DF)
02:05:53.886395 shadowman.example.com.netbios-ns > \
172.16.59.255.netbios-ns: NBT UDP PACKET(137): QUERY; BROADCAST
02:05:54.103355 802.1d config c000.00:05:74:8c:a1:2b.8043 root \
0001.00:d0:01:23:a5:2b pathcost 3004 age 1 max 20 hello 2 fdelay 15
02:05:54.636436 konsole.example.com.netbios-ns > 172.16.59.255.netbios-ns:\
NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
02:05:56.323715 pinky.example.com.1013 > heavenly.example.com.860:\
udp 56 (DF)
02:05:56.323882 heavenly.example.com.860 > pinky.example.com.1013:\
udp 28 (DF)
```



注意: `tcpdump` 将同样扫描并记录那些目的地不是我们这台主机的包(例如: `pinky.example.com`)。

6 NMAP 的使用

一个典型的评估可以通过某种信息收集工具来进行。对于一个网络，我们首先绘制出它的整个构架，找到正在运行的主机，定位后再逐一单独检测这些主机。分析这些主机可能又需要一系列其它的工具。因此，在寻找问题的时候，知道利用哪些工具来实现目的非常关键。

就像日常生活中一样，可以有多种不同的工具来完成同样的工作，在计算机中也一样，有各种专门的工具应用于操作系统、应用程序，或者是网络。有些工具是免费的，有些不是。有些工具直观而易用，而有些可能难懂又缺少文档资料，但却具有自己独特的性能。

找到一个合适的工具可能会是一件艰苦的任务，而且还要靠很多的经验。如果可能，您也许可以进行一项实验，尽可能找来所有可用的工具并逐一分析它们各自的优缺点，研究它们的文档或用户手册。此外，还可以到 Internet 上搜寻更多的诸如文章、进阶教程等技术资料，甚至获取所能提供的邮递资料。以下介绍的是其中一种可用的工具。

6.1 用 NMAP 扫描主机

Nmap 是 linux 系统中包含的一个用以获取网络架构的有效工具，它已经发展了很多年并成为应用最广泛的一个收集网络信息的工具。它具有完善的使用手册，其中详细描述了有关的选项和使用。管理员可以使用 Nmap 查找网络中的主机系统并打开其端口。

Nmap 在网络弱点评估中迈出了很有竞争力的第一步。您可以通过 Nmap 查找到网络中的所有主机，甚至可以通过传递一个参数而识别正在运行的操作系统。Nmap 对于建立使用安全服务和停掉无用服务的策略提供了很好的基础。

6.2 NMAP 的使用

Nmap 可以在 shell 环境下使用，键入 nmap 命令，后面接需要探测的主机名或 IP 地址。

```
nmap foo.example.com
```

检测的结果形式如下(因主机位置的不同，可能需要几分钟的时间):

```
Starting Nmap 4.11 ( www.insecure.org/nmap/ )
```

```
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1591 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open  ssh
25/tcp open  smtp
111/tcp open sunrpc
515/tcp open  printer
950/tcp open oftp-rpc
6000/tcp open X11
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Nmap 可用来测试出最常用于服务的端口，这能够帮助管理员关闭一些无用的服务器。您还可以通过以下官方网站获取更多关于 Nmap 的信息：
<http://www.insecure.org/>。