Getting Started with SE Linux HOWTO:

the new SE Linux

（简体中文版）

Getting Started with SE Linux HOWTO: the new SE Linux

（译者注：本文的最原始版本为 2004 年 3 月所写，此份 HOWTO 是作者在今年 2 月根据最新的 SE Linux 所作的修改后的版本。新的 SE Linux 与以前的有比较大的变化，而且这项技术本身也正在飞速的发展，并未最后成熟。阅读本文是需要对 Linux 本身有一定深度的了解作为基础的。本文并不是 Linux 的初级教程，但却是 SE Linux 技术的初级教程。）

原著：Faye Coker, March 2004.
faye@lurking-grue.org

中文译者：邹立巍, 2006 年 7 月
mini.jerry@gmail.com;

重要的提示！注意！
我正在根据最新的 SE Linux 来根本的升级这份文档。我用了很长时间来做这件事情，不过看来我好像永远没有足够的时间作完它。我现在正在继续做，请相信我！
Faye Coker, Feb 04, 2006
（译者注：本人技术出身，英语实在比较差。翻译如有不当之处，敬请指正！）

这份文档已经根据最新的 SE Linux 做了更改。旧的"Getting Started with SE Linux HOWTO"的内容将保留在此份文档里，不过绝大多数的内容是根据最新的 SE Linux 的特点作了修改。新的 SE Linux 是基于 2.6.*内核的，但是仍然支持 2.4.*的内核。这份文档的大部分内容是原来的，我在需要修改的地方做了调整。

这份文档是美国国家安全局的安全加强的 Linux（NSA SE Linux）的概述性的说明。我们主要的环境是基于 Debian Linux 的，而且大部分的软件包的操作命令实例都是基于 Debian 的。这份文档主要是针对那些想要基础了解 SE Linux 的人，所以这里没有对 SE Linux 比较进介的介绍。你可以在附录的**资源**部分找到其它介绍 SE Linux 的资料。

这份文档已经被Ivan Pesin翻译成了俄文。你可以在 http://gazette.linux.ru.net/rus/articles/intro_selinux.html找到俄文的版本。谢谢Ivan做了这些。

目录

1. 介绍

这份文档是一个 SE Linux 的简介，可以指导一部分人初步的学会 SE Linux。它涵盖和解释了 SE Linux 的各方面的术语,安装和添加用户并且涵盖了一小部分别的知识。一个更高级的帮助文档将会在不久发布（译者注：正在翻译中），包含了如何编辑策略等内容。（which causes a little too much information overload with users new to SE Linux and is not included here).

1.1. 欢迎反馈！

我们欢迎对这份文档的反馈信息，请发邮件给 faye@lurking-grue.org （中文的就给我吧！；）mini.jerry@gmail.com）

1.2. 注意！

这份文档只是一份指导。我强烈的建议你在实际工作的机器上应用之前先找一台试验机器来做练习 。

1.3. 最新的 SE Linux 的特点

最新的 SE Linux 有一些新的特点，下面先介绍一下：

/selinux 文件系统
加入了一个*/selinux* 文件系统. 因此有些安装程序需要你编辑*/etc/fstab* 文件。 */selinux* 文件系统和 */proc* 文件系统类似，都是虚拟的文件系统。你可以用 *ls -l /selinux* 命令来显示。

```
total 0
-rw-rw-rw-    1 root     root            0 Nov 25 11:27 access
-rw-rw-rw-    1 root     root            0 Nov 25 11:27 context
-rw-rw-rw-    1 root     root            0 Nov 25 11:27 create
-rw-------    1 root     root            0 Nov 25 14:19 enforce
-rw-------    1 root     root            0 Nov 25 11:27 load
-r--r--r--    1 root     root            0 Nov 25 11:27 policyvers
-rw-rw-rw-    1 root     root            0 Nov 25 11:27 relabel
-rw-rw-rw-    1 root     root            0 Nov 25 11:27 user
```

运行 cat 命令查看 "enforce"文件将会显示一个值，代表 SE Linux 当前的状态，1 代表 enforcing 状态, 0 代表 permissive 状态。

使用了文件系统的扩展属性
新的 SE Linux 使用了文件系统的扩展属性（Extended attributes）来存放安全上下文（security contexts）。你必须让你的内核支持这种扩展属性属性。 扩展属性是一个 名称—数据 元组 （name-data tuple）-- 举个例子说, **security.selinux** 就是一个属性的名称，安全上下文（security context）就是要存的数据。 当 SE Linux 正在运行时，你可以用 *ls --context filename* 命令来查看一个文件的安全上下文(我们将在后面进一步解释这个命令),无论 SE Linux 是否打开，你都可以用 *getfattr* 命令查看文件系统的扩展属性。不过你要先装支持 **attr** 的软件包并且通过 *getfattr* 命令的 manpage 学会使用它。这个命令的运行方法是：

```
faye@kaos:~$ getfattr -m . -d /etc/passwd
getfattr: Removing leading '/' from absolute path names
# file: etc/passwd
security.selinux="system_u:object_r:etc_t\000"
```

你所查看的文件的 **security.selinux** 属性中储存了此文件的安全上下文，所以上面例子中的上下文就是 **system_u:object_r:etc_t** 。所有运行了 SE Linux 的 ext2/3 文件系统上都有 **security.selinux** 这个属性(这个新特性的关键). 如果你引导了一个没有 SE Linux 的内核，你将仍然看到这个扩展属性. 当你用 *make relabel* 操作设置了文件的安全上下文期间，扩展属性就被 *setfiles* 设置了。

从 init 加载 SE Linux 策略

打开了 SE Linux 的系统在引导时，init 进程既要挂载 */selinux* 文件系统，并在那之后载入 SE Linux 的策略。

安全ID（SIDs） 和 父进程安全ID（PSIDs） 不再使用

SIDs（安全ID）在旧的 SE Linux 是用户进程的内核接口. PSIDs（父进程安全 ID SIDs）是内核映射（设置）磁盘上的文件的上下文的根据（译者注：这里的概念可能不是很清晰，总的来说就是SID和PSID在旧的SE Linux中起着标记安全上下文的作用）。 请看NSA的 Configuring the SELinux Policy 获得更多的帮助。 在新的SE Linux中，扩展属性记录了安全上下文，所以SIDs和PSIDs 也就不必要了。

-Z 参数

-Z 可以替代 --context 命令参数，比如 ls -Z 和 ps -Z.

用 chcon 命令替代了 chsid 命令

chsid 命令在旧的 SE Linux 中用来设置文件的安全上下文。新的 SE Linux 中用 *chcon* 命令来设置。 *chcon* 在旧的 SE Linux 中已经可以使用，但是在新的 SE Linux 中的设置用户或类型方面得到了进一步改善。可以查看 manpage 获得更多的提示。

1.4. Fedora 用户的策略（Policy）源代码目录介绍

在 Debian 中，策略的源代码目录是*/etc/selinux*. 在 Fedora 中是*/etc/security/selinux/src/policy*。在这份文档中我市参照 Debian 的源代码目录做的操作，如果你是 Fedora 用户，请用 */etc/security/selinux/src/policy* 替换。

---

2. 总揽

接下来是有关在什么情况下你该使用SE Linux和它的基本使用的简短介绍。2.2部分规定了后面章节将常使用的的术语。，所以请熟悉他们。

2.1 为什么使用SE Linux?

SE Linux 可以为你的系统提供较棒的安全防护。 使用者能被分配预先定义好的角色，以便他们不能存取文件或者访问他们不拥有的程序。 这可不是简单的 ″ chmod 777″ 同等物操作。 这在角色, 或他所在的安全上下文已经限制接触的文件和其他的资源的使用者定义中是不同于一般的 Unix 许可权限的，除了在一种比较受约束的流行之外。 带一个用户的.rhosts 文件在一个一般的 Unix 系统上申请。如果他们使它成为任何人可写入 ，那么任何能登录的人都可以作危险的操作。 在 SE Linux 之下，你能控制其它用户是否有能力改变他们的.rhosts文件, 以及阻止其他的人写入 ，就算拥有者已经使它成为任何人可写入。

一个通常的疑问是 SE Linux 的权限设置如何与标准的 Unix 的权限设置共存。当你做特定的操作的时候， Unix 权限首先被检查。 如果他们允许你的操作 ，那么然后， SE Linux 将会检查并且允许或拒绝使用者的使用。 但是如果 Unix 许可不让你做某事,在那里的运行的操作被禁止和 SE Linux 检查没关系。

另外的一个例子是，如果有一个设置了SUID的可执行文件，如/usr/bin/passwd他可以运行命令 chmod 666 /etc/shadow，SE Linux 会阻止任何人非法的这样设置文件。

2.2 术语

接下来的术语将在本文当中经常出现, 也是来自 SE Linux 的基本概念. It is somewhat tricky to define one word without including the other terms so I realise my definitions include things that need defining（译者注：这句实在不敢乱译，sorry。不过不耽误学习；-)） ;)

2.2.1 （身份）*identity*

在 SE Linux 中，身份的概念不同于传统的 Unix uid (user id). 它们可以共存于一个系统, 但却是十分不同的概念。 在 SE Linux 中的身份是安全上下文的一部分，它会影响哪个域可以进入,也就是本质上的可以被执行。一个 SE Linux 的身份（identity）会跟标准的 Unix 登录名有很相似的文本表示 (大部分情况下它们是这样), 无论如何，了解它们是两个完全不同的概念是很重要的。 运行 *su* 命令不会改变 SE Linux 中的身份（identity）。（译者注：我在红帽系统中做的实验却不是这样，不过这无所谓，可能红帽系统的策略设置不同，我目前还没来得及研究具体是什么问题，只是猜测。）

举例:
一个无特权用户 faye 运行 *id* 命令（在启动 SE Linux 的情况下）可以看到用户的安全上下文：

```
context=faye:user_r:user_t
```

安全上下文中的身份部分就是 "faye"。 现在, 如果 faye su 切换成 root 再运行 *id*, 他将发现安全上下文仍然是：

```
context=faye:user_r:user_t
```

身份保持相同, 跟没切换到 root 时一样。, 不管怎样，如果 faye 身份被允许进入 sysadm_r 角色并转换成了 sysadm_r (这里可以使用 *newrole -r* 命令),再运行 *id* 命令,他将看到：

```
context=faye:sysadm_r:sysadm_t
```

身份字段保持一样但是角色和域(第二和第三字段)的字段已经变了。这样保持身份的方式是用户职责所必需的。身份将影响系统决定哪个角色和域可以被什么身份所使用，这将对系统安全期决定性的作用。

2.2.2 *域*

所有进程都在域中运行。域直接决定了进程的访问。 域基本上是一个进程允许做的操作的列表, 或者说它决定了一个进程可以对哪些类型进行操作。 域就好像一个标准 UNIX 的 uid 的概念。 假设一个属于 root 用户的可执行程序被设置了 setuid。 在这个系统上的任何用户，只要可以执行这个程序，它就有可能获得 root 的权限。这是一个很大的安全漏洞。 再有 SE Linux 的系统上, 如果一个正在执行的进程想要转换进入特权域执行时,如果这个进程的角色被设置成不允许进入特权与的话, 这个进程就不能执行。

常见的例子是 sysadm_t 是系统管理域, user_t 是无特权用户域。 *Init* 运行在 init_t 域, *named* 运行在 named_t 域。

2.2.3 *类型*

类型分配给一个对象并决定谁可以访问这个对象。 它的定义和域基本相同, 不同就是域是对进程的应用而类型是分配给目录，文件，和套接字的。

2.2.4 *角色*

角色决定了那些域可以使用。 有关哪些与可以被哪些角色使用可以预先定义在策略的配置文件里。如果一个策略数据库中定义了一个角色不可以使用一个域, 它将被拒绝。

例子:
如果允许一个属于 user_t 域（无特权用户域）的用户执行 *passwd* 命令，那么必需在相关的策略配置文件中进行如下设置:

```
role user_r types user_passwd_t
```

这样设置了一个属于 user_r 角色的用户允许进入 user_passwd_t 域。也就是说他可以执行 *passwd* 命令。

2.2.5 *安全上下文*

安全上下文包括了所有事情的属性的描述，包括文件, 目录, 进程, TCP sockets 何以上所有的东西。安全上下文包括了身份,角色和域或者类型。在 SE Linux 系统上你可以用 *id* 命令来查看你当前用户的安全上下文。

一间很重要的事情是我们需要明白的是域和类型是有区别的，雨果不明白这一点的话，将使你产生困惑。

域是为进程设置的。当你查肯一个进程的安全上下文的时候（举个例子,你可以查看后面"**转换**"中的解释），最后一个字段的设置，例如 user_passwd_t 就是这个进程的域(如果你运行了 passwd 命令)。

一个像文件，目录，套接字等这样的对象会有一个类型。 当你运行了 *ls --context* 命令时，最后一个字段就是类型的设置，比如 user_home_t 这个类型就是一个有 user_r 角色的用户在他的主目录下建立的文件的类型。

总的来说，域是分配给进程的，而类型是分配给除进程外其他对象的。那么在这里会有一个小小的混淆，就是/proc 文件系统。我们知道/proc 文件系统是虚拟的文件系统。并且里面的以数字命名的目录就是代表了各个进程，数字就是他们的 pid。那么这里如果我们用 *ls -context* 显示/proc 目录下的 1 这个目录，它会显示：

```
dr-xr-xr-x  root     root     system_u:system_r:init_t          1
```

那么这个安全上下文中显示的类型为 init_t。在这里的含义就是 pid 为 1 的这个进程的域也就是 init_t 。（译者注：这样的区分不知道能不能说清楚？）

另一个需要说明的是 *chsid* 命令(改变安全 id) 和 *chcon* 命令(改变安全上下文) 不能在 /proc 文件系统上使用,就是说 /proc 文件系统不支持这种标记的改变。

文件的安全上下文是会根据创建这个文件的进程的域而改变的。默认情况下，一个文件或者目录的安全上下文是从它们父目录那里继承来的,当然我们可以通过策略的改变来改变这种设置。

例子:
faye 用户在他的主目录下建立了一个叫做 test 的文件。 运行 *ls --context test* 可以看到 ：

-rw-r--r-- faye faye faye:object_r:user_home_t test
他又在 /tmp 下建立了一个叫做 tmptest 的文件，再次运行 *ls --context /tmp/tmptest* 这次显示的是：
-rw-r--r-- faye faye faye:object_r:user_tmp_t /tmp/tmptest
第一个例子，安全上下文中的类型是 "user_home_t" 这是一个 user_r 角色的无特权用户默认的主目录设置。 在第二次运行了 *ls --context* 命令后，你可以发现类型变成了 user_tmp_t ，这是由于执行建立文件命令的进程的域是 user_t, 并且在/tmp 下的文件类型要继承 tmp_t 类型。

2.2.6 *转换*

是否发生转换，主要要根据安全上下文来判断。有两种主要的转换。 第一种, 当你执行了一个被限定了类型的程序时会发生进程域的转换。第二种, 在特殊的目录下创建文件时会发生文件类型的转换。

例子:
对于第二种转换 (文件类型的转换), 参照“安全上下文”部分中的例子。当运行了 ls --context 命令之后你会看到文件被标记成了什么样的类型(也就是上面例子中的 user_home_t 和 user_tmp_t)。我们也可以看到当在/tmp 下建立一个文件时，新的文件的类型为 user_tmp_t。

对于进程域的转换, 请参考以下的例子。 以无特权用户的身份运行 ssh, 或者说我们就是运行了一个 user_t 域的进程 (你可以用 id 命令查看你的安全上下文)。运行 *ps ax --context* 查看谁在运行 ssh。 假设是用户 faye, 他将看到：

faye:user_r:user_ssh_t
这是显示的一部分。由于可执行程序的类型是 ssh_exec_t 并且我们用户的角色 user_r 允许访问 user_ssh_t 域，所以 ssh 进程就运行在了 user_ssh_t 域中。

2.2.7 *策略*

策略就是可以设置的规则，决定了例如一个角色的用户可以访问什么; 哪个角色可以进入哪个域 and 哪个域可以访问哪个类型等这样的问题。 你可以根据你想要建立的系统的特点来决定设置什么样的策略。

---

3. 安装

接下来的一章我们讲解释怎样获得软件包以及安装，和怎样获得新的 SE Linux 的软件包以及安装。 因为我运行的是 Debian，所以我演示的安装过程式基于它的。 我们假定你知道该怎样在你所用的发布版上安装软件，编译内核，并且给内核打补丁。

如果你是从旧的 SE Linux 上升级的，并且运行了 SE Linux kernel，请进入 permissive 模式（用 avc_toggle 命令）继续运行指令。

3.1 基于 Debian 的安装

对于 Debian 的开发版（不稳定版）：

将下面的文字写到你的 */etc/apt/sources.list* 文件：

```
deb http://www.coker.com.au/newselinux/ ./
```

这个包是由 Russell Coker维护的.

在写这篇文档的时候（2003 年 11 月末）还没有可以 Debian 上使用的稳定版的新的 SE Linux 安装包。 开发版的 .deb 文件可以从上面的网站获得。请确保获得了最新版本的包。 因为包的名字一直都在变，所以我没有列出，不过他们列出了需要的所有包的名字。

下面我们列出了对于新的 SE Linux，哪些报需要在 Debian 上安装。在安装之前你不需要引导 SE Linux kernel，所以你可以安装它们了：

- **libselinux1**
  包含了新 SE Linux 的共享库。
- **selinux-policy-default**
  包含了范例策略文件，这个策略文件应用到了很多一般应用程序例如 postfix, sendmail, X 等等。
- **checkpolicy**
  包含了安全策略的编译器。
- **policycoreutils**
  包含了核心工具如 setfiles, load_policy, newrole 等等。
- **selinux-utils**
  包含了例如查询策略的操作工具。
- **selinux-doc**
  包括了一些帮助文档 。

Debian 系统所需的附加软件包列表：

- **kernel-patch-2.4-lsm**
  一个支持 LSM 和 SE Linux 的内核补丁。

- **coreutils**
  包含了改进版的命令例如 cp, mv, ls 。
- **procps**
  包含了改进版的 ps and top 命令。
- **sysvinit**
  是一个在引导时加载策略的补丁。
- **dpkg**
  我们需要一个改进版的 dpkg，安装之后可以保证对文件的正确标记。
- **libpam-modules**
  因为一些安全原因。
- **logrotate**
  一个改进版的 logrotate 可以保留一个新建文件的 SE Linux 安全上下文。
- **cron**
  一个改进版的 cron，保证计划任务执行的脚本运行在正确的域内。

3.2 基于 Fedora 的安装

新SE Linux的RPM包可以在这里找到ftp://people.redhat.com/dwalsh/SELinux

这些 RPM 包的维护者是 Dan Walsh.

我在我的 Fedora 测试机上安装 SE Linux 时，我做了这些工作：

* 编辑 yum.conf 文件包含如下内容：

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=fedora-release
tolerant=1
exactarch=1

[development]
name=Fedora Core $releasever - Development Tree
#baseurl=http://download.fedora.redhat.com/pub/fedora/linux/core/deve
lopment/i386
baseurl=http://mirror.dulug.duke.edu/pub/fedora/linux/core/developmen
t/i386

[SELinux]
name=SELinux repository
baseurl=ftp://people.redhat.com/dwalsh/SELinux/Fedora
```

* 运行的命令进行安装
yum install policy checkpolicy policycoreutils policy-sources pam passwd vixie-cron
* 在所有包安装完之后
cd /etc/security/selinux/src/policy
make load
make relabel
*重起机器.

4. 登录

接下来的一部分描述了系统登录，而且解释了更多的关于用户安全上下文一些内容。 本章的最后部分讨论permissive模式和enforcing模式。

4.1在登录时提供用户上下文

在这一个阶段，你应该要重新启动系统并等待那个登录的提示。当你安装了 selinux 的缺省策略包后 (Fedora上是策略的源代码包), 政策文件的安装使你能够以一个缺省用户角色登录系统。(当我们还没有没有添加一个属于我们自己的用户的时候)
以root身份正常登录你的系统。 你的安全上下文默认情况下为 root:user_r:user_t。 *id* 命令显示的类型和你的安全上下文显示应该是相同的，如下所示我们需要看安全上下文部分，所以不必关心其它字段):
uid=0(root) gid=0(root) groups=0(root) context=root:user_r:user_t

所以安全上下文是

root:user_r:user_t
现在我们假设你先前已经把你自己的帐户设置成另外的一个角色。你可以参考 第五章: 建立用户帐户.。对于角色转变有两个方法。 第一是 ，你登录的时候。假设使用者 faye 被认可进入 sysadm_t 域。 使用者 faye 在控制台登录。 在那 "Your default context is faye:user_r:user_t. Do you want to choose a different one? [n]" 这是提示，她选择，y 并按了回车。 她将会见到如下信息:
[1] faye:user_r:user_t
[2] faye:sysadm_r:sysadm_t
Enter number of choice:
在这一个例子中，你能见到那使用者身份 " faye" 先前已经被允许访问sysadm_r 角色和 sysadm_t 域。 这里将会被显示的选项是那些你的使用者身份已经被允许访问的对象。 请注意，这在旧的 SE Linux 已经实现了，而且将会在新的 SE Linux(在写这文档的时候是不可以的) 中被设置为可配置选项，默认的设置为关闭（OFF）。

如果用户 faye 选择了选项二（ 变成 sysadm_r）然后运行 id 命令，她将会见到安全上下文的内容为：

context=faye:sysadm_r:sysadm_t
意味着他现在是 sysadm_r 角色。

接下来是第二个改变用户安全上下文的方法。

4.2 用 newrole -r 命令改变上下文

变更你的安全上下文的第二个方法将使用 newrole-r 的指令。语法是

newrole -r role
这里的 role 替换你想要转换成什么角色。假设是 sysadm_r。那么既可以运行：
newrole -r sysadm_r
你将会被要求为你的使用者身份提供密码，你可以运行id指令检查。 如果你没有授权进入一个新的角色，你将会见到这样的显示(假设使用者fred 尝试运行的指令)
fred:sysadm_r:sysadm_t is not a valid context
这一个信息意味着 fred 用户不能进入 sysadm_r:sysadm_t 角色:域，因为他没有被授权可以这么做。

在成功地变更角色之后，运行 id 指令检查你的安全上下文。

4.3 在 sysadm_t 域中执行命令

你的用户现在已经在 sysadm_r 角色，运行的程序是在 sysadm_t 域。此时我们需要把我们的安装做的稍微完善一些，因此让我们去虚拟的控制台并用root用户登录。 你将不被要求是否想要改变上下文。

我必须在这里说明一些事情。 我们实际上到目前为止没有在这份HOWTO 中说明怎样让root用户允许访问 sysadm_r 角色，所以你的思路可能还停在那里,root用户只允许访问user_r:user_t，这样我们怎么进行系统管理? 好吧，我们正在运行的模式是permissive，这是一种并不真正强制的执行安全策略的设置的模式。你依旧能在上面使用 newrole-r 指令换成 sysadm_r 角色。运行newrole 指令是转换的方法。 如果你试着做你不被允许的事情,你将看到一屏接一屏的错误显示信息，这并不好玩。

所以，转换成 sysadm_r 角色并且运行 id 来检查你实际的上下文是不是 sysadm_r:sysadm_t。

现在我们能在 sysadm_r 角色中得到一些乐趣。当我们在第 3 节中安装了所有东西的时候，当时系统上所有的文件都被标记了一个类型,但是电脑却没有在运行 SE Linux 。 因此如果一个文件在将程序分类发生之后建立的话，并且在系统没有重新启动 SE Linux 的内核之前，那么那一个文件将不属于任何一种类型。想象一些文件可能在关机期间被创建。 这些文件都没有标记类型。 于是，考虑到这一个情况。 如果你删除一个文件,那么那个文件的 inode 号可能被用来标

记一个新的文件，而且这个新的文件可能是删除的那个文件的类型。 这是一个严重的问题。

关于/etc/nologin 文件。 当shutdown指令被执行的时候，这一个文件产生。 如果这一个文件在引导的时候存在,只有root将会被允许登录。 如果你的启动脚本不能删除这个文件， 而且/etc/nologin 有错误的标记, 启动脚本就不能touch it，于是就会产生一些小问题。 如果你的root身份配置成在登录之后有一个 sysadm_r 的缺省角色, 那么你就能登录并且删除这一个文件,问题解决。

但是如果你已经配置你的root身份在登录之后不能得到 sysadm_r 角色怎么办? 在这样的情况的下，你的root身份的上下文可能是root:user_r:user_t。但是 user_t 域不允许你删除任何在 /etc目录下的文件。于是问题出现了，你能用root身份登录, 但是做不了sysadm_r 角色的特权允许做的任何事。

再次想象, 这样一种情况，你有你自己的用户身份,再一次让我们使用 " faye" 身份。 身份 faye 配置成一登录就会变成 sysadm_r角色。因此身份 faye 能执行所有的 sysadm_r 角色的事情,但是root身份 (以user_r 角色在 user_t 域中运行) 不能。 faye 用户可能有很高的权限，但是faye 身份却不行, 因为由于事实它将由于 /etc/nologin 文件的存在而不能够登录，此文件不让非root用户登录。

这就是为什么正确地将文件分类是至关重要的。 让我们回到进程被标记之后文件已经建立，但是还没引导SE Linux内核的情况。 为了修复这个问题, 我们必须运行

```
make -C /etc/selinux relabel
```

这一个命令将会确定在你的系统上的所有的文件正确地被标记。 执行的速度和你的机器上有多少文件有关,这可能需要一会儿。 一个粗糙的估计, 它将会像一个‘find /’指令一样的久。 这是为什么你想要使用 newrole 指令换成 sysadm_r 然后运行指令上述make command 命令——如果你在一个不能访问其他域的域中 (例如user_t), 你将会收到数以万计的 ″permission denied″ 提示。

4.4 Permissive 模式 和 Enforcing 模式

Permissive模式是指，你的 SE Linux 机器在本质上并没进入SE Linux 只显示相关信息的状态, 没有其它什么了。所以你仍然可以用root用户做相同的操作就像你在一部非 SE Linux 机器上一样。 Enforcing模式强制使你的所有安全策略生效。 就是说在这种模式下，你配置的所有SE Linux的策略已经生效。所以，你可以用Permissive模式来检查你的策略配置是否合乎要求。(通过检查 dmesg 信息)

这里需要强调的是:在启动到enforcing模式之前请确定你做了合适的策略配置。所以你可以在Permissive模式中运行以下来检查。Permissive模式标记了文件, 但是不实际上运行任何事, 除非所有事情都被确认。 一些人编译了一个没有 CONFIG_SECURITY_SELINUX_DEVELOP 支持的内核，那意味着你不能运行

Permissive模式。

在 permissive 模式和 enforcing 模式间转变， 你需要运行 echo ″1″ > /etc/selinux/enforce 以打开 enforcing 模式。将 1 替换成 0 则意味着运行在 permissive 模式。 旧的 SE Linux 用了在新的 SE Linux 中已经不用的 avc_toggle 指令。 用 cat /etc/selinux/enforce 命令可以知道你正在哪种模式下运行。

你可以查看 ″第七章：日志文件信息的说明″，找到有关转变模式的信息提示的例子。

如果你编译的内核使用了 development 模式（意味着你的机器运行在了 permissive 模式，并且还没有设置成 enforcing 模式)，你可以写一个启动脚本来转换成 enforcing 模式，或者在启动 bootloader 的时候将内核参数设置为 *enforcing=1* 。(编辑你的 lilo.conf 文件，添加 *append=″enforcing=1″*).

4.5 不同角色运行命令的比较

我们现在会在不同的安全上下文的环境下运行一些命令。 转换到 enforcing 模式。在 user_r 角色的环境下，运行 *ps ax --context* 命令并观察输出信息。 别忘了 *ps ax -Z* 命令可以做同样的事情。 在角色为 user_r 的时候，你可以看到那些被允许访问/proc 目录的而且运行在 user_t 域中的进程。 如果哪个进程不能访问/proc 目录，那么哪个进程不会显示在 *ps ax* 命令的输出中。

现在转换到sysadm_t 域中,运行 *ps ax --context* 命令。这次，你将会见到在系统里的所有的进程，不管他们是在哪个域下运行。 当在 sysadm_t 域的时候，你可以访问到user_t域不可以访问到的其它域的进程。 这就是问什么在 user_t 域中你不可以系统上所有的进程。想像一个恶意用户能够见到所有的系统程序。她能看到一个有安全漏洞的 daemon 在运行，于是她就可以针对这个漏洞进行攻击。 如果 user_t 域不能见到 daemon进程，那么这样的危险会被减少。

另外一个要考虑的问题就是命令行上的密码问题。 默认的linux设置，是可以让任何人读到这样的信息的。 当 SE Linux 阻止你看见ps 输出的一个程序的信息时，它将减少这样的危险。（当然，一个密码在命令行上显示，是一个差劲的主意)。

转换回 permissive 模式。 你将又会在 user_t 域下用 *ps ax* 命令看到所有的系统进程。

5. 建立用户帐户

现在来作点有意思的事情! 我们将会建立一个 SE Linux 用户并分配给他一个角色，然后为用户设定默认的安全上下文。 在旧的SE Linux环境下, 封装程序的建立用 vipw (svipw)来设置，比如， useradd (suseradd), passwd (spasswd), chfn (schfn)等， 在新的 SE Linux环境下, 这些程序有其它的名称。

5.1 建立一个新的用户

我们现在建立一个新用户。我们叫它 setest。

转换到 sysadm_r:sysadm_t 角色:用户。 现在用 *useradd* 命令添加用户 setest:

```
root@kaos:~# id
uid=0(root) gid=0(root) groups=0(root) context=faye:sysadm_r:sysadm_t
sid=398
```

运行 id 命令检查确认你的 uid 是 0 并且你的身份是在 sysadm_r:sysadm_t 角色:域中。 如果你的 uid 是你其它似有用户的, 请先用 *su* 命令转换乘 root 身份, 然后运行 *newrole -r* 命令。

```
root@kaos:~# useradd -c "SE Linux test user" -m -d /home/setest -g users
-s /bin/bash -u 1005 setest
root@kaos:~# finger setest
Login: setest                           Name: SE Linux test user
Directory: /home/setest                 Shell: /bin/bash
Never logged in.
No mail.
No Plan.
root@kaos:~# passwd setest
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

setest 用户现在已经添加完了。

5.2 给用户分配角色和申请改变

现在我们想给 setest 用户设置一个角色。 我们希望他可以访问 user_r 角色。需要配置的文件是*/etc/selinux/users* ，你现在可以用你最喜欢的编辑器打开他，并先浏览一遍。

在文件的结尾添加如下内容:

```
user setest roles { user_r };
```

这行的意思是允许 setest 用户进入 user_r 角色。 如果你还希望 setest 用户还可以访问 sysadm_r 角色, 你可以添加：

```
user setest roles { user_r sysadm_r };
```
我们现在要让我们的设置生效，所以我们可以在 sysadm_r:sysadm_t 角色:域的情况下运行以下命令:
```
make -C /etc/selinux load
```
这将花费一段时间，这段时间里会创建策略的数据文件并用 *gzip* 压缩。如果命令成功执行并退出, 你将看到以下提示:
```
Success
touch tmp/load
make: Leaving directory `/usr/share/selinux/policy/current'
```
在默认的角色 user_r 下的用户是不允许向 */etc/selinux/users* 文件添加内容的。如果你想要他们可以使用 user_r 之外的一个用户角色或者让他们能够改变他们自己的密码,就要他们加入这一个文件，或在 SE Linux 的记录信息的适当部分添加他们的用户名。

现在我们来设置一个缺省的安全上下文。

5.3 给用户设置缺省的安全上下文

在向 */etc/selinux/users* 文件添加完新用户之后， 缺省的安全上下文必须要再登陆的时候被指定。配置文件是*/etc/security/default_context* 。你将看到如下信息:

```
system_r:local_login_t  user_r:user_t
```
当一个用户从本地登陆的时候 (或者说从控制台登陆), */bin/login* 程序会在local_login_t 域中运行并分别地分配一个用户角色是 user_r 和域为 user_t 。
如果显示的是：
```
system_r:local_login_t  sysadm_r:sysadm_t user_r:user_t
```
那么用户登录时允许进入 sysadm_t 域, 那么他就将以 sysadm_t 域的身份登录进来。 如果不允许, 就会使用 user_t 域。

请看这一行：

```
system_r:sshd_t          user_r:user_t
```
这意味着所有铜壶 ssh 登录的用户将使用 user_r:user_t 角色:域。

5.4 重新标记用户主目录

如果你已经用 useradd 添加了一个角色为 user_r 的新用户, 那么那么你需要仔细的改变它已有的标记。 如果用户角色不是 user_r, 那么你就不能重新标记，于是你必须运行以下命令:

```
find /home/setest -print0 | xargs -0 chcon -h
system_u:object_r:user_home_t ;\
chcon -h system_u:object_r:user_home_dir_t /home/setest
```

这一个命令使所有在/home/setest目录下的文件都运行了 chcon命令(变化文件安全上下文)改变了文件的安全上下文。 用户主目录被标记成类型为 user_home_dir_t ，而且在户主目录下的文件都被标记成类型为 user_home_t 。 有时，一个程序可能被允许访问一个用户主目录，但是不能访问何再此目录下的文件，两种不同的类型由此而来。

6. 添加新的用户域

现在让我们建立一个我们自己的用户域，并把它叫做second_t 。 我们也将会建立一个新的角色叫做second_r 。要建立second_r 角色首先在前面的部分(刚刚分配了 user_r 角色并且并不真正建立这样的用户) 中一步一步跟着作，但是不要运行第 5.2 节中的make命令。在你已经编辑 /etc/selinux/user之后，回到这里并且继续下一部分，关于编辑使用者领域文件的部分。

关于为什么我不想要你运行make指令的原因，是因为早先的部分刚刚分配了缺省的一个 user_r 的角色。 但是我们将会建立一个新的角色，而且同样地我们需要一个新的域与它搭配。以下部分进行概略说明。

6.1 编辑用户的域文件

用户域的配置文件是 */etc/selinux/domains/user.te* 。请先看一遍。 添加以下几行：

```
full_user_role(second)
allow system_r second_r
allow sysadm_r second_r
```

在文件的什么位置添加是无所谓的，在上面添加注释:

```
# if adding new user roles make sure you edit the in_user_role macro in
# macros/user_macros.te to match
```

再来编辑 */etc/selinux/macros/user_macros.te* 文件进行匹配。 打开此文件并找到 in_user_role (差不多在文件的结尾)这行。添加 "role second_r types $1;" 现在这部分文件应该像这样：

```
undefine(`in_user_role')
define(`in_user_role', `
role user_r types $1;
role second_r types $1;
')
```

回到我们编辑的第一个文件 (full_user_role(second))，这样我们就建立了 second_t 域和 second_home_dir_t 类型以及 second_home_t 类型(用户主目录的类型和主目录里文件的类型)。一个 second_tmp_t 类型 ，当在/tmp 目录下建立文件时。 类型 second_tmpfs_t 是在 tmpfs 中共享内存文件系统中建立文件的上下文。 最后，second_tty_device_t 和 second_devpts_t 类型分别被用来

标记终端设备（tty）和虚拟终端设备。当然也建立了相应的策略应用于这些标记。

SE Linux 内部并不支持任何类型的标定，以及类型/域的继承，等。当然策略语言的编写也不支持这些特征。 所以我们用 M4 宏编译器来设置简单的域和类型。

我们现在来建立一个用户在这个新域中使用 (second_t)并访问 second_r 角色。

6.2 再次建立一个新的测试用户

使用 useradd，建立一个新的用户。(让我们假设用户叫做 ″spike″) 把 spike 加入 /etc/selinux/users 只给予他对 second_r 角色的访问权和没有其它权限。 然后运行

```
make -C /etc/selinux load
```

应用新的策略。

接下来是设置缺省域和新的角色。 我们可以编辑 */etc/security/default_type* 文件并添加以下行：

```
second_r:second_t
```

我们现在必须手动设定/home/spike 和它的上下文。*useradd* ，命令不会做这些事情,它只支持重新标记用户的 user_r 角色。 运行下面的命令:

```
find /home/spike -print0 | xargs -0 chcon -h
system_u:object_r:second_home_t ;\
chcon -h system_u:object_r:second_home_dir_t /home/spike
```

现在试着用 spike 用户身份登录。

---

7. 日志信息的说明

接下来介绍提示的说明信息。我将解释每部分信息的意义。 对于比较容易的部分，我会直接在输出的内容上标记出来。

有些时候,日志信息不会以你喜欢的形式清晰的表达出来，所以我们首先要清除 ReiserFS 和 Ext2/Ext3 文件系统(SE Linux 支持的文件系统) 的 root inode 是 2。

XFS 文件系统和 JFS 文件系统现在正在测试中。

例 1

avc: denied { getattr } for pid=6011 exe=/usr/bin/vim path=/etc/shadow dev=03:03 ino=123456 \

scontext=faye:user_r:user_t tcontext=system_u:object_r:shadow_t tclass=file

这一个例子显示的是,在enforcing模式时一个无特权用户(faye) 尝试编辑 /etc/shadow文件的显示信息。

"avc: denied" 意味着这样的操作被拒绝。
"{ getattr }" 意味着有人对文件使用 stat()函数。 在这个操作中，必须首先查看文件的属性 (或者说至少要查看文件属性)，如果不能查看，于是停止操作。

braces {}里的内容包含了操作的动作，或者说是 SE Linux 正在做的有关操作。SE Linux 可以做出包括 allow 和 deny 两种检查结果，在这个例子中检查的结果是被拒绝，并通知你你想要的操作被拒绝。

"for pid=" 你操作的进程的 pid。
"exe=/usr/bin/vim" 是你执行的命令 (在这个例子里，执行的是 vim)。
"path=/etc/shadow" 实行要操作的目标文件路径。
"dev=03:03" 是跟此操作有关的文件系统所在的块设备的设备号。第一个 "03" （主设备号）意思是使用了 hda ，第二个 "03" 是 3(辅助设备号或从设备号)，所以这个"dev=03:03" 就意味着 /dev/hda3 (如果你在 devfs 上运行那么应该是/dev/ide/host0/bus0/target0/lun0/part3). 当 SE Linux 核查权限的时候他并不知道你要操作的对象的完整路径所以它不能记录任何事情。它只知道文件系统的相关路径，和文件系统所在块设备的设备号。 比如说我们要访问的 /etc/shadow. SE Linux 并不知道这个文件在根目录下。它仅仅知道 /etc/shadow 所在的文件系统。
"ino=123456" 目标的 inode 号 (这个例子中的目标是 /etc/shadow)
"scontext=faye:user_r:user_t" 操作动作来源进程的上下文
"tcontext=system_u:object_r:shadow_t" 操作对象的安全上下文 (/etc/shadow)。
"tclass=file" 意味着目标对象是个文件。

Example 2

avc: granted { avc_toggle } for pid=6073 exe=/sbin/avc_toggle \

scontext=faye:sysadm_r:sysadm_t tcontext=system_u:system_r:kernel_t tclass=system

"avc: granted"意思是你的操作被允许正常运行。
"{ avc_toggle }" 表示你的程序调用了 avc_toggle()系统调用。
"tclass=system" 表示目标程序属于 system class。

例 3

```
avc:  denied  { append } for  pid=6153 exe=/bin/bash
path=/.bash_history dev=03:03 ino=498 \

scontext=faye:user_r:user_t tcontext=faye:object_r:root_t tclass=file
```

这条提示是说属于 user_r:user_t 角色:域的 faye 身份的用户想要在属于 root 的类型为 root_t .*bash_history* 文件里添加内容被拒绝。

例 4

```
avc:  denied  { write } for  pid=605 exe=/bin/touch dev=09:03 ino=2 \
scontext=root:user_r:user_t tcontext=system_u:object_r:root_t
tclass=dir
```

这个例子显示的是路径找不到。然而我们可以知道的是因为 inode 号是 2，所以是根目录。

---

8. 资源

在下面是和 SE Linux 的相关的链接材料。 请参照 NSA 的白皮书。

NSA 官方网站

NSA的官方SE Linux 网站： http://www.nsa.gov/selinux

官方SE Linux FAQ在： http://www.nsa.gov/selinux/info/faq.cfm

NSA 出版的文件，技术上的报告在 http://www.nsa.gov/selinux/info/docs.cfm

邮件列表

NSA 为所有有关 SE Linux 的讨论建立了一个邮件列表
http://www.nsa.gov/selinux/list.html ，相同的网址上也说明了该如何获得目录文件。

SE Linux 的 Sourceforge 项目

SE Linux的Sourceforge 在： http:/sourceforge.net/projects/selinux.

Fedora Core 2 SE Linux FAQ

Karsten Wade 已经为在Fedora Core 2 上使用SE Linux的用户写了一份 FAQ ，可以在下面地

址上找到：http://people.redhat.com/kwade/fedora-docs/selinux-faq-en/