SELinux 简明学习指南

Hyphen Wang

www.linuxfly.org

2012-07-06

目录

1. 介绍

SELinux存在系统中已经有不短的历史了，但由于各种原因吧，经常会被人们有意无意的忽略或禁用。老实说，在过去，我也是其中之一。主要是感觉配置实在太麻烦了，受限太多（涉及安全的东西都这样，SELinux不能例外）。但这次因为某项目的测试需要，只能死马当活马医，努力学习吧。

国内相关的资料不多，而且大多是一些简单的命令介绍，最详细的是harrytaurus2002 写的SELinux学习笔记，2012-02-14 版本有 444 页，够多的。可能内容太多，或关注于开发的角度吧，反正刚开始我是没看懂。（不好意思啊，劳烦作者的一番苦心，^_^）好吧，找国外的资料，《SELinux by Example Using Security Enhanced Linux》一书应该是最经典的教材，当然Fedora SELinux Project 、Tresys Technology公司的SELinux Reference Policy 、Dan Walsh's Blog等网站也是很好的参考资料。

回过头来，我下面从实用性的角度出发，整理一下这几周学习SELinux的内容，以便后来者继续学习和研究。当然，我写的东西肯定没有SELinux学习笔记的详尽，可理解为就是一个精简版。文章会分开好几部分，可通过SELinux TAG查询。而且受限于学习水平，语句比较通俗，文中问题肯定不少，也希望大家提供反馈和意见。在最后，若允许的话，我会把各部分内容整理为单份文档格式，便于传递和阅读。

因这次的学习是以项目主导的，运行平台为红旗安全操作系统 4.0（RFSOS 4.0），所以，文中部分内容与其他发行版会略有不同。但RFSOS中安全策略是基于标准的referrence policy的，与RHEL还是比较雷同的，可以参考和比较学习。

一、SELinux简介

SELinux 是由美国国家安全局 (NSA) 开发的，为了防止内部资源管理设置错误引发的问题，或受攻击入侵时，把可访问内容控制在最小的范围之内。例如，为了方便，你把HTTP服务的/var/www/html目录设置为 777 权限。但万一由于一些漏洞问题，黑客可在该目录下写入执行文件，那么问题是很严重的（暂不考虑chroot设定的情况）。此外，时不时爆出的本地提权漏洞（root exploit）问题，对系统的安全也有很大的威胁。

为了控管这方面的权限与程序的问题，所以美国国家安全局就著手处理操作系统这方面的控管。他们在SELinux整合到核心中，用于在进行程序、文件、网络服务等访问时增加一道关卡，而且这道坎使得原来的管理员root不再是万能的，应用程序的配置也不再是任意的，而是受限的。

1.DAC（Discretionary Access Control）自主式存取控制

这简单来说，就是类Unix系统的默认安全管理权限，即rwxrwxrwx问题。在DAC模式下，依据程序与文件资源的rwx权限来决定是否有可访问、写入、执行的能力。例如：

引用

root 具有最高权限：uid/euid 和 gid/egid 的存在，使得程序存在安全漏洞时，将很容易获得 root 的权利，这样，他就是万能的主；

使用者本身可改变文件存取权限：假设用户对系统不熟悉，给文件赋予 777 的设置时，结果可想而知的。

这些都是问题，怎么解决？那就采用 MAC 模式。（不是 ext 文件系统的 MAC 扩展）

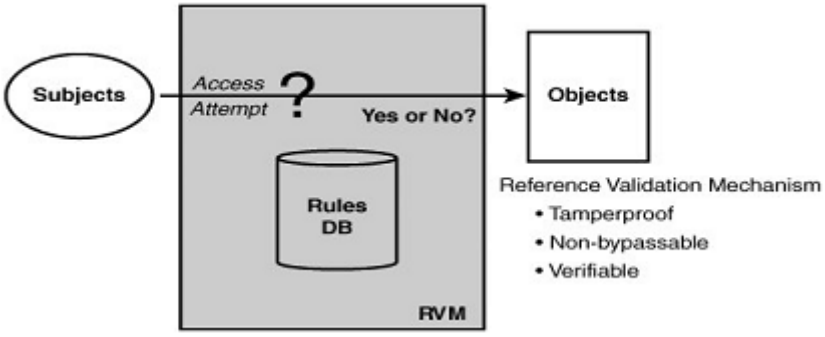
2.MAC（Mandatory Access Control）委任式存取控制

在该模式下，不再由访问对象的属主定义不同用户对其的许可访问方式，而是由固定的规则库决定。在规则库中，定义了主体对客体的访问控制规则，默认是禁止的，也就是说，没有允许你就不能读写，即使你是 root 也不行。

在此环境下，万一黑客通过 httpd 得到/var/www/html 目录的读写、执行权限，甚至是 root 身份，他也无法对其他非允许的文件进行访问，因为他们的类型不同，规则不允许。那么，系统受到的影响将降到最低。

那么，DAC 与 MAC 有冲突吗？这是没有的，可以把 MAC 理解为 DAC 的下一层，换句话说。在激活 SELinux 的环境下，访问一个文件资源（类 Unix 系统中，一切都是文件），**你将需要先获得 DAC 的许可，然后再获得 MAC 的许可，那么你才能访问该文件资源。**

3.访问控制模型



由四部分组成：

引用

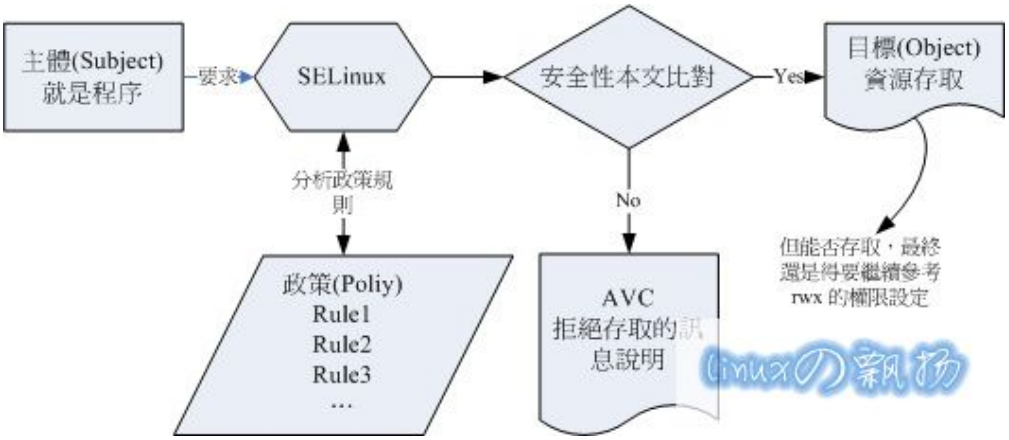
1）Subject：主体，比如系统中的进程、执行者；

2）Object：客体，被访问的对象，如系统中的资源、数据，其实就是文件；

3）Rules DB：规则库，定义了 Subject 可以通过什么方式对 Object 进行访问；

4）RVM（Reference Validation Mechanism）：操作系统内部的机制，是 Rules 的执行者，判断当前操作是否合法。

SELinux是作为一个安全模块嵌入到Linux Kernel 中的，由policy rules（策略规则）驱动。首先它需要标识主体和客体的**类型**，并把它们和判断规则等都会放入Access Vector Cache（AVC）中。AVC是一个缓存数据库，可以提高性能。在操作系统安装完毕的最后一步，进行label （标签）后，磁盘上各文件（客体）就会拥有各自的context（安全上下文），主体对客体的操作，将由AVC判断后执行。



（图片来自鸟哥的私房菜：程序管理与 SELinux 初探）

4.优势

采用SELinux有以下几点好处：

（这部分内容来自Red Hat Enterprise Linux 6 Security-Enhanced Linux User Guide Edition 2）

引用

● All processes and files are labeled with a type. A type defines a domain for processes, and a type for files. Processes are separated from each other by running in their own domains, and SELinux policy rules define how processes interact with files, as well as how processes interact with each other. Access is only allowed if an SELinux policy rule exists that specifically allows it.

● Fine-grained access control. Stepping beyond traditional UNIX® permissions that are controlled at user discretion and based on Linux user and group IDs, SELinux access decisions are based on all available information, such as an SELinux user, role, type, and, optionally, a level.

● SELinux policy is administratively-defined, enforced system-wide, and is not set at user discretion.

● Reduced vulnerability to privilege escalation attacks. One example: since processes run in domains, and are therefore separated from each other, and because SELinux policy rules define how processes

access files and other processes, if a process is compromised, the attacker only has access to the normal functions of that process, and to files the process has been configured to have access to. For example, if the Apache HTTP Server is compromised, an attacker can not use that process to read files in user home directories, unless a specific SELinux policy rule was added or configured to allow such access.

● SELinux can be used to enforce data confidentiality and integrity, as well as protecting processes from untrusted inputs.

但 SELinux 不是：

引用

防病毒软件

不能代替密码、防火墙或其他安装系统；

不是一个整体化的安全解决方案。

SELinux 是用来加强已有的安全架构，而并不是替代它们。所以 RFSOS 不单采用 SELinux，还包括加密文件系统、验证密钥、模块签名等一系列的安全措施。

2. 运行模式

既然 SELinux 有不少好处，但为啥经常被禁用呢？主要是因为其配置太复杂，而且涉及的资源、权限很多，这也是安全配置的通病。不过，若对 SELinux 有一定了解，适当的配置使用，还是对系统安全很有好处的，特别是针对一些关键应用。通常系统运行的 SELinux 有三种模式：disabled、permissive、enforcing。

二、运行模式

1.说明

SELinux 的三种运行模式分别是：

引用

enforcing（启用）：启用安全保护如果有安全违规行为发生，，将生成一个审计日志并阻止该非法行为；

permissive （报警）：不启用安全保护，如果有安全违规行为发生，将生成一个审计日志，但不阻止违规行为；

disabled （不启用）：不启用 SELinux 安全模块功能。

启动 SELinux 时，应运行在 enforcing 模式下，而调试策略时，可切换到 permissive 模式。disabled 就是不用 SELinux 安全模型。

此外，常见有两种 SEinux 安全策略：

引用

targeted ：仅保护一些系统服务，一般不使用该策略；

mls：提供基于 BLP、MLS/MCS 模型的完整保护；

RHEL 系列相同默认采用 targeted 策略，而红旗安全操作系统 4.0 的默认生效策略为 mls。它们之间的区别是：

引用

mls 策略是红旗安全操作系统默认安全策略，为系统及应用提供最大化的安全保护。

targeted 为系统提供的可选安全策略，其目标是隔离高风险程序（如暴露在外易受黑客攻击的组件）。使用 targeted 策略的好处是一方面可以向 Linux 系统添加大量的安全保护，同时又尽量少影响现有的用户程序（提高应用软件与系统兼容性）。

在红旗安全操作系统中，targeted 策略和 mls 策略之间主要差异是不支持管理特权三权分立，也不支持 BLP 模型。（后续会进一步说明）

2.配置文件

模式及策略在/etc/sysconfig/selinux 中设置：

引用

```
# cat /etc/sysconfig/selinux
# 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 - SELinux is fully disabled.
SELINUX=enforcing
#SELINUX=permissive
# SELINUXTYPE= type of policy in use. Possible values are:
```

```
#       targeted - Only targeted network daemons are protected.
#       mls - Multi Level Security protection.
SELINUXTYPE=mls
#SELINUXTYPE=targeted
```

由于 RFSOS4 默认采用 mls 策略，初始会创建 sysadm、secadm、auditadm 三个权限不同的用户，而对于安全配置的修改，必须使用 secadm 用户进行，其他用户对该配置文件都是“只读”的。

类似的，RFSOS4 在本地终端和 SSH 服务禁止 root 用户登录。若要修改上述配置文件，请以 secadm 用户登录后，执行以下操作：

引用

```
login as: secadm
secadm@192.168.228.174's password: ***************
* Password Status:
   Password won't be expired in recent 90 days.
* Success List
User              Login Time              From
secadm  Jun 14 11:39:50 +0800 2012      192.168.228.221
secadm  Jun 14 11:36:42 +0800 2012      192.168.228.221
secadm  Jun 14 11:06:54 +0800 2012      tty1
secadm  Jun 14 09:47:56 +0800 2012      192.168.228.221
Last login: Fri Jun 15 10:40:37 2012 from 192.168.228.221
[secadm@localhost ~]$ id
uid=513(secadm) gid=513(secadm) groups=513(secadm)
context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023
[secadm@localhost ~]$ su -
口令：
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023
```

登录 secadm 时，请输入 secadm 用户的密码。执行 su - 命令时，需输入 root 的密码。蓝色标记的称为安

全上下文，将在后续说明。

※ 请注意，本节的命令操作都以 secadm 用户的上述登陆状态进行。

3.查看状态

执行：

引用

```
[root@localhost ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 21
Policy from config file:        mls
[root@localhost ~]# getenforce
Enforcing
```

4.修改模式

enforcing 和 permissive 模式的切换，可通过 setenforce 命令进行：

引用

```
[root@localhost ~]# setenforce 0
[root@localhost ~]# getenforce
Permissive
```

或直接对 SELinux 文件系统的文件进行修改：

引用

```
[root@localhost ~]# cat /selinux/enforce
0
[root@localhost ~]# setenforce 1
[root@localhost ~]# cat /selinux/enforce
1
[root@localhost ~]# getenforce
Enforcing
```

可见，1 表示 enforing 模式，0 表示 permissive 模式。

而把配置文件修改为 disabled 模式，可禁用 SELinux，但必须重启才能生效。

安全策略的修改只能通过修改配置文件进行，而且是立刻生效的：

引用

[root@localhost ~]# vi /etc/sysconfig/selinux

[root@localhost ~]# sestatus

SELinux status: enabled

SELinuxfs mount: /selinux

Current mode: enforcing

Mode from config file: enforcing

Policy version: 21

Policy from config file: **targeted**

但由于两种策略所标记的标签不同，AVC 中记录的数据不一致，实时的修改会产生异常的问题。因此，修改安全策略后，应该重启服务器，并重打标签。

5.临时设定模式

若只是由于某些原因，需要临时关闭 SELinux，可在 Grub 引导菜单中，通过 e 按钮，切换到 edit 状态后，在 kernel 一行，如：

引用

kernel /boot/vmlinuz-2.6.18-128.7AXS3 ro root=LABEL=/

的后面加入 selinux=0，即：

引用

kernel /boot/vmlinuz-2.6.18-128.7AXS3 ro root=LABEL=/ **selinux=0**

修改完毕后，用 esc 键退出，b 键启动进入系统。

当然，如果你修改/boot/grub/menu.lst 中的配置，那结果与修改配置文件是相同的，都可把 SELinux 设为 disabled 模式，禁用了。

类似的，把 enforcing=0 加入 kernel 行，如：

引用

kernel /boot/vmlinuz-2.6.18-128.7AXS3 ro root=LABEL=/ **enforcing=0**

就是设置为 permissive 模式。

如果 SELinux 为 disabled 状态，将不会把 selinuxfs 文件系统挂接到/selinux 下面，也就是不存在该目录。

6.重打标签

在操作系统安装完毕时，系统会自动对磁盘中的文件根据 fcontext 规则打上标签，以便 AVC 进行判断。在 SELinux 处于 enforcing 或 permissive 模式下，对文件标签的修改都会被记录到磁盘上。但如果在 disabled 模式下，这动作将不会进行。所以，如果在 disabled 模式下，对文件进行了改动，重启进入激活 SELinux 状态的系统时，就会报 avc 错误。

类似的，targted 和 mls 安全规则所使用的标签也是不相同的。在两个安全规则间切换时，也必须重打标签。

重打标签的通常做法是，通过在 Grub 菜单临时进入 permissive 模式（见上面的描述），以 root 执行：

touch /.autorelabel

然后重启系统，还是进入permissive模式，系统即会自行对根分区所有文件重打标签。完成后，可恢复到enforcing模式状态。更详细的信息，请见[原]解决启动时提示Entering runlevel 及大量avc提示的问题。

3. 安全上下文

正如上文所说的，SELinux 会对进程和文件（实际上，进程也是文件之一）进行标记（label），包括 SELinux user、role、level 等。这些成为安全上下文（Security Context）。SELinux 使用 Role-Based Access （RBAC）、Type Enforcement（TE）和 Multi-Level Security （MLS）等模型。

三、安全上下文

1.TE 模型

SELinux 作为 MAC 的一种实现，通过中央规则库（policy.X）给所有进程、文件、内核数据结构定义各自的安全标识（label）或标签（type），明确定义被访问对象所支持的访问方式，并规定进程标签（主体）对被访问对象（客体）的合法访问方式。

对文件系统的标签在操作系统安装完毕后标注（或手动设置重标，见上篇），而在系统启动过程中 init 进程经由 selinuxfs 接口装载 policy.X 到内核空间，由内核中的 Security Server（如 policydb、AVC 等）在处理用户态系统调用时实时查询。

policy.X 的 X 是版本号，在编译时决定，其可以由几百个子模块链接而成，这些子模块称为 policy package（简称 pp）。通常位于系统的以下位置：

引用

/etc/selinux/mls/policy/policy.21

/etc/selinux/targeted/policy/policy.21

根据所选择的安全策略不同而加载。以 RFSOS4 运行在 mls 下为例，实时可用的 pp 模块位于：

引用

/etc/selinux/mls/modules/active/modules/

基础模块是 base.pp，其他模块都是后来链接上去的。

2.磁盘上的安全上下文

磁盘上的文件，其安全上下文（Security Context）是存放在文件的扩展属性（Extended Attribute，xattr）中的。但并不是所有的文件系统都支持该属性，ext3/4 可以，Samba 的 cifs 和 NFS 就不行，不行的需采用其他的办法弥补（后面会讲到）。

先来看看磁盘上的文件：

引用

```
# ll -Z install.log
-rw-r--r--  root root root:object_r:sysadm_home_t:s0    install.log
```

关注第四列（必须加-Z 参数才能看到），通过冒号区分为四部分：

user:role:type:level

1）user 部分

这不是普通的系统用户，而是 SELinux User，由 login 程序根据当前规则库的定义，把系统用户映射到 SELinux User 上的。RFSOS4 上就是这样：

引用

semanage login -l

Login Name SELinux User MLS/MCS Range

__default__ user_u s0

auditadm auditadm_u s0-s15:c0.c1023

root root s0-s15:c0.c1023

secadm secadm_u s0-s15:c0.c1023

sysadm sysadm_u s0-s15:c0.c1023

system_u system_u s0-s15:c0.c1023

可以看到，之前我们提到的登陆用户 secadm，在 SELinux User 下就是 secadm_u。同样的，还有其他的 SELinux User：

引用

root：表示 root 的用户，方便标识而已，因通常很少会以 root 身份运行（系统默认禁止其本地登陆），所以，实际上没什么特别的；

system_u：表示系统程序方面的识别，通常就是程序，或程序创建的文件；

※ 通常为了方便识别，除 root 外，都会加上【_u】的标识。

2）role 部分

这是 SELinux Role-Based Access Control（RBAC）安全模型的一部分。role （角色）就是 RBAC 的属性。SELinux User 由 roles 验证，而 roles 由 domains （域，后面会提到）验证。roles 扮演 domains 和 SELinux users 两者的中间人。获得 roles，就可以获得对应的 domains，也就是可以访问其里面的资源客体（object）。这可以避免权限提升攻击发生时的问题。例如：

引用

id -Z

sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

假设 sysadm_u 被攻破，黑客也只能得到 system_r 可访问的 domain，而 sysadm_r 是不能进行安全配置

的，所以，secadm 所拥有的权限他是无法获得的。

除此以外，常见的 role 还有：

引用

object_r：代表文件或目录等文件资源；

system_r：代表的是程序，或一般使用者。

※ 通常为了方便识别，除 root 外，都会加上【_r】的标识。

3）type 部分

这是 TE 模型的关键属性。type 定义进程的 domain，和文件的类型。SELinux policy rules 定义各个 types 之间能否访问，以及允许 type 进入某个 domian，以及各 domain 访问的内容。默认的规则是禁止的，换句话说只有显示的定义允许某个 type 的规则，那么它才有访问的权限。

在 targetd 安全策略上，user、role 都是可以忽略的，最关键就是 type。但 mls 就不行。

4）level 部分

这是 MLS 和 Multi-Category Security （MCS）模型的属性，是一个范围的级别，lowlevel - highlevel。

共 16 个等级，从低到高，s0 为低等级、s15 为高等级，最高就是 s15。（s0-s0 与 s0 是相同的意思）

每个 level 是一个 sensitivity-category 对，category（分类）共 1024 个级别，是可选的，没有，则只写 sensitivity，如：s0。

如果还存在 category（分类），那么也是通过":"冒号区分，以 sensitivity:category-set 表示，例如：

引用

s0-s15:c0.c1023

这是涵盖了全范围的level，具体的访问限制暂不是，后续另行描述。

MLS 安全策略使用在Labeled Security Protection Profile （LSPP）环境中使用Bell-La Padula Mandatory Access Model 模型，并采用upstream SELinux Reference Policy 策略，比Tagerted 安全要求更高。

3.访问规则

为policy.X 链接规则，需采用访问向量（Access Vector）规则来定义，语法是：

<av_kind> <source_type(s)><target_type(s)>:<class(es)><permission(s)>

含义是：

引用

av_kind：定义动作，有 allow（允许）、dontaudit（不审计，但也不允许）、auditallow（允许并审计）、neverallow（不允许）；

source_type：主体类型，通常是进程尝试访问的域类型；

target_type：客体类型，被访问的域类型；

class：客体的类别；

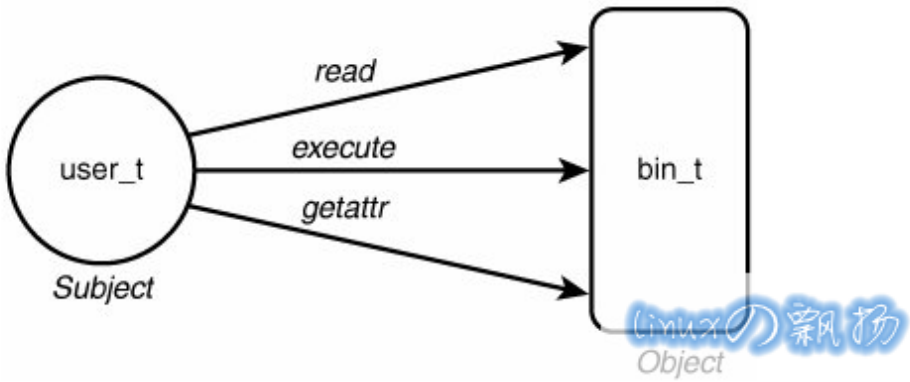
permission：要设置的权限。

一个典型的定义如：

引用

allow user_t bin_t : file {read execute getattr};

这个allow规则的源类型为user_t，目标类型为bin_t，客体类别file，许可execute等，这个规则可以解读为“允许user_t执行类型为bin_t的文件”。（因为要执行文件，必须先可以获取属性getattr，读read，然后执行execute，缺一不可）



4.domain transition（域转移）

域就是上文提到的domain。为什么要设置域呢？安全要解决的一个重要问题就是：隔离。即当一个应用程序（用户）被攻破时，不影响其他应用程序。域就是用于限制的。

一个进程，从某个域通过应用，访问entrypoint（入口）而转移到另一个域，是需要在policy中明确允许的。

举个例：用户修改密码。

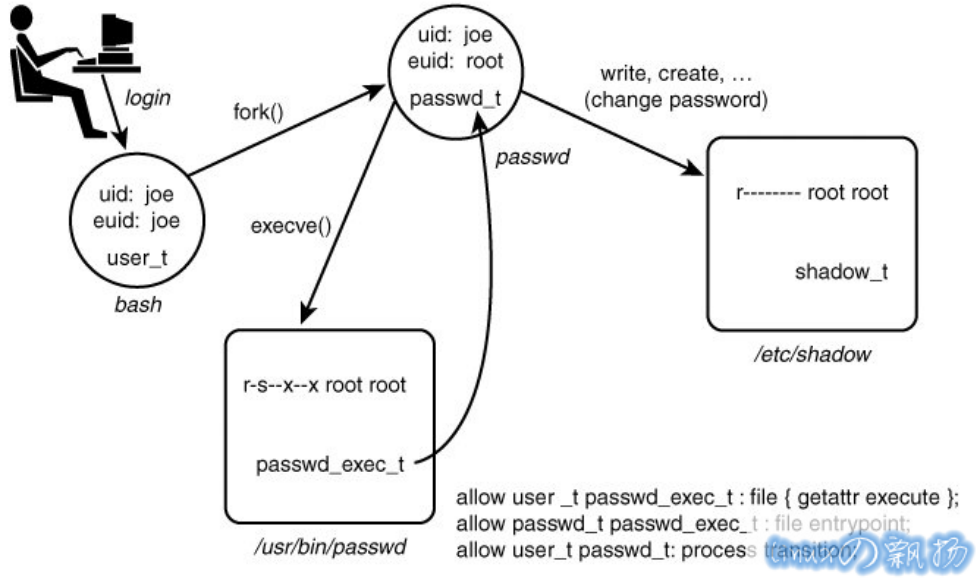
先来看看文件的属性：

引用

```
# ll -Z /usr/bin/passwd
-rwsr-xr-x  root root system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd
# ll -Z /etc/shadow
-r--------  root root system_u:object_r:shadow_t:s0    /etc/shadow
```

（在 RFSOS4 中，需要以 sysadm 登陆后，su - 到 root 才能执行）

可见，类型与域是相关的。蓝色和红色标记的类型，不同，表明两者不在同一个域里面。所以，用户通过 passwd 修改/etc/shadow 文件，就需要进行域转移的许可。



定义一些规则：

引用

```
allow user_t passwd_exec_t : file {getattr execute}
```

允许用户 shell（user_t）在 passwd 可执行文件（passwd_exec_t）上启动 execve()系统调用。

引用

```
allow  passwd_t passwd_exec_t : file entrypoint
```

这条规则提供了对 passwd_t 域的入口访问权，entrypoint 许可在 SELinux 中是一个相当有用的许可权限，这个权限所做的事情是定义哪个可执行文件（程序）可以"进入"某个特定的域。

引用

```
allow user_t passwd_t : process transition
```

原始的类型（user_t）到新的类型（passwd_t）进行域转变必须有 transition 许可才允许进行。

这就是域转移。当然，除此以外，当执行程序还需要一些额外的权限，例如：例如：使用父进程的文件描述符（例如标准输入，输出，以及出错设备），子进程运行结束时，发送 sigchld 等，也是需要明确定义的：

引用

allow passwd_t user_t:fd use;

allow passwd_t user_t:fifo_file rw_fifo_file_perms;

allow passwd_t user_t:process sigchld;

※ 注意：

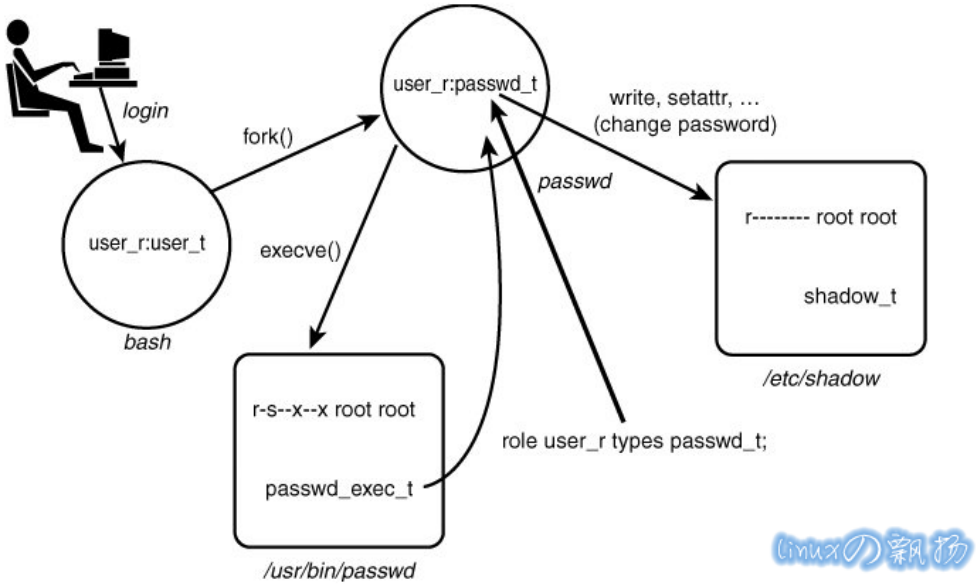
上述提及的是限制域进程（Confined Processes）运行的情况，而系统中，还有一个由非限制域进程（Unconfined Processes）运行的**unconfined_t** 域，

对于非限制域进程，SELinux 规则时有效的，但策略是对于运行在非限制域的进程都是可访问几乎任意资源的，这时，实际上域的限制就不存在了。换句话说，这又回到了DAC模型的限制规则中，因此，除特殊的进程外，不应该采用这个unconfined_t 域。

更详细的例子，可参考：Unconfined Processes

5.Role-Based Access Control（RBAC）安全模型

如果采用的是Targeted安全策略，那么上面定义的TE规则已经足够。但在MLS安全策略下，增加的RBAC模型，则还需要对角色进行判断。前面提过，roles 扮演domains 和SELinux users 两者的中间人。普通用户的角色必须允许与新的域类型建立关联后，才可允许密码程序通过user_t域类型执行，然后转移到新的passwd_t域上。



安全规则为：

引用

role user_r type passwd_t;

所以说，RBAC进一步增加了TE模型的限制：**主客体域类型的角色要一致**！

可能你会问，这些在那里定义？在.te文件中编写，然后编译为.pp模块，然后才能加载到内核中。后续会讲到一些基本方法，暂时只需知道有这样的概念即可。

6.Multi-Level Security （MLS）和 Multi-Category Security （MCS）

这就是level 属性，也是MLS安全策略比Targeted增强的地方。简单来说，就是做了上面工作还不足够，如果level 属性限制你，你还需要匹配规则，够烦吧！！

MLS 和 MCS 定义的level 就是用于Bell-La Padula Mandatory Access Model 模型的：

引用

BLP 模型的基本安全策略是“**下读上写**”，即主体对客体向下读、向上写。主体可以读安全级别比他低或相等的客体，可以写安全级别比他高或相等的客体。“下读上写”的安全策略保证了数据库中的所有数据只能按照安全级别从低到高的流向流动，从而保证了敏感数据不泄露。

怎么看级别呢？上面已经讲过，每个 level 都是一个 sensitivity-category 对（简称 SC），而每个 Sensitivity Level （简称 SL）和 Category Level （简称 CL）也都是一段范围的，例如：

引用

id -Z

sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

这表明，当前 SELinux User 是 sysadm_u（在 RFSOS4 下，登陆用户是 sysadm），角色是 sysadm_r，类型是 sysadm_t。当前级别 SL 为 s0，最大可达到级别 SL 为 s15:c0.c1023。

先看 SL，是由 lowlevel - highlevel 构成，lowlevel 就是当前级别，highlevel 就是可达到最高级别。各级 SL 是单调递增的。CL 类似。

SL 之间的运算关系有四种：dom、domby、eq、incomp。

引用

dom:

(dominates) SL1 dom SL2 if the sensitivity of SL1 is higher or equal to the sensitivity of SL2, and the categories of SL1 are a superset of the categories of SL2.

domby:

(dominated by) SL1 domby SL2 if the sensitivity of SL1 is lower than or equal to the sensitivity of SL2, and the categories of SL1 are a subset of the categories of SL2.

eq:

(equals) SL1 eq SL2 if the sensitivity of SL1 and SL2 are equal, and the categories of SL1 and SL2 are the same set.

incomp:

(incomparable or noncomparable) SL1 incomp SL2 if the categories of SL1 and SL2 cannot be compared (that is, neither is a subset of the other).

假设我们把权限分两类：

引用

读权限和写权限，例如文件的写权限 write create setattr relabelfrom append unlink link rename mounton，其余的归为读权限。

在源码 policy/mls 文件中，有这样的定义：

引用

```
# the file "read" ops (note the check is dominance of the low level)
mlsconstrain { dir file lnk_file chr_file blk_file sock_file fifo_file } { read getattr execute }
        (( l1 dom l2 ) or
```

((t1 == mlsfilereadtoclr) and (h1 dom l2)) or

(t1 == mlsfileread) or

(t2 == mlstrustedobject));

在其他条件不满足的情况下，l1 和 l2 的关系就决定了读的权限（read getattr execute ）。即 SL1 小于等于 SL2，可读。（“下读”嘛！）

同样的，有：

引用

mlsconstrain file { write create setattr relabelfrom append unlink link rename mounton } (l1 domby l2);

那就是 SL1 大于等于 SL2，可写。（“上写”！）因此，SL1 等于 SL2，就是可读可写了！

由此可见，基于多层次访问模型（MLS）在 RBAC(基于角色的访问模型)和 TE 模型基础上进一步限制——除了满足 RBAC 和 TE 规则外还必须满足“下读上写”规则安全策略实现。

※ 注意

红旗安全操作系统 4.0 中 BLP 模型改了，变成“**高等级可读低等级，相等可写可读，其他都是非读非写**”！

7.进程的安全上下文

进程在 Linux 系统中也是以文件的形式出现的，所以，进程也有安全上下文（Security Contexts，简称 contexts）。

我们以上面提到“域转移”的例子来看看。首先，打开两个终端，以 sysadm 登陆，并在其中一个上运行 pssswd，会提示输入新密码。这时，切换到另一个终端上，运行下面的命令：

引用

ps -eZ|grep passwd

sysadm_u:sysadm_r:passwd_t:s0-s15:c0.c1023 10057 pts/0 00:00:00 passwd

蓝色标记的就是进程的安全上下文，它们在/proc 目录中是以文件的形式存在的。另外，这里还可看到已发生域转移：

引用

sysadm_t > passwd_t

8.用户的安全上下文

用户要访问资源，同样受 SELinux 的限制，有它自己的安全上下文，可用下面的命令查看：

引用

```
# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023
# id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023
```

id 命令在之前已多次用过，这很重要。因为用户名对应 uid，也就是说，只是在 DAC 模型有用，而 SELinux 采用的 MAC 模型，对比的是安全上下文，所以，实际操作时，千万不要给用户名所误导了！

※ 通常，命令中都是使用【-Z】参数查看与安全上下文有关的信息。

※ 此外，系统中还存在一个 **unconfined_u** 的 SELinux User，他与 unconfined_t 域是不同的。被映射到 unconfined_u 的用户还是会限制执行和读写内存等，并且受 MCS 和 MLS 规则限制。

RHEL 6 创建的默认用户就是处在这个域：

引用

```
# /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
```

其还定义了一堆的特殊用户，如guest_u、xguest_u 等，详细可见：Confined and Unconfined Users。RFSOS4 有点不同，默认User映射到user_u，还有sysadm在远程终端会映射到staff_u（本地登陆还是sysadm_u）。

概念说得够多了，虽然很累，但必须弄清楚。后面的大多都是操作，轻松些。

4. 登陆系统

大部分概念和术语都已讲完，其余的会穿插在后面的操作描述中。由于 RFSOS4 不单采用 SELinux 这一安全架构，还增加了不少安全方面的设置，所以在实际操作中与标准的 RHEL 是略有差异的。我是在学习 RFSOS4 的过程中编写这篇文章的，故配置文件和命令操作结果均以 RFSOS4 显示为准。正式使用的第一步，当然是登陆进入操作系统。

四、登陆系统

1.三权分立

RFSOS4 采用 MLS 作为默认安全策略，把原来一人独大的 root 用户，拆分为三个管理员用户，分别如下：

引用

sysadm：系统管理员，主要负责原来 root 的大部分系统管理工作，例如：服务启动、用户管理、系统配置等；

secadm：安全管理员，主要负责与安全功能相关的配置和管理服务，例如：安全模式修改、安全配置文件修改等；

auditadm：审计管理员，主要负责安全日志审计工作，例如：检索日志、汇总日志、日志分类等；

三个用户各不相关，采用独立的密码，映射到不同的 SELinux User 上，并赋予不同的角色权限：

引用

```
# semanage login -l

Login Name           SELinux User         MLS/MCS Range

__default__          user_u               s0
auditadm             auditadm_u           s0-s15:c0.c1023
root                 root                 s0-s15:c0.c1023
secadm               secadm_u             s0-s15:c0.c1023
sysadm               sysadm_u             s0-s15:c0.c1023
system_u             system_u             s0-s15:c0.c1023
```

用户登录后，可通过 id -Z 命令查看当前用户的信息及安全上下文（简称 context）：

引用

id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

id -Z

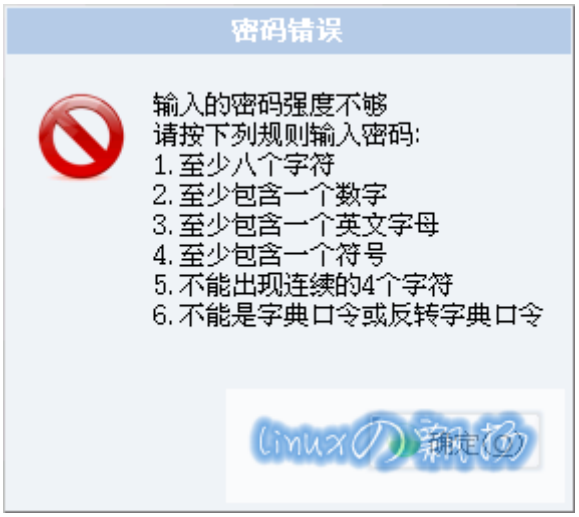
sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

RFSOS4 安装完毕后，默认有四个用户：root/sysadm/secadm/auditadm，他们的密码都是独立的，root 的密码在标准安装界面时输入：

安全用户 sysadm/secadm/auditadm 的密码在安装时的独立界面输入：

初始为三个安全用户采用相同的密码。但根据三权分立的要求，从安全的角度考虑，建议在系统启动后，使用 rolepasswd 命令修改为各自不同的密码。

※ 注意，RFSOS4 对用户密码有强制要求，至少包括 6 个字符，至少包含一个数字，至少包含一个英文字母，至少包含一个符号，不能出现连续的 4 个字符，不能是字典口令或反转字典口令！



2.本地终端登录

RFSOS4 禁止 root 从本地终端登录系统。这是在/etc/pam.d/login 文件中作的修改：

引用

auth required pam_succeed_if.so user != root quiet

还有一些对登录错误可允许次数，超过时禁止时间等，都在该配置文件中：

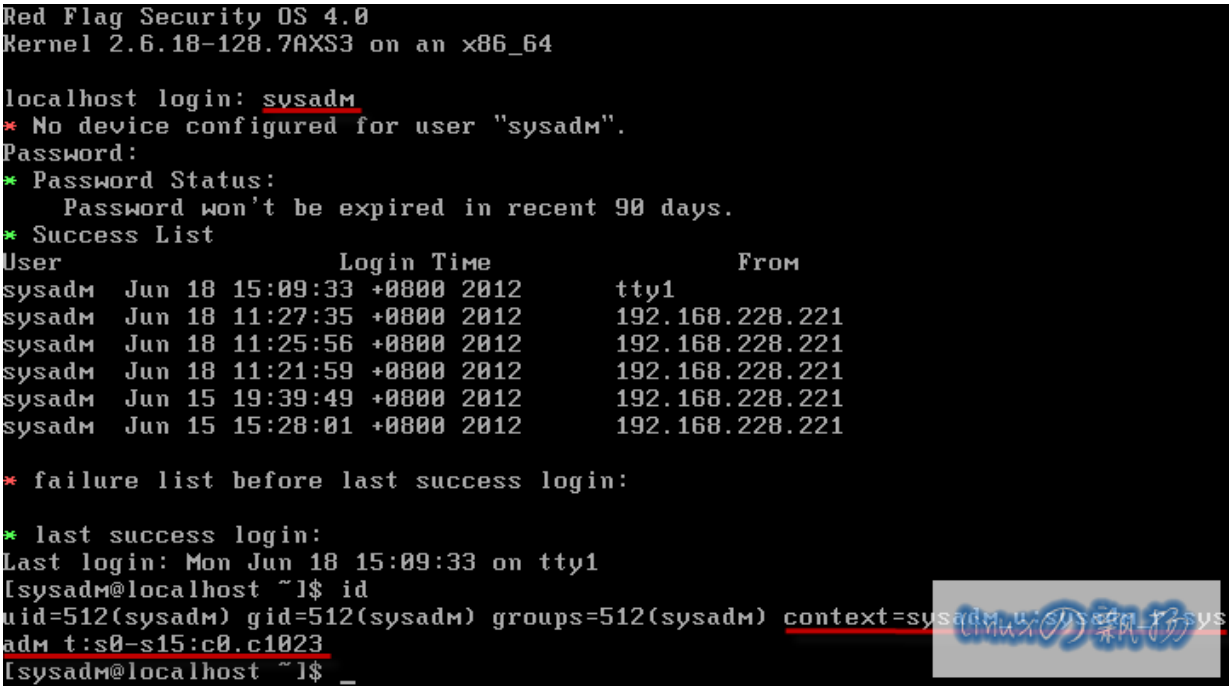
引用

auth required pam_tally2.so deny=10 onfail=err unlock_time=28800

这里不做详细说明，具体可参考man pam_tally2或系统中

/usr/share/doc/pam-0.99.6.2/txts/README.pam_tally2 说明。

本地终端登录的情况：



SELinux是在原有DAC模型的基础上对安全进行加强，而并不是丢弃原来的DAC规则。所以，普通用户 sysadm 是没有执行属主为root 的文件权限的。下面，*我们以用户身份表示DAC 的uid/gid形式，以用户角色表示MAC 的role形式，以便于描述。*

正确的登录流程是：

引用

1）使用 sysadm/secadm/aduitadm 等用户登录；

2）使用 su - 命令切换到 root 身份。

例如：

```
[sysadm@localhost ~]$ id
uid=512(sysadm) gid=512(sysadm) groups=512(sysadm) context=sysadm_u:sysadm_r:sys
adm_t:s0-s15:c0.c1023
[sysadm@localhost ~]$ id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023
[sysadm@localhost ~]$ su -
Password:
[root/sysadm_r/s0@~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel) context=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023
[root/sysadm_r/s0@~]# id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023
```

请留意，虽然用户名和 uid/gid 都改了，但至此至终 context 是没有改变的，MAC 规则是依据 context 判断的，而不是 uid/gid。

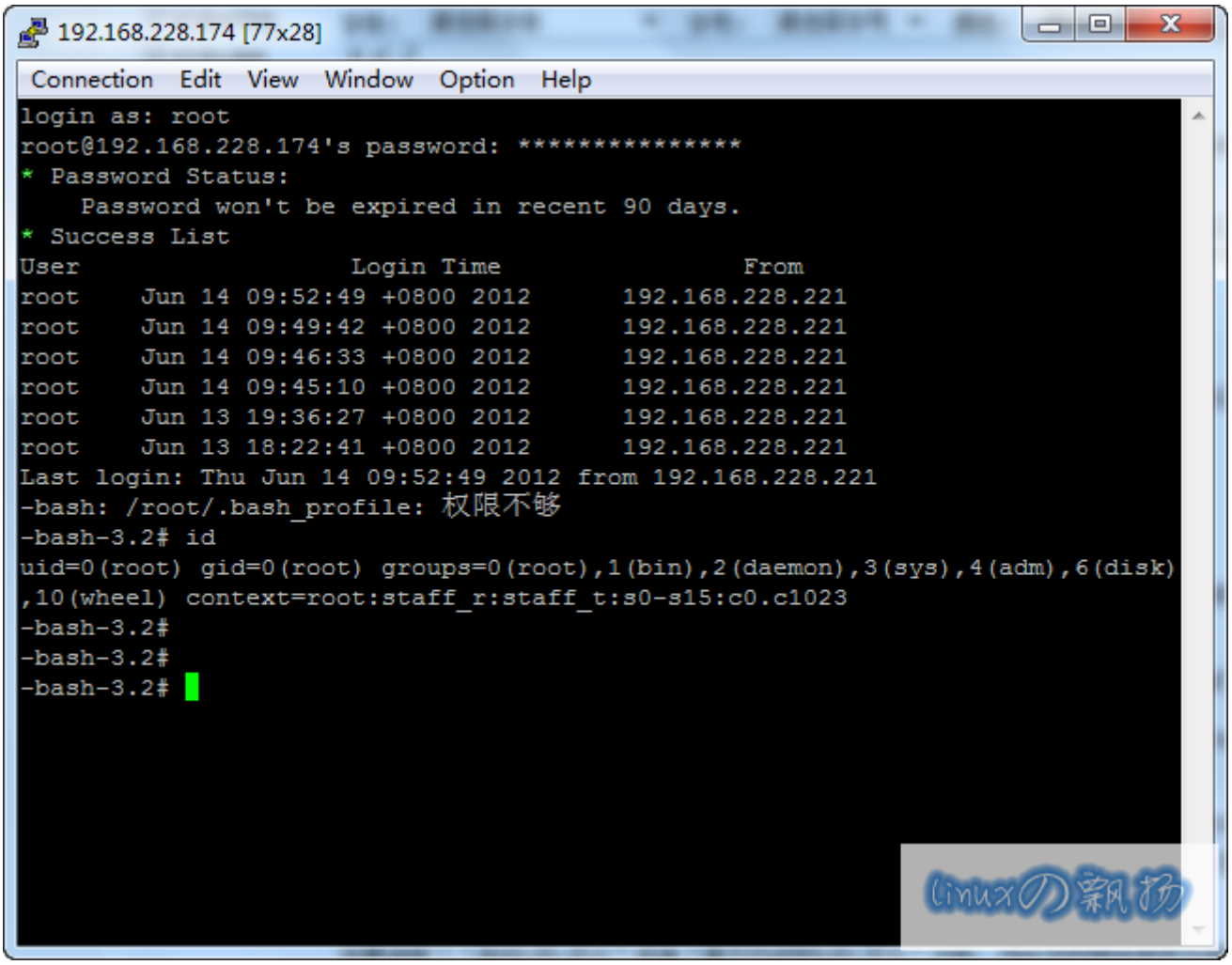
3.远程终端登录

使用远程终端登录时，情况有点不同。首先，root 是可以登陆的，但 sshd 服务把其禁掉了，这在 /etc/ssh/sshd_config 文件中：

引用

permitRootLogin no

只要把 no 改为 yes，重启 sshd 服务，root 就可以登陆了。但是，登陆后，你会发现其 context 并不是 root：



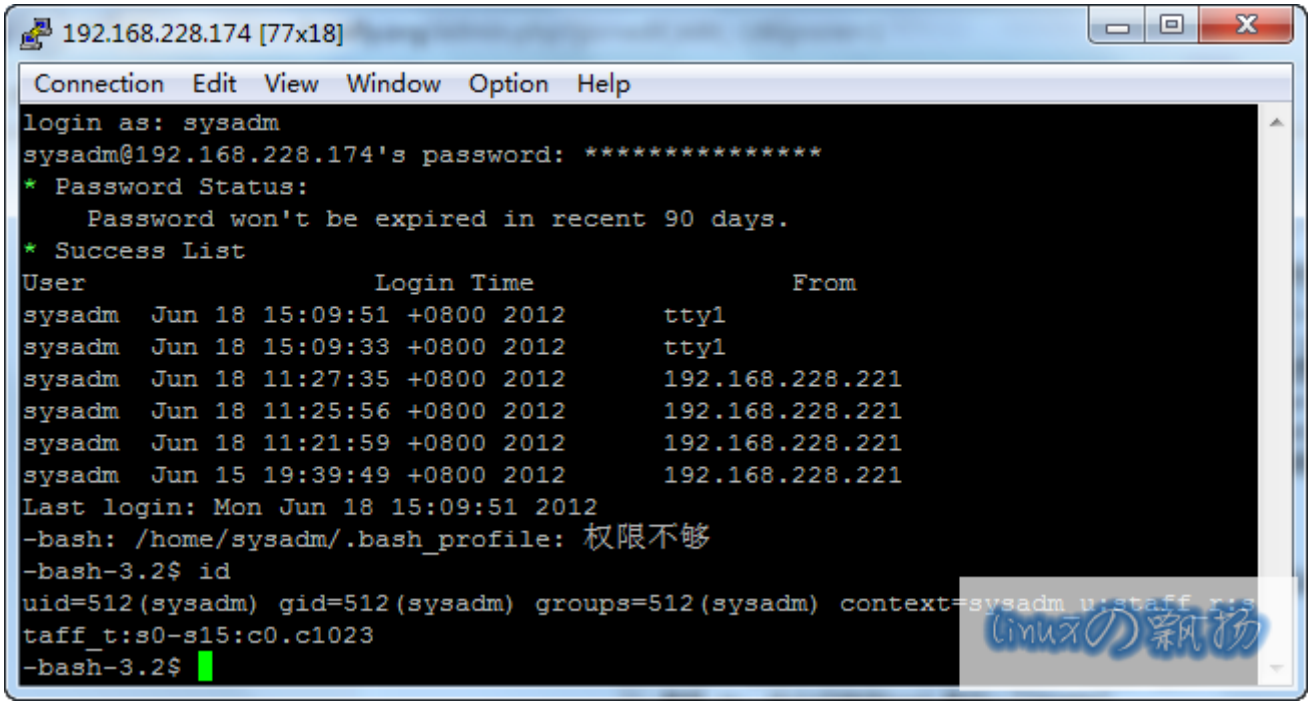
引用

-bash-3.2# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=root:staff_r:staff_t:s0-s15:c0.c1023

这是对 root 和 sysadm 远程访问时，把其角色权限压减了。你采用 sysadm 登陆也是一样的：



引用

-bash-3.2$ id

uid=512(sysadm) gid=512(sysadm) groups=512(sysadm)

context=sysadm_u:staff_r:staff_t:s0-s15:c0.c1023

※ 注意，虽然这时 root 或 sysadm 都是 staff_r 角色，但 SELinux User 是不相同的。

RFSOS4 只允许 staff_r 与 sysadm_r 之间相互转移。因此，在登陆 root ，可使用 newrole -r 切换用户身份：

引用

-bash-3.2# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=root:staff_r:staff_t:s0-s15:c0.c1023

-bash-3.2# newrole -r sysadm_r

口令：

id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=root:sysadm_r:sysadm_t:s0-s15:c0.c1023

root 是这样，sysadm 呢？还要麻烦一点，再多一步，由于 DAC 的要求，需要 su 到 root：

引用

-bash-3.2$ id

uid=512(sysadm) gid=512(sysadm) groups=512(sysadm)

context=sysadm_u:staff_r:staff_t:s0-s15:c0.c1023

-bash-3.2$ newrole -r sysadm_r

口令：

[sysadm@localhost ~]$ id

uid=512(sysadm) gid=512(sysadm) groups=512(sysadm)

context=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

[sysadm@localhost ~]$ su -

口令：

[root/sysadm_r/s0@~]# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

可见，这里要多输入一次 root 的密码口令。

若以 secadm/auditadm 登陆就不用执行 newrole -r 这步了：

引用

[secadm@localhost ~]$ id

uid=513(secadm) gid=513(secadm) groups=513(secadm)

context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023

[secadm@localhost ~]$ su -

口令：

[root/secadm_r/s0@~]# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023

但 root 不能切换到 secadm_r 或 audit_adm_r 身份：

引用

-bash-3.2# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=root:staff_r:staff_t:s0-s15:c0.c1023

-bash-3.2# newrole -r secadm_r

口令：

执行 shell 失败

: 权限不够

够麻烦的，小结一下吧：

引用

1）远程登录时，修改配置可让 root 登陆，但由于安全上下文不同，不建议采用；

2）应该以 sysadm/secadm/auditadm 用户登陆系统；

3）当以 sysadm 登陆后，需使用 newrole -r sysadm_r 切换到 sysadm_r 角色，其他用户不需要；

4）为满足 DAC 的要求，需使用 su - 切换到 root 身份，至此，登陆完成。

※ 以后若不做特殊说明，以某安全用户执行命令，均指已完成上述登陆工作，并获得正确的身份（uid/gid）和角色（role）。

4.修改密码

三个管理员登陆后，切换到正确的角色和 root 身份，即可使用 rolepasswd 修改各自的密码：

引用

[secadm@localhost ~]$ id

uid=513(secadm) gid=513(secadm) groups=513(secadm)

context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023

[secadm@localhost ~]$ rolepasswd

Only root can use this program.

[secadm@localhost ~]$ su -

口令：

[root/secadm_r/s0@~]# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023

[root/secadm_r/s0@~]# rolepasswd

context = secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023

Your Role: secadm [secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023]

Changing password for user secadm.

New UNIX password:

※ 注意：root 还是使用 passwd 修改密码。实际上，在该环境中，root 已几乎被弱化为普通用户，只是 DAC 模型上用于区分特殊权限而已。

5.忘记密码

忘记密码怎么办？实际上，SELinux User 是由 Unix User 映射而来的，因此，若忘记三个管理员的密码。还是与以往一样，启动系统时，在 Grub 菜单，输入 selinux=0 single，进入系统后，通过 passwd 修改即可。

6.添加新用户

添加新用户时，不带参数创建的新用户，其角色就是上面提及的__default__所对应的 user_t。

若要指定角色，可用-Z 参数：

引用

[root/sysadm_r/s0@~]# useradd -Z sysadm_u newuser

[root/sysadm_r/s0@~]# passwd newuser

Changing password for user newuser.

New UNIX password:

Retype new UNIX password:

passwd: all authentication tokens updated successfully.

[root/secadm_r/s0@~]# semanage login -l

Login Name SELinux User MLS/MCS Range

__default__ user_u s0

auditadm auditadm_u s0-s15:c0.c1023

newuser sysadm_u s0

root root s0-s15:c0.c1023

secadm secadm_u s0-s15:c0.c1023

sysadm sysadm_u s0-s15:c0.c1023

system_u system_u s0-s15:c0.c1023

注意，我这里是分别采用 sysadm 和 secadm 用户来执行不同命令的。若以 sysadm 执行 semanage，会提示“权限不足”：

引用

[root/sysadm_r/s0@~]# semanage

bash: /usr/sbin/semanage: 权限不够

以 newuser 登陆试试：

引用

login as: newuser

newuser@192.168.228.174's password: **********

* Password Status:

Password won't be expired in recent 90 days.

-bash: /home/newuser/.bash_profile: 权限不够

-bash-3.2$ id

uid=523(newuser) gid=523(newuser) groups=523(newuser) context=sysadm_u:staff_r:staff_t:s0

-bash-3.2$ newrole -r sysadm_r

口令：

[newuser@localhost ~]$ id

uid=523(newuser) gid=523(newuser) groups=523(newuser) context=sysadm_u:sysadm_r:sysadm_t:s0

[newuser@localhost ~]$ su -

口令：

[root/sysadm_r/s0@~]# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=sysadm_u:sysadm_r:sysadm_t:s0

除了 Security Level 没有设置外，与 sysadm 都是相同的。（Security Level 的设置以后再说）

删除用户：

引用

[root/sysadm_r/s0@~]# userdel -r newuser

7.优化 bash 显示

如果你是按照我上面的步骤进行模拟实验的，可能会发现，在以 secadm/auditadm 登陆后，并 su 到 root 后，命令提示符会显示为：

引用

-bash-3.2#

这是因为，secadm/auditadm 所在类型 secadm_t/auditadm_t 默认不能访问/root 目录的 sysadm_home_dir_t 和 sysadm_home_t 类型文件：

引用

[auditadm@localhost ~]$ su -

口令：

-bash: /root/.bash_profile: 权限不够

上面我为了说明，多次执行 id，边便说明应该以那个安全用户执行，这非常的麻烦。而在实际环境中，打开的窗口多了，这来回切换也很容混淆。

我这里编译了一个简单的 PP 模块：

下载文件

点击这里下载文件

加载到 SELinux 核心后，可让 secadm_t 和 audit_t 类型访问/root 目录。把文件上传到系统中，放在/tmp 目录下（这里权限比较宽松些），以 secadm 用户登录后，执行：

cd /tmp

tar xjvf enter_root.tar.bz2

semodule -i enter_root.pp

semodule -l |grep enter

enter_root 1.0

然后编辑/root/.bash_profile，加入蓝色部分的内容：

引用

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

if [ ! -z "$PS1" ]; then
   PS1="[\u/\$(secon -rP 2>/dev/null)/\$(secon -lP 2>/dev/null)@\W]\\$ "
   export PS1
fi
```

工作做完了，你用 secadm 或 auditadm 再次登陆看看，是否变成这样：

引用

```
[root/secadm_r/s0@~]#  id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023
[root/auditadm_r/s0@~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=auditadm_u:auditadm_r:auditadm_t:s0-s15:c0.c1023
```

当然，这部分操作是可选的，如果你觉得不安全可以不用。但后续进行命令演示时，我不再特殊说明执行的用户，请留意命令前的用户角色吧。

5. 运行服务

成功分别以不同用户登录 RFSOS4 系统后，接下来，就是启动服务，并使其正常运行。这就涉及到安全上下文（简称 context，也就是强制访问控制模型中的敏感标记）的识别与规则匹配的问题。如果你不是开发者，而是系统管理员，那通常情况下，你需要维护系统中文件的 context，使得其处于合适的域中，以便系统服务可以正确的访问这些文件。

五、启动服务

1.SELinux 状态

使用 sestatus 命令查看 SELiux 状态：

引用

[root/secadm_r/s0@~]# sestatus

SELinux status: enabled

SELinuxfs mount: /selinux

Current mode: enforcing

Mode from config file: enforcing

Policy version: 21

Policy from config file: mls

2.创建文件

我们以最常见的 Apache 服务为例，系统中服务名称为 httpd，默认网页存放路径位于“/var/www/html”目录中。先看看该目录的属性和安全上下文情况：

引用

[root/sysadm_r/s0@~]# ll -Zd /var/www/html

drwxr-xr-x root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html

DAC 规则定义的是，root 可以对该目录进行读写，其他用户只能读和访问该目录。同时，MAC 定义的 SELinux User 为 system_u，Role（角色）为 object_t，Type（类型）为 httpd_sys_content_t，Level （级别）是 s0。暂时把 SL 级别都定在 s0，以免影响示例结果。

以 sysadm 用户创建一个文件：

引用

[root/sysadm_r/s0@~]# cd /var/www/html

[root/sysadm_r/s0@html]# touch file1

[root/sysadm_r/s0@html]# ll -Z file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

3.启动服务

引用

[root/sysadm_r/s0@~]# service httpd start

启动 httpd： [确定]

打开另一个终端，用 wget 下载试试：

引用

[root/sysadm_r/s0@~]# cd /tmp

[root/sysadm_r/s0@tmp]# wget http://localhost/file1

--19:53:13-- http://localhost/file1

正在解析主机 localhost... 127.0.0.1

Connecting to localhost|127.0.0.1|:80... 已连接。

已发出 HTTP 请求，正在等待回应... 200 OK

长度：0 [text/plain]

Saving to: `file1'

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

19:53:13 (0.00 B/s) - `file1' saved [0/0]

没有问题，很好！

4.创建第二个文件

在/root 目录下创建第二个文件，并拷贝到/var/www/html 目录中：

引用

[root/sysadm_r/s0@~]# pwd

/root

[root/sysadm_r/s0@~]# touch file2

[root/sysadm_r/s0@~]# mv file2 /var/www/html

[root/sysadm_r/s0@~]# ll -Z /var/www/html/

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

-rw-r--r-- root root sysadm_u:object_r:sysadm_home_t:s0 file2

哦？file2 除了 context 与 file1 的不同，其他都是一样的。

试试用 wget 下载：

引用

[root/sysadm_r/s0@tmp]# wget http://localhost/file2

--19:58:53-- http://localhost/file2

正在解析主机 localhost... 127.0.0.1

Connecting to localhost|127.0.0.1|:80... 已连接。

已发出 HTTP 请求，正在等待回应... 403 Forbidden

19:58:53 错误 403：Forbidden。

报 403 错误。如果你以 auditadm 去查看审计日志，会发现类似的信息：

引用

[root/auditadm_r/s0@~]# grep httpd /var/log/audit/audit.log

type=AVC msg=audit(1340193533.883:20566): avc: denied { getattr } for pid=12030 comm="httpd"

path="/var/www/html/file2" dev=sda1 ino=2556185

scontext=sysadm_u:system_r:httpd_t:s0-s15:c0.c1023 tcontext=sysadm_u:object_r:sysadm_home_t:s0

tclass=file

type=SYSCALL msg=audit(1340193533.883:20566): arch=c000003e syscall=6 success=no exit=-13

a0=2b9ce05d7350 a1=7fffbcf977a0 a2=7fffbcf977a0 a3=0 items=0 ppid=12020 pid=12030 auid=512

uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=3279 comm="httpd"

exe="/usr/sbin/httpd" subj=sysadm_u:system_r:httpd_t:s0-s15:c0.c1023 key=(null)

原因就是 context 不正确。file2 是在/root 目录下创建的，其默认继承父目录的 context。当 mv 到 /var/www/html 后，其 context 不会改变（这个后续会进一步说明）。而受规则的限制，httpd 进程所在的 httpd_t 域是不能访问 sysadm_home_t 的，这导致访问失败。

5.恢复安全上下文

执行 **restorecon** 命令可依据规则定义，恢复文件的默认 context：

引用

[root/sysadm_r/s0@html]# restorecon -v file2

restorecon reset /var/www/html/file2 context

sysadm_u:object_r:sysadm_home_t:s0->system_u:object_r:httpd_sys_content_t:s0

[root/sysadm_r/s0@html]# ll -Z file2

-rw-r--r-- root root system_u:object_r:httpd_sys_content_t:s0 file2

再用 wget 就没问题了：

引用

[root/sysadm_r/s0@tmp]# wget http://localhost/file2

--20:11:54-- http://localhost/file2

正在解析主机 localhost... 127.0.0.1

Connecting to localhost|127.0.0.1|:80... 已连接。

已发出 HTTP 请求，正在等待回应... 200 OK

长度：0 [text/plain]

Saving to: `file2'

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

20:11:54 (0.00 B/s) - `file2' saved [0/0]

也可以 **chcon** 命令手动修改文件的 context：

引用

[root/sysadm_r/s0@html]# chcon -t samba_share_t file2

[root/sysadm_r/s0@html]# ll -Z file2

-rw-r--r-- root root system_u:object_r:samba_share_t:s0 file2

[root/sysadm_r/s0@html]# chcon -t httpd_sys_content_t file2

[root/sysadm_r/s0@html]# ll -Z file2

-rw-r--r-- root root system_u:object_r:httpd_sys_content_t:s0 file2

6.查看允许规则

有时候，我们对服务可访问的规则不了解，可以查询规则。RFSOS4 提供一个叫“安全管理”的图形工具（启动图形后，在桌面可以打开），用于查询系统内置的规则。但默认没有安装 setools 工具包。我这里提供一个：

下载文件

点击这里下载文件

（含 x86_64 的 rpm，及 src.rpm 源码包）

安装后，可使用 **sesearch** 命令查询规则。不过，我们先用 **ps -eZ** 命令来看看 Apache 服务所处在的域（类型）：

引用

```
[root/sysadm_r/s0@~]# ps -eZ|grep httpd
sysadm_u:system_r:httpd_t:s0-s15:c0.c1023 12020 ? 00:00:00 httpd
sysadm_u:system_r:httpd_t:s0-s15:c0.c1023 12028 ? 00:00:00 httpd
```

好了，知道主体和客体，那就可以查询：

引用

```
[root/secadm_r/s0@~]# sesearch -s httpd_t -t httpd_sys_content_t --allow
Found 6 av rules:
   allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock };
   allow httpd_t httpd_sys_content_t : dir { ioctl read getattr lock search };
   allow httpd_t httpd_sys_content_t : lnk_file { ioctl read getattr lock };
   allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock };
   allow httpd_t httpd_sys_content_t : dir { ioctl read getattr lock search };
   allow httpd_t httpd_sys_content_t : lnk_file { read getattr };
```

查询以主体 httpd_t 访问的规则，可使用：

```
[root/secadm_r/s0@~]# sesearch -s httpd_t  -a
```

具体的，请看 man，或用--help 帮助。

小结一下，如果你想让服务可正常运行，那应让其作为主体，可顺利的访问客体。客体（通常是文件或资源）必须拥有正确的 context。访问规则，context 类型（域）都是在 policy 中定义好的。换句话说，如果你放得位置不对，或 context 不正确，那访问就可能失败。
（当然，这仅是部分的原因，还有 Booleans，MLS/MCS 等限制，后面陆续会提到）
可见，这把文件的位置都固定好，不像以往可随便存放，任意修改。所以称强制访问控制（MAC）。你要改怎么办？后面继续。。

6. 处理安全上下文

主体是否能顺利访问客体，依赖于其两者（或更多中间者）所处的类型域和规则。系统中为很多的自带服务提供了类型域和规则定义。作为系统管理员，就是需要让这些服务可顺利的运行。其中最重要的，就是要处理文件的安全上下文（context）。处理的动作包括：创建、拷贝、移动、修改等。

六、处理安全上下文

1.创建文件

创建新文件时，若文件不带 context（例如解压包），或不指定 context，会采用默认标签规则。例如：

引用

```
[root/secadm_r/s0@~]# semanage fcontext -l|grep '/var/www(/.*)?'
/var/www(/.*)?                    all files      system_u:object_r:httpd_sys_content_t:s0
/var/www(/.*)?/logs(/.*)?         all files      system_u:object_r:httpd_log_t:s0
```

第一列是正规表达式，表示/var/www/目录下的所有文件。故：

引用

```
[root/sysadm_r/s0@~]# cd /var/www/html
[root/sysadm_r/s0@html]# touch file1
[root/sysadm_r/s0@html]# ll -Z file1
-rw-r--r--  root root sysadm_u:object_r:httpd_sys_content_t:s0 file1
```

2.拷贝文件

拷贝时，在没有额外参数的情况下，新创建的文件是基于默认标签规则，而不是保留源文件的 context 的。

引用

```
[root/sysadm_r/s0@html]# ll -Z file1
-rw-r--r--  root root sysadm_u:object_r:httpd_sys_content_t:s0 file1
[root/sysadm_r/s0@html]# ll /etc/file1
ls: /etc/file1: 没有那个文件或目录
[root/sysadm_r/s0@html]# cp file1 /etc/
[root/sysadm_r/s0@html]# ll -Z /etc/file1
-rw-r--r--  root root sysadm_u:object_r:etc_t:s0       /etc/file1
```

如果目标文件已经存在，那么，默认情况下，目标的 context 是不会改变的：

引用

[root/sysadm_r/s0@html]# chcon -t samba_share_t /etc/file1

[root/sysadm_r/s0@html]# ll -Z /etc/file1

-rw-r--r-- root root sysadm_u:object_r:samba_share_t:s0 /etc/file1

[root/sysadm_r/s0@html]# ll -Z file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

[root/sysadm_r/s0@html]# cp file1 /etc/

cp：是否覆盖“/etc/file1”? y

[root/sysadm_r/s0@html]# ll -Z /etc/file1

-rw-r--r-- root root sysadm_u:object_r:samba_share_t:s0 /etc/file1

使用**--preserve=context** 参数（简写-c），则可以保留源文件的安全上下文：

引用

[root/sysadm_r/s0@html]# cp --preserve=context file1 /etc/

cp：是否覆盖“/etc/file1”? y

[root/sysadm_r/s0@html]# ll -Z /etc/file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 /etc/file1

使用-Z 参数，可指定目标的安全上下文：

引用

[root/sysadm_r/s0@html]# cp -Z system_u:object_r:samba_share_t:s0 file1 file2

[root/sysadm_r/s0@html]# ll -Z file*

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

-rw-r--r-- root root system_u:object_r:samba_share_t:s0 file2

3.移动文件

移动是会保持 context 的。但对于目标路径来讲，这通常是不合适的。

引用

[root/sysadm_r/s0@html]# ll -Z file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

[root/sysadm_r/s0@html]# rm /etc/file1

rm：是否删除 一般空文件 “/etc/file1”? y

[root/sysadm_r/s0@html]# mv file1 /etc/

[root/sysadm_r/s0@html]# ll -Z /etc/file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 /etc/file1

※注意：拷贝文件和目录，而不是移动。这有助于确保它们拥有合适的 context。不正确的 context 会防止进程访问这些文件和目录。

4.使用 tar 命令

默认情况下，tar 不会保留 context。例如，普通压缩：

引用

[root/sysadm_r/s0@html]# touch file{1,2,3}

[root/sysadm_r/s0@html]# ll -Z file*

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file2

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file3

[root/sysadm_r/s0@html]# tar czvf test.tar.gz file*

解压：

引用

[root/sysadm_r/s0@html]# mkdir /test

[root/sysadm_r/s0@html]# cd /test

[root/sysadm_r/s0@test]# tar xzvf /var/www/html/test.tar.gz

file1

file2

file3

[root/sysadm_r/s0@test]# ll -Z file*

-rw-r--r-- root root sysadm_u:object_r:root_t:s0 file1

-rw-r--r-- root root sysadm_u:object_r:root_t:s0 file2

-rw-r--r-- root root sysadm_u:object_r:root_t:s0 file3

可加入--selinux 参数来记录 context 信息：

引用

[root/sysadm_r/s0@test]# rm -f file*

[root/sysadm_r/s0@test]# cd /var/www/html

[root/sysadm_r/s0@html]# ll -Z file*

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file2

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file3

[root/sysadm_r/s0@html]# tar --selinux -czvf test.tar.gz file*

[root/sysadm_r/s0@html]# cd /test

[root/sysadm_r/s0@test]# tar xzvf /var/www/html/test.tar.gz

file1

file2

file3

[root/sysadm_r/s0@test]# ll -Z file*

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file1

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file2

-rw-r--r-- root root sysadm_u:object_r:httpd_sys_content_t:s0 file3

对于没有保留 context 的 tar 包，可使用下面的方式，使得解压出来的文件与系统中的上下文一致：

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

※ 此外，tar 还有一个--xattrs 参数用于保留扩展属性 acls 和 selinux 的 context。

5.检查安全上下文

使用 **matchpathcon** 命令可以检查文件和目录的 context 是否与规则一致：

引用

[root/sysadm_r/s0@html]# cd /var/www/html

[root/sysadm_r/s0@html]# chcon -t samba_share_t file1

[root/sysadm_r/s0@html]# ll -Z file*

-rw-r--r-- root root system_u:object_r:samba_share_t:s0 file1

-rw-r--r-- root root system_u:object_r:httpd_sys_content_t:s0 file2

-rw-r--r-- root root system_u:object_r:httpd_sys_content_t:s0 file3

[root/sysadm_r/s0@html]# matchpathcon -V /var/www/html/*

/var/www/html/file1 has context system_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.

matchpathcon 命令需要使用绝对路径，以便于规则中定义的正规表达式一致。

6.恢复安全上下文

如果文件的 context 不正确，而规则中又定义了可用 fcontext，那么就可以使用 **restorecon** 命令恢复为正确的 context：

引用

[root/sysadm_r/s0@html]# restorecon -v file*

restorecon reset /var/www/html/file1 context

system_u:object_r:samba_share_t:s0->system_u:object_r:httpd_sys_content_t:s0

7.设置安全上下文规则

系统使用 file_t 和 default_t 作为没有定义规则的文件的默认类型：

引用

[root/sysadm_r/s0@html]# cd /test

[root/sysadm_r/s0@test]# restorecon -v *

restorecon reset /test/file1 context

sysadm_u:object_r:httpd_sys_content_t:s0->system_u:object_r:default_t:s0

restorecon reset /test/file2 context

sysadm_u:object_r:httpd_sys_content_t:s0->system_u:object_r:default_t:s0

restorecon reset /test/file3 context

sysadm_u:object_r:httpd_sys_content_t:s0->system_u:object_r:default_t:s0

[root/sysadm_r/s0@test]# ll -Z file*

-rw-r--r-- root root system_u:object_r:default_t:s0 file1

-rw-r--r-- root root system_u:object_r:default_t:s0 file2

-rw-r--r-- root root system_u:object_r:default_t:s0 file3

但很多时候，我们可能需要自定义规则。（例如，网页文件不放在/var/www/html 目录下，而是放在/test 目录下）那么，我们就应该把/test 目录下的文件定义为 httpd_sys_content_t 类型。

可以通过 **semanage fcontext** 命令来进行：

[root/secadm_r/s0@www]# semanage fcontext -a -t 'httpd_sys_content_t' '/test(/.*)?'

※ 再次强调，安全策略的调整，必须使用 secadm 用户（或拥有 secadm_r 角色）来进行！不再重复了。

该命令的结果，会保存在/etc/selinux/mls/contexts/files/file_contexts.local 文件：

引用

[root/secadm_r/s0@www]# cat /etc/selinux/mls/contexts/files/file_contexts.local

This file is auto-generated by libsemanage

Please use the semanage command to make changes

/test(/.*)? system_u:object_r:httpd_sys_content_t:s0

默认文件 context 规则在/etc/selinux/mls/contexts/files/file_contexts 文件。但这两个文件（包括 contexts 目录中的文件）都不应该手动修改，而且改了也没用！应该使用 semanage，或基于 libsemanage 库开发的工具来设置，这文件仅是一个易于检索的输出而已。

有了这个 fcontext 规则，就可以使用 **restorecon** 命令修复 context：

引用

[root/sysadm_r/s0@test]# restorecon -v file*

restorecon reset /test/file1 context

system_u:object_r:default_t:s0->system_u:object_r:httpd_sys_content_t:s0

restorecon reset /test/file2 context

system_u:object_r:default_t:s0->system_u:object_r:httpd_sys_content_t:s0

restorecon reset /test/file3 context

system_u:object_r:default_t:s0->system_u:object_r:httpd_sys_content_t:s0

删除规则，就是把上面 semanage fcontext 命令中的-a （表示 add），改为-d （表示 del）：

引用

[root/secadm_r/s0@www]# semanage fcontext -d -t 'httpd_sys_content_t' '/test(/.*)?'

[root/secadm_r/s0@www]# cat /etc/selinux/mls/contexts/files/file_contexts.local

[root/secadm_r/s0@www]# cd /test

[root/secadm_r/s0@test]# restorecon -v *

restorecon reset /test/file1 context

system_u:object_r:httpd_sys_content_t:s0->system_u:object_r:default_t:s0

restorecon reset /test/file2 context

system_u:object_r:httpd_sys_content_t:s0->system_u:object_r:default_t:s0

restorecon reset /test/file3 context

system_u:object_r:httpd_sys_content_t:s0->system_u:object_r:default_t:s0

更多的参数，请看 man 或 semanage --help。

例如，这里没有定义 SL 级别，可通过**-r** 参数定义，则有：

引用

[root/secadm_r/s0@test]# semanage fcontext -a -t 'httpd_sys_content_t' -r 's0-s15:c0.c1023' '/test(/.*)?'

[root/secadm_r/s0@test]# semanage fcontext -l|grep '/test(/.*)?'

/test(/.*)? all files system_u:object_r:httpd_sys_content_t:s0-s15:c0.c1023

8.挂载其他文件系统

除规则定义外，系统使用**file_t** 和 **default_t** 两种作为默认类型，它们的差别主要是作用域不同。具体请见：The file_t and default_t Types

并不是所有文件系统都支持SELinux context 的。因为SELinux 的context 实际是存放在磁盘文件的xattr 属性节点上，而很多文件系统并不支持该属性。例如：vfat、NTFS、NFS、CIFS（即Samba）等。

所以，系统提供了一些参数，在挂载时赋予特定的context。例如，默认NFS会使用nfs_t 类别，而使用**-o context** 参数，可赋予httpd_sys_content_t 类别，使得其可以被httpd_t 访问：

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

但这个属性实际上并不会存入NFS对应的磁盘上的，每次挂载时定义而已。具体可见：Context Mounts 和 Mounting an NFS File System

有些时候，我们会对NFS 进行多层挂载（子目录下）。而上面使用的参数，在这情况下会报错，例如：

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

错误为：

引用

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

应采用 **-o nosharecache,context** 参数，如：

引用

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

这样，NFS 提供的/export 下面的/web 和/database 目录可被分别挂载到不同的目录下，而且拥有不同的安全上下文。

※ 这时也可以把同一个子目录挂载多次，并且采用不同的 context。但是，千万不要这样做。这会产生覆盖挂载，文件的 context 会混乱。

若要实现持久化的挂载，可把配置写入/etc/fstab 中：

引用

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

具体请见：Multiple NFS Mounts

还有一种情况，挂载的文件系统是支持该扩展属性的。但其本身没有使用，或自带的扩展属性不安全（外来设备）。这时，可采用-o defcontext参数，例如：

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

context参数时，安全上下文不会写入磁盘，除非采用相同的参数来挂载，否则不能保持；而defcontext，安全属性时写入磁盘的，可以保持的。context参数用于挂载的文件系统是不支持扩展安全属性的，例如NFS；defcontext参数则可用于挂载支持安全属性的文件系统，例如ext3/4。具体可以参考：Changing the Default Context

※ 还有一点要提醒的是，MLS 策略下，对外部介质的访问更严格。在挂载U盘等设备时，若不配置安全规则，是禁止访问的。在数据可信的情况下，可临时把模式设为Permissive，以便读取。

7. 配置布尔值和属性

适当的给文件赋予正确的安全上下文，可让系统服务在规则允许的情况下顺利的运行。系统 SELinux policy 为自带的服务定义了许多规则（rules）和 User、Port 等属性（elements），也把多种规则组合起来定义为 Booleans（布尔值）。修改规则需要编辑策略源文件（policy source）为新的模块，然后重新加入 SELinux 核心中，工作复杂而且耗时。因此，系统提供了在无需进行这些编译工作的情况下，实时修改布尔值和属性，使之满足应用服务需要的功能。

七、配置布尔值和属性

1.系统自带规则和属性

可以使用 **seinfo** 命令查询系统 SELinux policy 所提供各属性值的情况：

引用

[root/secadm_r/s0@~]# seinfo

Statistics for policy file: /etc/selinux/mls/policy/policy.21

Policy Version & Type: v.21 (binary, MLS)

Classes:	61	Permissions:	220
Types:	2327	Attributes:	163
Users:	7	Roles:	8
Booleans:	116	Cond. Expr.:	138
Sensitivities:	16	Categories:	1024
Allow:	233466	Neverallow:	0

Auditallow: 26 Dontaudit: 119374

Role allow: 10 Role trans: 131

Type_trans: 3857 Type_change: 56

Type_member: 8 Range_trans: 203

Constraints: 185 Validatetrans: 7

Fs_use: 19 Genfscon: 75

Portcon: 322 Netifcon: 1

Nodecon: 8 Initial SIDs: 27

RFSOS4 提供图形工具来查询policy 内容，而**seinfo** 等命令由setools 包提供，需要另行安装，请见[原]SELinux 简明学习指南——运行服务 一文。

2.布尔值

布尔值（Booleans）可理解为是把一组规则（rules）合并起来，定义一个名称，以便在运行时可修改，实时生效，以满足某些许可的要求。它不需要任何SELinux 规则编写的知识，只需简单的启动（On）或关闭（Off）即可。

使用**getsebool** 命令查看当前Booleans状态：

引用

[root/secadm_r/s0@~]# getsebool -a|more

allow_console_login --> off

allow_cvs_read_shadow --> off

.....

RHEL 6 版本的 semanage boolean -l 命令提供每个 Booleans 值的状态及意思，但 RFSOS4 版本不支持。

使用 **semanage** 命令查看系统自带的 Booleans：

引用

#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

例如：普通用户默认为 user_t 身份，是没有权限执行 dmesg 命令的：

引用

[newuser@localhost ~]$ id

uid=524(newuser) gid=524(newuser) groups=524(newuser) context=user_u:user_r:user_t:s0

[newuser@localhost ~]$ dmesg

klogctl: 权限不够

修改 Booleans 值：

引用

[root/secadm_r/s0@~]# getsebool user_dmesg

user_dmesg --> off

[root/secadm_r/s0@~]# setsebool user_dmesg on

[root/secadm_r/s0@~]# getsebool user_dmesg

user_dmesg --> on

这样，newuser 用户就可以运行 dmesg 命令了：

引用

[newuser@localhost ~]$ dmesg|head -n1

Linux version 2.6.18-194.1.AXS3 (packager@asianux.com) (gcc version 4.1.2 20080704 (Asianux 3.0 4.1.2-48)) #1 SMP Fri May 7 10:03:53 CST 2010

不过，这只是临时修改，若要写入配置文件，需加**-P** 参数：

引用

[root/secadm_r/s0@~]# setsebool -P user_dmesg on

关闭是 off 。更多的 Booleans 设置，如：allow_console_login、httpd_can_network_connect_db、httpd_enable_homedirs 等，请自行查询。

3.配置 SELinux 属性

类似的，SELinux 也定义的一些属性（elements）用于辅助策略匹配的。通过 **semanage** 命令可在不修改或重编策略源文件（policy sources）的情况下，对 SELinux 定义的属性进行配置。

1）首选参数

有几个参数作为首选参数，是必须择其一的：

引用

-l ：可查询已有定义记录（record）；

-a ：添加一条新记录；

-d：删除一套记录；

-m：修改已有记录；

2）login 属性

该属性用于定义等Unix 用户登录系统后，所得到的对应SELinux User，以及context标记，在登陆系统部分已经用过，例如：

引用

[root/secadm_r/s0@~]# semanage login -l

Login Name SELinux User MLS/MCS Range

__default__ user_u s0

auditadm auditadm_u s0-s15:c0.c1023

root root s0-s15:c0.c1023

secadm secadm_u s0-s15:c0.c1023

sysadm sysadm_u s0-s15:c0.c1023

system_u system_u s0-s15:c0.c1023

3）user 属性

该属性定义的是，SELinux User 对应的 Unix User、MLS/MCS 范围，以及其可扮演的角色。

引用

[root/secadm_r/s0@~]# semanage user -l

Labeling MLS/ MLS/

SELinux User Prefix MCS Level MCS Range SELinux Roles

auditadm_u auditadm s0 s0-s15:c0.c1023 system_r auditadm_r

root sysadm s0 s0-s15:c0.c1023 system_r sysadm_r staff_r secadm_r auditadm_r

secadm_u secadm s0 s0-s15:c0.c1023 system_r secadm_r

staff_u staff s0 s0-s15:c0.c1023 sysadm_r staff_r secadm_r auditadm_r

sysadm_u sysadm s0 s0-s15:c0.c1023 system_r sysadm_r staff_r

system_u user s0 s0-s15:c0.c1023 system_r

user_u user s0 s0 user_r

例如，当 sysadm 在本地终端登陆，是 sysadm_u:sysadm_r:sysadm_t；而通过 ssh 登陆时，是 sysadm_u:staff_r:staff_t ，secadm 等用户是不能扮演 staff_r 角色的。

4）port 属性

这主要是定义了一些针对某个服务的类型（Type）所使用的端口。除已定义的属性端口外，该服务就不能使用额外的端口：

引用

[root/secadm_r/s0@~]# semanage port -l|grep -w http_port_t

http_port_t tcp 80, 443, 488, 8008, 8009, 8443

5）interface 属性

对应接口对象对应的 context：

引用

[root/secadm_r/s0@~]# semanage interface -l

SELinux Interface Context

lo system_u:object_r:lo_netif_t:s0-s15:c0.c1023

6）fcontext 属性

我们在前面处理安全上下文部分已经提过 ：

引用

[root/secadm_r/s0@test]# semanage fcontext -l|grep '/test(/.*)?'

/test(/.*)? all files system_u:object_r:httpd_sys_content_t:s0-s15:c0.c1023

7）translation 属性

这主要是方便使用 MLS/MCS 模型时，把 SL 级别翻译为人性化、易理解的信息：

引用

[root/secadm_r/s0@~]# semanage translation -l|grep SystemLow-SystemHigh

s0-s15:c0.c1023 SystemLow-SystemHigh

每个属性针对不同对象，可使用的次级参数是不同的。具体见 man，或用--help 查询。

4.补充说明

1）semange user 中 MLS/MCS Range 设定值

使用 **semanage user** 创建 SELinux User 的命令通常是这样：

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s0-s5:c1.c20 -L s0 chuzhang_u

其中，-R 是 SELinux Roles，-P 是 SELinux Prefix，-r 是 MLS/MCS Security Range，-L 是 Default SELinux Level for SELinux use。这在 **man semanage** 中有说明，大多不难理解。但-r 和 -L 两个值的设定，对 **semanage login** 的-r 参数是有直接影响的，其值不能超过 **semanage user** 中定义的某个 SELinux User 的 MCS Range 范围。例如，以上面的设定情况，则下面的操作会失败：

引用

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s5:c30 chuzhang

libsemanage.validate_handler: MLS range s5:c30 for Unix user chuzhang exceeds allowed range s0-s5:c1.c20 for SELinux user chuzhang_u

libsemanage.validate_handler: seuser mapping [chuzhang -> (chuzhang_u, s5:c30)] is invalid

libsemanage.dbase_llist_iterate: could not iterate over records

/usr/sbin/semanage: 无法为 chuzhang 添加登录映射

只要在 SELinux User 的 MCS Range 许可范围内设定都是可以的：

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s5:c20 chuzhang

以 chuzhang 用户登录系统，可获得的 context 为：

引用

[chuzhang@localhost ~]$ id -Z

chuzhang_u:user_r:user_t:s5:c20

2）semanage user 中MLS/MCS Level 设定值

-L 是 Default SELinux Level for SELinux use，默认值为 s0。例如：

[root/secadm_r/s0@~]# semanage user -m -R user_r -P user -r s0-s5:c1.c20 -L s0 chuzhang_u

[root/secadm_r/s0@~]# semanage login -m -s chuzhang_u -r s0-s5:c1.c10 chuzhang

则有：

引用

[chuzhang@localhost ~]$ id -Z

chuzhang_u:user_r:user_t:s0-s5:c1.c10

若设置为非 s0，则会影响 **semanage login** 的-r 参数的设置。例如：

引用

[root/secadm_r/s0@~]# semanage user -m -R user_r -P user -r s0-s5:c1.c20 -L s1:c10 chuzhang_u

[root/secadm_r/s0@~]# semanage login -m -s chuzhang_u -r s0-s5:c1.c10 chuzhang

这里把 MLS/MCS Level 默认值改为 s1:c10，而 **semanage login** 命令是没变的，但以 chuzhang 登录获得的 context 是不同的：

引用

[chuzhang@localhost ~]$ id -Z

chuzhang_u:user_r:user_t:s1:c10-s5:c1.c10

可见，它会使用 **semanage user** 的 MLS/MCS Level 替换 **semanage login** 中-r 参数的 s0 部分。

如果不指定 semanage login -r 的 s0 部分呢？（这例子与补充说明 1 中的第二个例子是类似的）

引用

[root/secadm_r/s0@~]# semanage login -m -s chuzhang_u -r s5:c1.c10 chuzhang

那将只有一个安全等级，而不能在一个范围内变动：

引用

[chuzhang@localhost ~]$ id -Z

chuzhang_u:user_r:user_t:s5:c1.c10

另外，执行 **semanage user** 时，是不会检查-L 是否在-r 的范围内，但使用 **semanage login** 时会出问题的：

引用

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s2:c0.c10-s5:c1.c20 -L s6:c30

chuzhang_u

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s2:c0 chuzhang

libsemanage.validate_handler: MLS range s2:c0 for Unix user chuzhang exceeds allowed range

s2:c0.c10-s5:c1.c20 for SELinux user chuzhang_u

libsemanage.validate_handler: seuser mapping [chuzhang -> (chuzhang_u, s2:c0)] is invalid

libsemanage.dbase_llist_iterate: could not iterate over records

/usr/sbin/semanage: 无法为 chuzhang 添加登录映射

（-L 的 s5:c10 在 -r 的 s2:c0.c10-s5:c1.c20 之外）

也不要随意改变-r 和-L 的范围：

引用

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s2:c0.c10-s5:c1.c20 -L s2:c0 chuzhang_u

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s2:c0 chuzhang

libsemanage.validate_handler: MLS range s2:c0 for Unix user chuzhang exceeds allowed range

s2:c0.c10-s5:c1.c20 for SELinux user chuzhang_u

libsemanage.validate_handler: seuser mapping [chuzhang -> (chuzhang_u, s2:c0)] is invalid

libsemanage.dbase_llist_iterate: could not iterate over records

/usr/sbin/semanage: 无法为 chuzhang 添加登录映射

（-r 不是从 s0 开始）

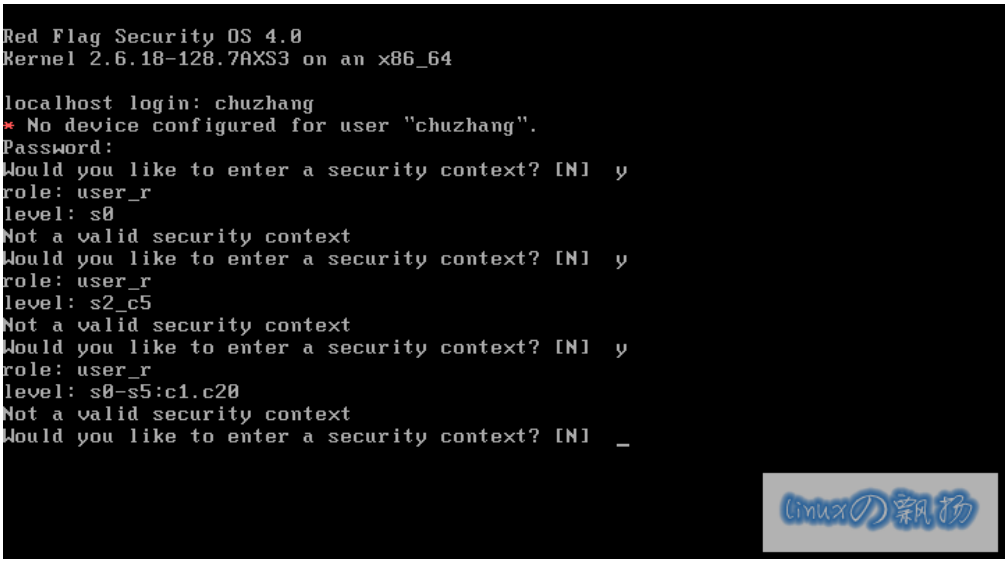
这样的设定可以通过：

引用

[root/secadm_r/s0@~]# semanage user -m -R user_r -P user -r s0-s5:c1.c20 -L s6:c30 chuzhang_u

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s2:c5 chuzhang

但由于 MLS/MCS Level 超出范围，chuzhang 用户登录时会提示下面的信息，context 非法，不能登入系统：



一句话，给 semanage user 设定-L 参数，可能会引发很多意想不到的问题！建议，设置时，保留其为默认值 s0。下面这些都没问题：

引用

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s0-s5:c1.c20 chuzhang_u

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s2:c3 chuzhang

[root/secadm_r/s0@~]# semanage login -m -s chuzhang_u -r s0-s2:c3 chuzhang

3）semanage user 和 semanage login 冲突

如果已经存在 **semanage login** 设置的 SELinux User，在修改 **semanage user** 时可能会报错：

引用

[root/secadm_r/s0@~]# semanage user -m -R user_r -P user -r s2:c0-s5:c1.c20 -L s2:c0 chuzhang_u

libsemanage.validate_handler: MLS range s5:c1.c10 for Unix user chuzhang exceeds allowed range s2:c0-s5:c1.c20 for SELinux user chuzhang_u

libsemanage.validate_handler: seuser mapping [chuzhang -> (chuzhang_u, s5:c1.c10)] is invalid

libsemanage.dbase_llist_iterate: could not iterate over records

/usr/sbin/semanage: 无法修改 SELinux 用户 chuzhang_u

[root/secadm_r/s0@~]# semanage login -l|grep chuzhang

chuzhang chuzhang_u s5:c1.c10

解决办法是，先把 **semanage login** 设置的删掉，再重新设置即可。

引用

[root/secadm_r/s0@~]# semanage login -d chuzhang

[root/secadm_r/s0@~]# semanage user -m -R user_r -P user -r s2:c0-s5:c1.c20 -L s2:c0 chuzhang_u

类似的，若要删除 SELinux User，请先执行 **semanage login -d user** 删除用户，然后再执行 **semanage user -d** 命令。（若有 Unix User 依赖，还必须先执行 **userdel -r user** 命令）

以上描述的配置，都是在无需重新编译策略模块的情况下，配合应用服务要求，可进行的实时改动。但若要修改策略规则，就必须编写 TE 策略文件，并生成 PP 模块加入 SELinux 核心中才行。

8. 审计日志

系统会对主体访问客体的动作是否允许进行判断，当拒绝发生时，除规则动作定义为 dontaudit 外，会生成AVC消息记录在审计日志/var/log/audit/audit.log 文件。当允许发生时，只有匹配规则动作为auditallow的消息才写入日志。这些消息可以用来监控系统，管理和开发策略模块。

八、审计日志

1.查看日志

用户 auditadm 或 auditadm_r 角色是专门的审计管理员，secadm 也能访问审计日志，但 sysadm 是没有权限的。审计日志位于/var/log/audit 目录下，当前日志为 audit.log。注意，它的安全级别是最高的 s15。

引用

[root/auditadm_r/s0@~]# ll -Z /var/log/audit/audit.log

-rw------- root root system_u:object_r:auditd_log_t:s15:c0.c1023 /var/log/audit/audit.log

例如，以 sysadm 访问审计日志为例：

引用

[root/sysadm_r/s0@~]# ll /var/log/audit

ls: /var/log/audit: 权限不够

那么，在审计日志中就会出现：

引用

type=AVC msg=audit(1340382307.771:270): avc: denied { getattr } for pid=4288 comm="ls" path="/var/log/audit" dev=sda1 ino=1900937 scontext=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023 tcontext=system_u:object_r:auditd_log_t:s15:c0.c1023 tclass=dir

type=SYSCALL msg=audit(1340382307.771:270): arch=c000003e syscall=6 success=no exit=-13 a0=7fffee3b2b66 a1=13e58808 a2=13e58808 a3=3395751a30 items=0 ppid=4252 pid=4288 auid=512 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts3 ses=32 comm="ls" exe="/bin/ls" subj=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023 key=(null)

AVC 消息几乎每个字段都是以“名称=值”的形式出现的，如 pid=4288。

2.日志字段

AVC 消息分下面 6 个字段：

引用

1）type：审核守护进程产生的消息可以有多种类型，消息的类型是由“type=”的前缀和消息类型共同标识的，这个例子中的前缀是 AVC，标识出这条消息是一条 AVC 消息，其它消息类型包括 USER_AUTH, LOGIN, SYSCALL 和 PATH；

2）msg：审核框架在所有审核消息的开始位置加上一个消息头，包括一个时间戳和由分号隔开的序列号，在这个消息中的时间戳是 1135098961.471，它是从基准时间以来的以纳秒为单位的时长，在这个例子中序列号是 1770，它用来标识由相同事件产生的多个相关的审核消息，例如，一个事件可以产生两个系统调用和 AVC 审核消息，这些消息的序列号是相同的；

3）avc：这个字段不再以名称/值的形式了，它标识出审核消息是来自允许访问还是拒绝访问，以及允许或拒绝的许可，这里可以有一个或多个许可，都来自一个客体类别，关键词 denied 表示这个消息来自访问被拒绝，允许访问是用 granted 表示的；

4）scontext：主体的 context；

5）tcontext：客体的 context；

6）tclass：客体的客体类别，允许或拒绝的许可来自为这个客体类别定义的访问向量（AV）。

3.辅助信息

avc 字段提供关于访问被允许或拒绝的详细消息，这通常是特定的客体类别。例如，与文件有关的客体类

别的审核消息通常包括客体的 inode 号，与网络有关的客体类别的审核消息通常包括 IP 地址和端口号。

在上面的例子中，就有 4 个字段：

引用

1）pid：尝试访问的进程的标识符，它在一个应用程序产生多个调用时特别有用，可明确标识子进程；

2）comm：与进程关联的可执行文件名，但不包含路径信息；

3）path：执行命令的实际路径；

4）dev 和 ino：与目标关联的与文件有关的客体的设备（dev）和 inode 号（ino），如果在审核消息中没有完整的路径，可以使用它们来标识客体；

此外，SYSCALL 与 AVC 常成对出现，SYSCALL 中 success 字段表示操作是否成功。no 是失败，yes 成功。成功的原因可能是 AVC 规则为 denied，但运行模式为 Permissive，所以，实际操作是完成了。

4.服务及命令

审计功能依赖于 auditd 服务，其默认是启动的：

引用

[root/auditadm_r/s0@~]# service auditd status

auditd (pid 2336) 正在运行...

通过 **aureport** 命令可查看汇总的审计信息：

引用

[root/auditadm_r/s0@~]# aureport -a

NOTE - using built-in logs: /var/log/audit/audit.log

AVC Report

==

date time comm subj syscall class permission obj event

==

1. 2012 年 06 月 13 日 16:56:01 modprobe system_u:system_r:insmod_t:s15:c0.c1023 17

5 capability sys_nice system_u:system_r:insmod_t:s15:c0.c1023 denied 41

2. 2012 年 06 月 13 日 16:56:01 modprobe system_u:system_r:insmod_t:s15:c0.c1023 17

5 capability sys_nice system_u:system_r:insmod_t:s15:c0.c1023 denied 42

通过 **ausearch** 命令可对审计日志进行检索：

引用

[root/auditadm_r/s0@~]# ausearch -m avc

NOTE - using built-in logs: /var/log/audit/audit.log

---

time->Wed Jun 13 16:56:01 2012

type=SYSCALL msg=audit(1339577761.740:41): arch=c000003e syscall=175 success=y

es exit=0 a0=19038e90 a1=bf58 a2=19029440 a3=19029440 items=0 ppid=2383 pid=23

84 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty

=(none) ses=4294967295 comm="modprobe" exe="/sbin/modprobe" subj=system_u:syst

em_r:insmod_t:s15:c0.c1023 key=(null)

type=AVC msg=audit(1339577761.740:41): avc: denied { sys_nice } for pid=238

4 comm="modprobe" capability=23 scontext=system_u:system_r:insmod_t:s15:c0.c10

23 tcontext=system_u:system_r:insmod_t:s15:c0.c1023 tclass=capability

还有一些有用的排除参数：

引用

ausearch -m avc -ts today <-- 今天被拒绝的日志

ausearch -m avc -ts recent <-- 仅 10 分钟被拒绝的日志

ausearch -m avc -c httpd <-- httpd 命令产生的拒绝日志

通过 **audit2why** 命令可从审计日志得到一些解决建议：

引用

[root/auditadm_r/s0@~]# audit2why < /var/log/audit/audit.log

type=AVC msg=audit(1340382307.771:270): avc: denied { getattr } for pid=4288 comm="ls"

path="/var/log/audit" dev=sda1 ino=1900937 scontext=sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

tcontext=system_u:object_r:auditd_log_t:s15:c0.c1023 tclass=dir

Was caused by:

Constraint violation.

Check policy/constraints.

Typically, you just need to add a type attribute to the domain to satisfy the constraint.

RHEL 等系统还由setroubleshoot、setroubleshoot-server 包提供了setroubleshoot 服务，sealert 命令等更详细的信息描述，具体可参考：Searching For and Viewing Denials。但RFSOS 4 没提供这功能。

有了这些审计日志，我们就可通过**audit2allow** 命令创建简单的允许规则模块，以把那些被拒绝的操作顺利通过。

9. 添加允许规则

从前面的描述可知，系统已定义了许多规则，也允许通过属性（Elements）和布尔值（Booleans）对匹配规则进行微调。但实际应用需求是复杂的，很多时候，我们还需要添加额外的规则。可是，附加的规则必须以模块的形式加载的SELinux核心中，而规则的编写特别麻烦、复杂，需要有很系统的学习。为此，系统提供了**audit2allow** 工具，它能通过分析审计日志信息，把原被禁止的操作转为允许的规则，并生成规则模块，以便使用。

RFSOS4 的SELinux 安全策略是基于referrence policy，采用基础模块加附加模块的形式。例如，我们在[原]SELinux 简明学习指南——登陆系统一文的“优化bash显示”部分添加的PP模块，就是附加模块。这里，就以该例子来说明“如何通过**audit2allow** 生成新的允许规则模块”。

九、添加允许规则

1.SELinux 策略模块

系统当前使用的PP 模块，存放在/etc/selinux/mls/modules/active/modules/目录：

引用

```
[root/secadm_r/s0@~]# ll /etc/selinux/mls/modules/active/modules/apache.pp
-rw------- 1 root root 588975 06-27 15:30 /etc/selinux/mls/modules/active/modules/apache.pp
[root/secadm_r/s0@~]# ll /etc/selinux/mls/modules/active/modules/*.pp|wc -l
160
```

如要你要修改规则，就需要修改源码提供的.te 文件，编译为.pp 模块后，重新加入 policy 策略中。但是，TE 文件是很复杂的（后面会介绍），而规则又是只有明确允许才可通过，否则就是禁止的。如果我们只想创建一些许可规则，**audit2allow** 命令可帮到我们。

这里，我们就以上面提到的 PP 模块为例，使用 **audit2allow** 命令创建自己的 PP 模块。该模块的作用是：默认情况下，只有 sysadm_r 角色用户可对/root 目录读写；通过这模块，使得 secadm_r 角色用户也对/root 目录读写。

2.准备工作

首先，我们还原为系统的默认状态，用 **semodule** 命令把之前加入的 PP 模块移除：

引用

[root/secadm_r/s0@~]# semodule -l|grep enter

enter_root 1.0

[root/secadm_r/s0@~]# semodule -r enter_root

这样，secadm 用户（其角色为 secadm_r）就不能访问/root 目录了：

引用

[root/secadm_r/s0@~]# touch /root/test

touch: 无法触碰 “/root/test”: 权限不够

其次，查看规则对应的类型域：

引用

[root/secadm_r/s0@~]# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

context=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023

[root/secadm_r/s0@~]# ll -Zd /root/

drwxr-x--- root root root:object_r:sysadm_home_dir_t:s0-s15:c0.c1023 /root/

可见，secadm_r 所处的类型域是 secadm_t，/root 目录的类型域是 sysadm_home_dir_t 。让 secadm_r 角色用户也对/root 目录读写，即让 secadm_t 拥有对 sysadm_home_dir_t 的一些执行权限。

系统默认规则如下：

引用

[root/secadm_r/s0@~]# sesearch -s secadm_t -t sysadm_home_dir_t --allow

Found 7 av rules:

allow secadm_t sysadm_home_dir_t : file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : dir { ioctl read getattr lock relabelfrom relabelto search };

allow secadm_t sysadm_home_dir_t : lnk_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : chr_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : blk_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : sock_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : fifo_file { getattr relabelfrom relabelto };

3.创建新模块

使用 **audit2allow** 命令创建新模块的步骤通常如下：

1）切换到Permissive 模式

引用

[root/secadm_r/s0@~]# setenforce 0

[root/secadm_r/s0@~]# getenforce

Permissive

※ 这步是必须的，很重要，原因见补充说明。

2）清空审计日志

audit2allow 命令是根据审计日志来生成 TE 文件和 PP 模块的，为避免其他操作引起的误会，建议下一步前，把审计日志备份后，清空。

引用

[root/secadm_r/s0@~]# echo > /var/log/audit/audit.log

[root/secadm_r/s0@~]# ll /var/log/audit/audit.log

-rw------- 1 root root 1 06-26 13:34 /var/log/audit/audit.log

3）执行那些原来被禁止，而现在要允许的操作

例如，以 secadm 用户执行：

引用

[root/secadm_r/s0@~]# cd /root

[root/secadm_r/s0@~]# echo 1111 > test

[root/secadm_r/s0@~]# ll -Z test

-rw-r--r-- root root secadm_u:object_r:sysadm_home_dir_t:s0 test

[root/secadm_r/s0@~]# rm test

rm：是否删除 一般文件 “test”? y

[root/secadm_r/s0@~]# mkdir test

[root/secadm_r/s0@~]# rmdir test

一句话，就是要模拟那些操作，使得审计日志中可记录这些操作的详细信息。执行的命令应该尽可能全面、完整。例如，上面的操作就会产生类似的日志：

引用

type=AVC msg=audit(1340468838.415:1658): avc: denied { create } for pid=7763 comm="mkdir" name="test" scontext=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023 tcontext=secadm_u:object_r:sysadm_home_dir_t:s0 tclass=dir

type=SYSCALL msg=audit(1340468838.415:1658): arch=c000003e syscall=83 success=yes exit=0 a0=7fff697f9b5d a1=1ff a2=3395750114 a3=3395751a30 items=0 ppid=7162 pid=7763 auid=513 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=234 comm="mkdir" exe="/bin/mkdir" subj=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023 key=(null)

对比以前的日志，可发现操作是 denied 的，但 success=yes ，这就是 Permissive 模式。

*4）使用 **audit2allow** 命令创建 PP 模块*

引用

[root/secadm_r/s0@~]# cd /tmp

[root/secadm_r/s0@tmp]# grep 'secadm_t' /var/log/audit/audit.log|audit2allow -M mytest

****************** IMPORTANT **********************

To make this policy package active, execute:

semodule -i mytest.pp

※ 利用 **grep** 可过滤审计日志，以便生成特定的许可模块。若使用 **audit2allow -a -M mytest** 方式可直接对整个审计日志分析。（可能会包含一些不必要的许可规则）

TE 文件内容如下：

引用

```
[root/secadm_r/s0@~]# cat /tmp/mytest.te

module mytest 1.0;

require {
        type sysadm_home_dir_t;
        type secadm_t;
        class dir { write remove_name create add_name rmdir };
        class file { write create unlink };
}

#============= secadm_t ==============
allow secadm_t sysadm_home_dir_t:dir { write remove_name create add_name rmdir };
allow secadm_t sysadm_home_dir_t:file { write create unlink };
```

5）加载 PP 模块，并测试

执行：

引用

```
[root/secadm_r/s0@tmp]# semodule -i mytest.pp
[root/secadm_r/s0@tmp]# semodule -l|grep mytest
mytest  1.0
[root/secadm_r/s0@tmp]# ll /etc/selinux/mls/modules/active/modules/mytest.pp
-rw------- 1 root root 1113 06-24 00:41 /etc/selinux/mls/modules/active/modules/mytest.pp
```

恢复到 Enforcing 模式，重复原来被禁止的操作，进行测试：

引用

```
[root/secadm_r/s0@tmp]# setenforce 1
[root/secadm_r/s0@tmp]# cd /root/
[root/secadm_r/s0@~]# cd /root/
[root/secadm_r/s0@~]# echo 1111 > test
[root/secadm_r/s0@~]# rm test
rm：是否删除 一般文件 “test”? y
```

[root/secadm_r/s0@~]# mkdir test

[root/secadm_r/s0@~]# rmdir test

操作成功！查看审计日志，没有任何被禁止的记录，证明由 mytest.pp 模块加入的新规则已经生效。

使用 **sesearch** 命令查看规则：

引用

[root/secadm_r/s0@tmp]# sesearch -s secadm_t -t sysadm_home_dir_t --allow

Found 7 av rules:

allow secadm_t sysadm_home_dir_t : file { write read getattr relabelfrom relabelto unlink };

allow secadm_t sysadm_home_dir_t : dir { ioctl read write create getattr lock relabelfrom relabelto add_name remove_name search rmdir };

allow secadm_t sysadm_home_dir_t : lnk_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : chr_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : blk_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : sock_file { getattr relabelfrom relabelto };

allow secadm_t sysadm_home_dir_t : fifo_file { getattr relabelfrom relabelto };

以上就是使用 audit2allow 命令创建允许规则模块的基本步骤。

4.问题排查

不过，这还没有结束！？因为，我们新创建的规则不能与已有的规则冲突，而应该兼容。

1)工作没完成

我们以 secadm 重新登陆一次终端：

引用

[secadm@localhost ~]$ su -

口令：

-bash: /root/.bash_profile: 权限不够

-bash-3.2#

怎么还是读不了/root 目录的文件？看看/root/.bash_profile 的 context：

引用

[root/secadm_r/s0@~]# # ll -Z /root/.bash_profile

-rw-r--r-- root root root:object_r:sysadm_home_t:s0 /root/.bash_profile

类型是 sysadm_home_t，而不是/root 目录所在的 sysadm_home_dir_t。用 **restorecon** 命令恢复一下：

引用

[root/secadm_r/s0@~]# cd /root/

[root/secadm_r/s0@~]# echo 1111 > test

[root/secadm_r/s0@~]# ll -Z test

-rw-r--r-- root root secadm_u:object_r:sysadm_home_dir_t:s0 test

[root/secadm_r/s0@~]# restorecon -v test

restorecon reset /root/test context

secadm_u:object_r:sysadm_home_dir_t:s0->root:object_r:sysadm_home_t:s0

原来应该使用 sysadm_home_t 类型。

也就是说，在/root 目录下，secadm 创建的 test 文件与 sysadm 创建的文件 context 不同。看看 sysadm 用户创建的文件：

引用

[root/sysadm_r/s0@~]# echo '1111' > test1

[root/sysadm_r/s0@~]# ll -Z test1

-rw-r--r-- root root sysadm_u:object_r:sysadm_home_t:s0 test1

[root/sysadm_r/s0@~]# rm test1

rm：是否删除 一般空文件 “test1”? y

[root/sysadm_r/s0@~]# mkdir test1

[root/sysadm_r/s0@~]# ll -Zd test1

drwxr-xr-x root root sysadm_u:object_r:sysadm_home_t:s0 test1

[root/sysadm_r/s0@~]# rmdir test1

类型都是 sysadm_home_t 的。但 secadm 没有读取 sysadm_home_t 类型文件的权限：

引用

[root/secadm_r/s0@~]# ll -Z test

-rw-r--r-- root root root:object_r:sysadm_home_t:s0 test

[root/secadm_r/s0@~]# cat test

cat: test: 权限不够

2）进一步测试

若想让 secadm 可读取 sysadm_home_t 类型的文件，那就再重复一次 audit2allow 的步骤吧：

引用

[root/secadm_r/s0@~]# setenforce 0

[root/secadm_r/s0@~]# cd /root/

[root/secadm_r/s0@~]# rm -f test

[root/secadm_r/s0@~]# echo '1111' >test

[root/secadm_r/s0@~]# ll -Z test

-rw-r--r-- root root secadm_u:object_r:sysadm_home_dir_t:s0 test

[root/secadm_r/s0@~]# restorecon -v test

restorecon reset /root/test context

secadm_u:object_r:sysadm_home_dir_t:s0->root:object_r:sysadm_home_t:s0

[root/secadm_r/s0@~]# cat test

1111

[root/secadm_r/s0@~]# rm -f test

[root/secadm_r/s0@~]#

[root/secadm_r/s0@~]# mkdir testdir

[root/secadm_r/s0@~]# ll -Zd testdir

drwxr-xr-x root root secadm_u:object_r:sysadm_home_dir_t:s0 testdir

[root/secadm_r/s0@~]# restorecon -v testdir/

restorecon reset /root/testdir context

secadm_u:object_r:sysadm_home_dir_t:s0->root:object_r:sysadm_home_t:s0

[root/secadm_r/s0@~]# ll -Zd testdir

drwxr-xr-x root root root:object_r:sysadm_home_t:s0 testdir

[root/secadm_r/s0@~]# rmdir testdir

生成规则模块：

引用

```
[root/secadm_r/s0@~]# cd /tmp
[root/secadm_r/s0@tmp]# grep 'secadm_t' /var/log/audit/audit.log|audit2allow -M mytest
```

查看 TE 文件：

引用

```
[root/secadm_r/s0@tmp]# cat mytest.te

module mytest 1.0;

require {
        type sysadm_home_dir_t;
        type secadm_t;
        type sysadm_home_t;
        class dir { write rmdir remove_name create add_name };
        class file { rename write setattr read create unlink };
}

#============= secadm_t ==============
allow secadm_t sysadm_home_dir_t:dir { write remove_name create add_name rmdir };
allow secadm_t sysadm_home_dir_t:file { rename write setattr read create unlink };
allow secadm_t sysadm_home_t:dir rmdir;
allow secadm_t sysadm_home_t:file { write unlink };
```

有发现问题吗？

怎么新添加的规则中的动作只有一部分，那 read 呢？这说明，有些 denied 信息没有输出到日志中。

3）不审计操作

原因是，规则中有 dontaudit 的动作定义，也就是有些操作虽然被阻止，但并不记录到审计日志中。

引用

```
[root/secadm_r/s0@~]# sesearch -s "secadm_t" -t "sysadm_home_t" --audit
Found 7 av rules:
   dontaudit secadm_t sysadm_home_t : file { ioctl read getattr lock };
```

dontaudit secadm_t sysadm_home_t : dir { ioctl read getattr lock search };

dontaudit secadm_t sysadm_home_t : lnk_file getattr ;

dontaudit secadm_t sysadm_home_t : chr_file getattr ;

dontaudit secadm_t sysadm_home_t : blk_file getattr ;

dontaudit secadm_t sysadm_home_t : sock_file getattr ;

dontaudit secadm_t sysadm_home_t : fifo_file getattr ;

在 RFSOS4 上，可通过把基础模块替换为 enableaudit 模块，临时 dontauidit 规则，打开全审计功能：

[root/secadm_r/s0@~]# semodule -b /usr/share/selinux/mls/enableaudit.pp

再次重复上面以 secadm 对 sysadm_home_t 类型文件执行的操作，查看审计日志，就会有：

引用

type=AVC msg=audit(1340953123.475:4424): avc: denied { read } for pid=27480 comm="cat" name="test" dev=sda1 ino=2558904 scontext=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023 tcontext=root:object_r:sysadm_home_t:s0 tclass=file

type=SYSCALL msg=audit(1340953123.475:4424): arch=c000003e syscall=2 success=no exit=-13 a0=7fff899bab61 a1=0 a2=7fff899b9050 a3=3395751a30 items=0 ppid=24800 pid=27480 auid=513 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=276 comm="cat" exe="/bin/cat" subj=secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023 key=(null)

新生成的 TE 文件：

引用

[root/secadm_r/s0@tmp]# cat mytest.te

module mytest 1.0;

require {

type sysadm_home_dir_t;

type secadm_t;

type sysadm_home_t;

class process { siginh noatsecure rlimitinh };

```
        class dir { create write rmdir remove_name add_name };
        class file { rename execute setattr read create write unlink };
}

#============= secadm_t ==============
allow secadm_t sysadm_home_dir_t:dir { write remove_name create add_name rmdir };
allow secadm_t sysadm_home_dir_t:file { rename write setattr read create unlink };
allow secadm_t sysadm_home_t:dir rmdir;
allow secadm_t sysadm_home_t:file { write read unlink };
```

可见，read 动作已经被记录在审计日志中，并通过 audit2allow 生成新的规则模块。

测试：

引用

```
[root/secadm_r/s0@tmp]# semodule -i mytest.pp
[root/secadm_r/s0@tmp]# setenforce 1
[root/secadm_r/s0@tmp]# cd /root/
[root/secadm_r/s0@~]# echo '1111' > test
[root/secadm_r/s0@~]# ll -Z test
-rw-r--r--  root root secadm_u:object_r:sysadm_home_dir_t:s0 test
[root/secadm_r/s0@~]# restorecon -v test
restorecon reset /root/test context
secadm_u:object_r:sysadm_home_dir_t:s0->root:object_r:sysadm_home_t:s0
[root/secadm_r/s0@~]# ll -Z test
-rw-r--r--  root root root:object_r:sysadm_home_t:s0    test
[root/secadm_r/s0@~]# cat test
1111
```

很好！以 secadm 重新登陆系统试试：

引用

```
[secadm@localhost ~]$ su -
口令：
[root/secadm_r/s0@~]#
```

不错嘛，可以读取/root/.bash_profile 文件了。O(∩_∩)O 哈哈~

恢复支持 dontauidit 规则的审计，请执行：

```
[root/secadm_r/s0@~]# semodule -b /usr/share/selinux/mls/base.pp
```

5.补充说明

*1）为什么在使用**audit2allow** 命令创建新模块时，切换到Permissive 模式？*

原因是，创建文件或目录的操作不是只有一个，而是多个连续的操作产生的。例如，touch test 操作，会产生 write->add_name->create 等一系列的顺序行为。若在 Enforcing 模式下进行操作，这些行为都是被禁止的。

换句话说，如果在 Enforcing 下进行 touch test 操作，在 write 行为时就被阻止，后面的行为根本不会发生，也就不会记录到审计日志中，**audit2allow** 命令也无法产生对应的允许规则；而 Permissive 模式则不同，各行为会逐一产生、通过，并记录到审计日志中，**audit2allow** 命令就可用之产生合适的全部允许规则。

2）RHEL 6 版本的情况

a）在 RHEL 6 上的 **sesearch** 命令提供--dontaudit 参数可查询，但 RFSOS4 的版本不支持该参数，只能通过--audit 参数查询。

b）在 RHEL 6 上，可执行下面的命令，打开全审计：

```
# semodule -DB
```

恢复是：

```
# semodule -B
```

详细请见：Possible Causes of Silent Denials

3）手动编译TE 文件

通过**audit2allow** 命令生成的TE 文件是可修改的，这也便于我们自行添加一些新的操作许可。编辑完成后，可使用下面的命令手动创建PP 模块：

引用

[root/secadm_r/s0@tmp]# checkmodule -M -m -o mytest.mod mytest.te

checkmodule: loading policy configuration from mytest.te

checkmodule: policy configuration loaded

checkmodule: writing binary representation (version 6) to mytest.mod

[root/secadm_r/s0@tmp]# semodule_package -o mytest.pp -m mytest.mod

[root@localhost ~]# ll mytest.pp

-rw-r--r-- 1 root root 6016 06-26 17:06 mytest.pp

这会创建一个中转的.mod 文件，然后才能生成 PP 模块。（实际上，semodule -M 参数的动作是完全一样的）

还有一种方法，是完全按照常规创建自定义 PP 模块的步骤进行。这需要使用.if 、.fc、.te 三个文件

引用

[root/secadm_r/s0@tmp]# ll mytest.*

-rw-r--r-- 1 root root   0 06-26 18:15 mytest.fc

-rw-r--r-- 1 root root   0 06-26 18:15 mytest.if

-rw-r--r-- 1 root root 3021 06-26 18:14 mytest.te

[root/secadm_r/s0@tmp]# make -f /usr/share/selinux/devel/Makefile

[root@localhost ~]# ll mytest.pp

-rw-r--r-- 1 root root 12727 06-26 18:15 mytest.pp

三个文件的作用在创建自定义模块部分会讲到，可暂使用空白文件代替。这方式会加入.fc、.if 定义的类型或属性，故创建的 PP 模块比上一种方式创建的要大：

引用

This is done by creating three files: myapp.te, mayapp.fc, and myapp.if. The file myapp.te file will contain all of the policy private to this module, including any types or attributes. The file myapp.fc file will contain the file context labeling statement for this module. Finally, the file myapp.if will contain the interfaces for this module.

不过，**audit2allow** 虽然方便，但却不是万能的。它只能分析审计日志，创建简单的类型间操作的许可模块，而不能与完全自定义模块那样，创建自定义类型、域、接口等。另外，它是创建针对 TE 模型下的类型规则，而对 MLS/MCS 访问限制则无能为力。

10. MLS/MCS 限制

audit2allow 命令可把审计日志中的阻止信息，转换为允许规则，使用方便。不过，它创建的规则只能解决TE 模型中类型访问的问题，而不能处理MLS/MCS 模型的访问限制问题。在LSM（Linux Security Module）框架下，SELinux 若采用mls 安全策略，其访问判断顺序为：UNIX DAC -> SELinux TE rules -> MLS/MCS constraints 。所以，MLS/MCS 限制不能忽视！

十、MLS/MCS 限制

1.概念

MLS（Multi-Level Security ）和MCS（Multi-Category Security ）在[原]SELinux 简明学习指南——安全上下文 部分已做了简单介绍。例如：

引用

s0-s15:c0.c1023

是一个范围的级别，lowlevel - highlevel（低安全级别 - 高安全级别）。每个 level 是一个 sensitivity-category（敏感度-分类）对。灵敏度有着严格的分级，它反应了一个有序的数据灵敏度模型，如政府分类控制中的绝密，机密和无密级。分类是无序的，它反应的是数据划分的需要。基本思路是对于要访问的数据你同时需要足够的灵敏度和正确的分类。

灵敏度有着严格的分级，可以使用等价关系符（<，=，>）进行比较；分类是集合，有包含、被包含的关系。安全级别是它们两者的结合，安全级别是没有分级的，可以使用控制关系运算符（dom，domby，eq，incomp）进行比较。

在 SELinux 策略，使用 sensitivity 语句定义灵敏度：

```
sensitivity s0;
sensitivity s1;
```

dominance 语句指定灵敏度的分级顺序：

```
dominance { s0 s1 s2 s3 }  # s0 表示低，s3 表示高
```

category 语句定义分类和别名（分类是没有高低之分的）：

category c0 alias blue;

category c1 alias red;

level 语句定义安全级别，其规定了灵敏度如何与分类进行关联：

level s0:c0.c2;

level s1:c0.c2,c4;

这里，s0 只与分类 c0，c1 和 c2 关联，s1 与 c0，c1，c2 和 c4 关联，但没有 c3。分类中的点（.）表示一个分类的范围了，而逗号（,）表示一个非连续的分类列表了。

这会影响到我们使用安全上下文（Security Context）时的定义，对于一个有效的安全上下文，高级级别必须优先于低级级别，此外，与灵敏度关联的分类也必须是有效的。假设在上面所定义的安全级别下（level 语句定义），下面的两个安全上下文就是错误的：

引用

user_u:user_r:user_t:s0-s0:c2,c4

（没有 s0:c4 这样的安全级别）

user_u:user_r:user_t:s0:c0-s0:c2

（s0:c2 与 s0:c0 是同级的，也就是说，高级级别没有优于低级级别）

基于多层次访问模型（MLS）采用Bell-La Padula Mandatory Access Model 模型的基本策略是“**下读上写**”，即高等级可读低等级的数据，低等级可向高等级汇报写入。但RFSOS4.0 中有改变，是“**高等级可读低等级（这没变），相等可写可读，其他都是非读非写**”！

正如上面所说，安全级别由dom 等控制关系运算符决定高低，具体见这里。以SL1 访问 SL2，假设为dom 通过，即SL1 比SL2 的等级要高。那么也就是说：

引用

SL1 的敏感度高于或等于 SL2，同时，SL1 的分类包含 SL2 的分类。

那么，在 mls 安全策略下，最后的访问判断顺序为：

引用

UNIX DAC -> SELinux TE rules -> MLS/MCS constraints

所有规则都通过，主体才可成功访问客体！（注意，有些情况是例外的，请见补充说明）

2.例子一

我们以例子来说明 MLS/MCS 的限制问题。**注意，该例子建议不要在生产系统上使用，否则会破坏三权分立的设定，给系统带来危险。**

1）禁止访问的原因

默认情况下，sysadm_r 是不能访问/var/log/audit 目录及以下的文件的：

引用

[root/sysadm_r/s0@~]# ll /var/log/audit

ls: /var/log/audit: 权限不够

这不是因为 TE 规则的问题，sysadm_t 是可以访问 auditd_log_t 目录的：

引用

sesearch -s sysadm_t -t auditd_log_t --allow|grep dir

allow sysadm_t auditd_log_t : dir { ioctl read getattr lock relabelfrom relabelto search };

[root/secadm_r/s0@tmp]# ll -Zd /var/log/audit

drwxr-x--- root root system_u:object_r:auditd_log_t:s15:c0.c1023 /var/log/audit

红色的 SL 表明 audit 目录的级别是高级别 s15，而 sysadm 默认为 s0（记住，减号前面的为当前值）：

引用

[root/sysadm_r/s0@~]# id -Z

sysadm_u:sysadm_r:sysadm_t:s0-s15:c0.c1023

2）访问目录

id 命令可知，sysadm 的级别范围是 s0-s15:c0.c1023，可通过 **newrole -l** 切换当前级别：

引用

[root/sysadm_r/s0@~]# newrole -l s15:c0.c1023

口令：

[root@localhost root]# source /root/.bash_profile

[root/sysadm_r/s15:c0.c1023@root]# id -Z

sysadm_u:sysadm_r:sysadm_t:s15:c0.c1023

[root/sysadm_r/s15:c0.c1023@root]# ll -d /var/log/audit/

drwxr-x--- 2 root root 4096 07-02 15:23 /var/log/audit/

这样，就可以访问/var/log/audit 目录了。

※ 如果出现下面的错误提示，请看补充说明：

引用

Error: you are not allowed to change levels on a non secure terminal

3）读写日志文件

不过，这是 sysadm_r 还是不能访问日志文件：

引用

[root/sysadm_r/s15:c0.c1023@root]# cat /var/log/audit/audit.log

cat: /var/log/audit/audit.log: 权限不够

那还是按顺序逐一排查，DAC 权限是没问题的（都是 root 宿主）。TE 规则呢？

引用

[root/secadm_r/s0@tmp]# sesearch -s sysadm_t -t auditd_log_t --allow|grep file

allow sysadm_t auditd_log_t : file { getattr relabelfrom relabelto };

没有 read！那只能加一个了，TE 文件为（通过 audit2allow 和参考 auditadm_t 的规则编写）：

引用

[root/secadm_r/s0@tmp]# cat sysadm_audit_log.te

```
module sysadm_audit_log 1.0;

require {
        type sysadm_t;
        type auditd_log_t;
        class file {ioctl read write create getattr setattr lock append unlink link rename };
        class dir {ioctl create rmdir read write getattr lock relabelfrom relabelto add_name remove_name
search };
}
```

#============= sysadm_t ==============

allow sysadm_t auditd_log_t : file { ioctl read write create getattr setattr lock append unlink link rename };

allow sysadm_t auditd_log_t : dir { ioctl read write getattr lock relabelfrom relabelto add_name remove_name search }

编译为 PP 模块后，通过 **semodule** 加载到 SELinux 核心，再查询 TE 规则是：

引用

[root/secadm_r/s0@tmp]# sesearch -s sysadm_t -t auditd_log_t --allow|grep file

allow sysadm_t auditd_log_t : file { ioctl read write create getattr setattr lock relabelfrom relabelto append unlink link rename };

读写审计日志：

引用

[root/sysadm_r/s15:c0.c1023@root]# file /var/log/audit/audit.log

/var/log/audit/audit.log: ASCII text, with very long lines

可见，在同时满足 DAC、TE、MLS 三个条件的情况下，操作顺利通过。

（auditadm、secadm 在 s0 即可读取审计日志，原因见补充说明。但 auditadm 若要修改审计日志，必须切换到 s15 级别）

3.例子二

我们来看看 MLS/MCS 结合的情况。系统自带文件很少使用 category 分类，我们创建四个用户试试。

1）创建用户

引用

处长（chuzhang）级别最高，但同时也涵盖科长、科员的级别范围；

科长（kezhang）中间级别，低于处长，高于科员；

科员（keyuan0、keyuan1）级别最低，但两科员是同级的；

添加 Unix User：

[root/sysadm_r/s0@~]# useradd chuzhang;passwd chuzhang

[root/sysadm_r/s0@~]# useradd kezhang;passwd kezhang

[root/sysadm_r/s0@~]# useradd keyuan0;passwd keyuan0

[root/sysadm_r/s0@~]# useradd keyuan1;passwd keyuan1

创建 SELinux User：

引用

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s0-s5:c1.c20 -L s0 chuzhang_u

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s0-s3:c5.c13 -L s0 kezhang_u

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s0-s2:c10 -L s0 keyuan0_u

[root/secadm_r/s0@~]# semanage user -a -R user_r -P user -r s0-s2:c10 -L s0 keyuan1_u

[root/secadm_r/s0@~]# semanage user -l

Labeling MLS/ MLS/

SELinux User Prefix MCS Level MCS Range SELinux Roles

.....

chuzhang_u user s0 s0-s5:c1.c20 user_r

keyuan0_u user s0 s0-s2:c10 user_r

keyuan1_u user s0 s0-s2:c10 user_r

kezhang_u user s0 s0-s3:c5.c13 user_r

.....

设置登录 context：

引用

[root/secadm_r/s0@~]# semanage login -a -s chuzhang_u -r s2-s5:c1.c20 chuzhang

[root/secadm_r/s0@~]# semanage login -a -s kezhang_u -r s3:c5.c13 kezhang

[root/secadm_r/s0@~]# semanage login -a -s keyuan0_u -r s2:c10 keyuan0

[root/secadm_r/s0@~]# semanage login -a -s keyuan1_u -r s2:c10 keyuan1

[root/secadm_r/s0@~]# semanage login -l

Login Name SELinux User MLS/MCS Range

.....

chuzhang chuzhang_u s2-s5:c1.c20

keyuan0 keyuan0_u s2:c10

keyuan1 keyuan1_u s2:c10

kezhang kezhang_u s3:c5.c13

.....

2）创建文件

分别使用四个用户登录系统，可得到其 context 信息，例如：

引用

[chuzhang@localhost ~]$ id -Z

chuzhang_u:user_r:user_t:s2-s5:c1.c20

[kezhang@localhost ~]$ id -Z

kezhang_u:user_r:user_t:s3:c5.c13

[keyuan0@localhost ~]$ id -Z

keyuan0_u:user_r:user_t:s2:c10

[keyuan1@localhost ~]$ id -Z

keyuan1_u:user_r:user_t:s2:c10

由于/tmp 的 context 为 s0-s15:c0.c1023 ，与四个用户的 context 不相同，在 Enforcing 模式下，他们无法在/tmp 下创建文件：

引用

[root/secadm_r/s0@~]# ll -Zd /tmp

drwxrwxrwt root root system_u:object_r:tmp_t:s0-s15:c0.c1023 /tmp

[chuzhang@localhost ~]$ touch /tmp/chuzhang

touch: 无法触碰 “/tmp/chuzhang”: 权限不够

所以，我们需要先暂时设置为 Permissive 模式：

引用

[root/secadm_r/s0@~]# setenforce 0

[root/secadm_r/s0@~]# getenforce

Permissive

然后，以四个用户分别在/tmp 创建一个以用户名为文件名的测试文件，如：

引用

[chuzhang@localhost ~]$ cd /tmp

[chuzhang@localhost tmp]$ echo chuzhang > chuzhang

[chuzhang@localhost tmp]$ chmod 777 chuzhang

[kezhang@localhost ~]$ cd /tmp

[kezhang@localhost tmp]$ echo kezhang > kezhang

[kezhang@localhost tmp]$ chmod 777 kezhang

[keyuan0@localhost ~]$ cd /tmp

[keyuan0@localhost tmp]$ echo keyuan0 > keyuan0

[keyuan0@localhost tmp]$ chmod 777 keyuan0

[keyuan1@localhost ~]$ cd /tmp

[keyuan1@localhost tmp]$ echo keyuan1 > keyuan1

[keyuan1@localhost tmp]$ chmod 777 keyuan1

（使用 **chmod** 设置为 777，主要是为了避免 DAC 权限的限制）

结果：

引用

[chuzhang@localhost tmp]$ ll -Z chuzhang kezhang keyuan0 keyuan1

-rwxrwxrwx chuzhang chuzhang chuzhang_u:object_r:user_tmp_t:s2 chuzhang

-rwxrwxrwx keyuan0 keyuan0 keyuan0_u:object_r:user_tmp_t:s2:c10 keyuan0

-rwxrwxrwx keyuan1 keyuan1 keyuan1_u:object_r:user_tmp_t:s2:c10 keyuan1

-rwxrwxrwx kezhang kezhang kezhang_u:object_r:user_tmp_t:s3:c5.c13 kezhang

※ 注意

处长（chuzhang）的 context SL 为 s2-s5:c1.c20，当前为 s2，所以创建的文件 SL 也是 s2。为了后续的测试，我们用 newrole -l 命令切换到高级别，然后再创建一个文件：

引用

[chuzhang@localhost tmp]$ newrole -l s4:c1.c15

口令：

[chuzhang@localhost tmp]$ id -Z

chuzhang_u:user_r:user_t:s4:c1.c15-s5:c1.c20

[chuzhang@localhost tmp]$ echo chuzhang1 > chuzhang1

[chuzhang@localhost tmp]$ chmod 777 chuzhang1

[chuzhang@localhost tmp]$ ll -Z chuzhang1

-rwxrwxrwx chuzhang chuzhang chuzhang_u:object_r:user_tmp_t:s4:c1.c15 chuzhang1

3）读写测试

测试前，先把模式恢复到 Enforcing ：

[root/secadm_r/s0@tmp]# setenforce 1

处长的级别比较特殊，暂时不看。以科长、两个科员来看看。

高级别往下读低级别（下读）可行，低级别往上读高级别（上读）失败：

引用

[kezhang@localhost tmp]$ cat keyuan0 keyuan1

keyuan0

keyuan1

[keyuan0@localhost tmp]$ cat kezhang

cat: kezhang: 权限不够

[keyuan1@localhost tmp]$ cat kezhang

cat: kezhang: 权限不够

高级别往下读低级别（下写）失败，低级别往上读高级别（上写）失败：

引用

[kezhang@localhost tmp]$ echo 'test' > keyuan0

-bash: keyuan0: 权限不够

[keyuan0@localhost tmp]$ echo 'test' > kezhang

-bash: kezhang: 权限不够

同级别可读、可写：

引用

[keyuan0@localhost tmp]$ cat keyuan1

keyuan1

[keyuan0@localhost tmp]$ echo 'keyuan0' >> keyuan1

[keyuan0@localhost tmp]$ cat keyuan1

keyuan1

keyuan0

4）宽级别范围的情况

接下来看看处长的情况，它的 context SL 为 s2-s5:c1.c20，我们创建了两个文件，它们的 context SL 不同：

引用

[root/secadm_r/s0@tmp]# ll -Z chuzhang chuzhang1

-rwxrwxrwx chuzhang chuzhang chuzhang_u:object_r:user_tmp_t:s2 chuzhang

-rwxrwxrwx chuzhang chuzhang chuzhang_u:object_r:user_tmp_t:s4:c1.c15 chuzhang1

这就使得科长、两科员也可以访问“chuzhang”这个文件，但写失败：

引用

[kezhang@localhost tmp]$ cat chuzhang

chuzhang

[keyuan0@localhost tmp]$ cat chuzhang

chuzhang

[kezhang@localhost tmp]$ echo 'test' > chuzhang

-bash: chuzhang: 权限不够

[keyuan0@localhost tmp]$ echo 'test' > chuzhang

-bash: chuzhang: 权限不够

同样的，当处长的 SL 为 s2 时，自己创建的“chuzhang1”文件及科长、科员的文件也看不了：

引用

[chuzhang@localhost tmp]$ cat chuzhang1

cat: chuzhang1: 权限不够

[chuzhang@localhost tmp]$ id -Z

chuzhang_u:user_r:user_t:s2-s5:c1.c20

[chuzhang@localhost tmp]$ cat kezhang

cat: kezhang: 权限不够

[chuzhang@localhost tmp]$ cat keyuan0

cat: keyuan0: 权限不够

为什么科员的文件看不了？因为虽然两者同级为 s2，但 s2 的分类（category）没有涵盖 s2:c10 的分类！

试试：

引用

[chuzhang@localhost tmp]$ ll -Z keyuan0

-rwxrwxrwx keyuan0 keyuan0 keyuan0_u:object_r:user_tmp_t:s2:c10 keyuan0

[chuzhang@localhost tmp]$ newrole -l s2:c5.c15

口令：

[chuzhang@localhost tmp]$ id -Z

chuzhang_u:user_r:user_t:s2:c5.c15-s5:c1.c20

[chuzhang@localhost tmp]$ cat keyuan0

keyuan0

[chuzhang@localhost tmp]$ echo 'test' >> keyuan0

bash: keyuan0: 权限不够

那同级呢？

引用

[chuzhang@localhost tmp]$ newrole -l s2:c10

口令：

[chuzhang@localhost tmp]$ id -Z

chuzhang_u:user_r:user_t:s2:c10-s5:c1.c20

[chuzhang@localhost tmp]$ cat keyuan0

keyuan0

[chuzhang@localhost tmp]$ echo 'chuzhang' >> keyuan0

[chuzhang@localhost tmp]$ cat keyuan0

keyuan0

chuzhang

同级是可读可写！同级指的是 SL 和 CL 都必须相同（eq）。

对科长创建的文件当然也是一样的：

引用

[chuzhang@localhost tmp]$ ll -Z kezhang

-rwxrwxrwx kezhang kezhang kezhang_u:object_r:user_tmp_t:s3:c5.c13 kezhang

[chuzhang@localhost tmp]$ newrole -l s3:c1.c15

口令：

[chuzhang@localhost tmp]$ id -Z

chuzhang_u:user_r:user_t:s3:c1.c15-s5:c1.c20

[chuzhang@localhost tmp]$ cat kezhang

kezhang

[chuzhang@localhost tmp]$ echo 'chuzhang' >> kezhang

bash: kezhang: 权限不够

[chuzhang@localhost tmp]$ newrole -l s3:c5.c13

口令：

[chuzhang@localhost tmp]$ id -Z

chuzhang_u:user_r:user_t:s3:c5.c13-s5:c1.c20

[chuzhang@localhost tmp]$ cat kezhang

kezhang

[chuzhang@localhost tmp]$ echo 'chuzhang' >> kezhang

[chuzhang@localhost tmp]$ cat kezhang

kezhang

chuzhang

※ 注意：级别可变，分类可在规定范围内改变，超出范围则是不行的。

例如：

引用

[chuzhang@localhost ~]$ id -Z

chuzhang_u:user_r:user_t:s2-s5:c1.c20

[chuzhang@localhost ~]$ newrole -l s6

chuzhang_u:user_r:user_t:s6-s5:c1.c20 不是一个有效的 context

[chuzhang@localhost ~]$ newrole -l s3:c0

chuzhang_u:user_r:user_t:s3:c0-s5:c1.c20 不是一个有效的 context

[chuzhang@localhost ~]$ newrole -l s3:c1.c30

chuzhang_u:user_r:user_t:s3:c1.c30-s5:c1.c20 不是一个有效的 context

4.补充说明

1）允许ssh 终端切换安全级别

如果在使用 newrole -l 命令时报：

引用

newrole -l s15:c0.c1023

Error: you are not allowed to change levels on a non secure terminal

只需在/etc/selinux/mls/contexts/securetty_types 中加入许可的 tty 类型。例如，ssh 的 tty 为：

引用

[root/sysadm_r/s0@~]# tty

/dev/pts/1

那么，要允许 sysadm_r 就加入：

引用

sysadm_devpts_t

2）MLS 的限制条件在那里规定

请参考Harry Ciao 的《SELinux 学习笔记》：3.7 Constraints and MLS constraints 的介绍。

refpolicy中所有MLS限制条件都集中在policy/mls文件中，比如“no read up”概念可实现为：

mlsconstrain { dir file lnk_file chr_file blk_file sock_file fifo_file } { read getattr execute }

((l1 eq l2) or (l1 dom l2) or (t1 == mlsfileread)) ;

即只允许“read equal/down”，或者当前 subj_t 属于 mlsfileread 属性。

MLS 的“no write down”概念可实现为：

mlsconstrain { file lnk_file fifo_file dir chr_file blk_file sock_file } { write create setattr relabelfrom

append unlink link rename mounton }

((l1 eq l2) or (l1 domby l2) or (t1 == mlsfilewrite)) ;

即只允许“write equal/up”，或者当前 subj_t 属于 mlsfilewrite 属性。

3）为什么 secadm_r 和 auditadm_r 在默认 s0-s15:c0.c1023 的条件下，可读取审计日志呢？

这是因为，所有 MLS 属性都定义在 policy/modules/kernel/mls.te 中，并且在 mls.if 中定义了许多接口，使得调用

者 domain 能够绕过 MLS 约束的限制，比如：

```
interface(`mls_file_read_all_levels',`
        gen_require(`
                attribute mlsfileread;
        ')
        typeattribute $1 mlsfileread;
')
```

该接口使得调用者 domain 加入 mlsfileread 属性，从而不受“no read up”的约束。很多与系统管理相关的 domain，比如 semanage_t, setfiles_t, newrole_t, udev_t, secadm_t 都调用了类似的接口。

4）MCS translation service 服务

为了提供易识别的分类显示，系统提供一个叫 mcstrans 的服务，但 RFSOS 4 默认是关闭的，需手动打开：

引用

```
[root/sysadm_r/s0@~]# service mcstrans start
启动 mcstransd：                                [确定]
```

打开后，原来的 s0，会显示为 SystemLow，类似：

引用

```
[root/sysadm_r/SystemLow@~]# ll -Z mytest.pp
-rw-r--r--  root root root:object_r:sysadm_home_t:SystemLow mytest.pp
[root/secadm_r/SystemLow@~]# semanage login -l
Login Name              SELinux User            MLS/MCS Range
__default__             user_u                  SystemLow
```

auditadm auditadm_u SystemLow-SystemHigh

root root SystemLow-SystemHigh

secadm secadm_u SystemLow-SystemHigh

sysadm sysadm_u SystemLow-SystemHigh

system_u system_u SystemLow-SystemHigh

可通过 **chcat -L** 命令查看：

引用

[root/sysadm_r/SystemLow@~]# chcat -L

s0 SystemLow

s15:c0.c1023 SystemHigh

.....

这些都是定义在/etc/selinux/mls/setrans.conf 文件里面的：

引用

[root/secadm_r/SystemLow@~]# cat /etc/selinux/mls/setrans.conf|grep -v '^#'|grep -v '^$'|head -n10

s0=SystemLow

s15:c0.c1023=SystemHigh

s0-s15:c0.c1023=SystemLow-SystemHigh

s1=Unclassified

s2=Secret

s2:c0=A

s2:c1=B

RHEL 6 也可通过 **semanage translation** 命令进行修改。另外，chcat 命令可用于文件的安全上下文分类（SELinux security category），但 RFSOS4 自带的版本与系统默认编码有冲突（chcat 是一个 python 脚本），不能使用，需要升级到 policycoreutils-1.33.12-14.8 及以上版本解决。

11. 自定义安全模块

作为系统管理员来说，熟悉前面所描述的内容已基本可满足日常应用系统管理的需要。如：布尔值、安全上下文属性配置、使用**audit2allow** 生成TE模型类别的允许规则模块，以及MLS/MCS 属性配置等。不过，这些仅仅是基于系统自带安全模块的调整，而对于由第三方开发的应用而言，这是不够的。我们需要让这些应用拥有自己的类别（Type）、域（Domain）、规则（Rule），才能完整的利用SELinux 所提供的安全模型，而不能简单的把它们加到已有类别或域中。否则，类别控制、域隔离将无从谈起。这就需编写自定义模块，编译后加载到SELinux 核心中。

编写安全模块的工作最好由应用开发者协助或完成，因为这涉及很多应用调用的资源、顺序、权限等问题，由用户或系统管理员处理难度比较大。RFSOS4 的SELinux 安全策略是基于referrence policy，可从示例入手，并参考说明文档和现有安全模块的源码来编写。红旗在RFSOS4 上也提供了一个可简化安全策略模块编写的图形工具。这部分内容很多，我自问写不出比现有资料更好的文档，故仅做简介，并提供一些不错的文档供参考。

十一、自定义安全模块

1.简介

RFSOS4 的SELinux 安全策略是基于referrence policy，其官网为：http://oss.tresys.com/projects/refpolicy，特点为：

引用

1. 一套代码树同时支持 strict/targeted 两种类型的 policy；支持 MLS/MCS；支持将 policy 编译成两种格式：Monolithic 或 Modular；
2. 采用“高内聚”，“低耦合”的设计思想，pp 定义清晰的外部接口，所有 type 的定义都在模块内使用，取消了全局标识符（type/attribute 等）；
3. 对文档的支持，在.if 中定义的 interface 都有 XML 格式的语法，可以用 make html 生成所有接口的文档；
4. 简化并标准化了 policy 的编译参数，在社区发行版中都集中在 build.conf 中。

（这说明来自Harry Ciao 的《SELinux 学习笔记》）

在reference policy 中，任何一个PP 模块都由三个文件组成：

引用

1. 类型强制文件(te) ―包含应用程序的安全策略
2. 文件关联文件(fc) ―包含文件和目录的安全上下文定义
3. 界面文件(if) ―包含应用程序需要使用策略接口文件

最新版本，可从这里下载：点击。

2.示例

看一个官网上提供的例子：这里。

本地下载：

下载文件

点击这里下载文件

TE 文件 myapp.te 为：

查看复制打印关于

```
policy_module(myapp,1.0)

# 定义私有类别
type myapp_t;
type myapp_exec_t;
type myapp_log_t;
type myapp_tmp_t;

# 用宏来定义私有域
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t)
logging_log_file(myapp_log_t)
files_tmp_file(myapp_tmp_t)

# 定义规则
allow myapp_t myapp_log_t:file append_file_perms;
allow myapp_t myapp_tmp_t:file manage_file_perms;

# 允许在/tmp 写入文件
files_tmp_filetrans(myapp_t,myapp_tmp_t,file)
```

```
policy_module(myapp,1.0)

# 定义私有类别
type myapp_t;
type myapp_exec_t;
type myapp_log_t;
type myapp_tmp_t;

# 用宏来定义私有域
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t)
logging_log_file(myapp_log_t)
files_tmp_file(myapp_tmp_t)

# 定义规则
allow myapp_t myapp_log_t:file append_file_perms;
allow myapp_t myapp_tmp_t:file manage_file_perms;
```

FC 文件 myapp.fc 为：

查看复制打印关于

```
1. # 使用 gen_context 宏定义可执行文件的安全上下文，其中，MCS 是可选的
2. /usr/bin/myapp  --  gen_context(system_u:object_r:myapp_exec_t,s0)
3. # 若使用 MLS/MCS ，则可能类似这样：
4. # /usr/bin/myapp  --  gen_context(system_u:object_r:myapp_exec_t,s5:c1
   ,c0)
```

```
# 使用gen_context 宏定义可执行文件的安全上下文，其中，MCS 是可选的
/usr/bin/myapp  --  gen_context(system_u:object_r:myapp_exec_t,s0)
# 若使用MLS/MCS ，则可能类似这样：
# /usr/bin/myapp  --  gen_context(system_u:object_r:myapp_exec_t,s5:c1,c0)
```

IF 文件 myapp.if 为：

查看复制打印关于

```
1. ## <summary>Myapp example policy</summary>
2. ## <desc>
3. ##  <p>
4. ##    More descriptive text about myapp.  The desc
5. ##    tag can also use p, ul, and ol
6. ##    html tags for formatting.
```

```
##  </p>
##  <p>
##    This policy supports the following myapp features:
##    <ul>
##    <li>Feature A</li>
##    <li>Feature B</li>
##    <li>Feature C</li>
##    </ul>
##  </p>
## </desc>

########################################
## <summary>
##  Execute a domain transition to run myapp.
## </summary>
## <param name="domain">
##  <summary>
##  Domain allowed to transition.
##  </summary>
## </param>

interface(`myapp_domtrans',`
  gen_requires(`
    type myapp_t, myapp_exec_t;
  ')

  domtrans_pattern($1,myapp_exec_t,myapp_t)
')

########################################
## <summary>
##  Read myapp log files.
## </summary>
## <param name="domain">
##  <summary>
##  Domain allowed to read the log files.
##  </summary>
## </param>

interface(`myapp_read_log',`
  gen_requires(`
    type myapp_log_t;
  ')

```

```
    logging_search_logs($1)
    allow $1 myapp_log_t:file read_file_perms;
')
```

```
## <summary>Myapp example policy</summary>
## <desc>
##  <p>
##    More descriptive text about myapp.  The desc
##    tag can also use p, ul, and ol
##    html tags for formatting.
##  </p>
##  <p>
##    This policy supports the following myapp features:
##    <ul>
##    <li>Feature A</li>
##    <li>Feature B</li>
##    <li>Feature C</li>
##    </ul>
##  </p>
## </desc>

########################################
```

该文件采用XML 语法。其定义了其他模块进入本资源的接口，提供单点入口形式。第一个接口允许其他域通过执行标签为myapp_exec_t 的程序迁移到myapp_t 域上；第二个接口允许其他域读取myapp的日志文件。

三个文件编写完成后，即可编译为PP 模块（步骤请见[原]SELinux 简明学习指南——添加允许规则的补充说明部分）

引用

[root/secadm_r/SystemLow@tmp]# ll myapp*

-rw-r--r-- 1 root root 65 06-28 14:55 myapp.fc

-rw-r--r-- 1 root root 1031 06-28 14:55 myapp.if

-rw-r--r-- 1 root root 431 06-28 14:55 myapp.te

[root/secadm_r/SystemLow@tmp]# make -f /usr/share/selinux/devel/Makefile

Compiling mls myapp module

/usr/bin/checkmodule: loading policy configuration from tmp/myapp.tmp

/usr/bin/checkmodule: policy configuration loaded

/usr/bin/checkmodule: writing binary representation (version 6) to tmp/myapp.mod

Creating mls myapp.pp policy package

rm tmp/myapp.mod.fc tmp/myapp.mod

[root/secadm_r/SystemLow@tmp]# ll myapp.pp

-rw-r--r-- 1 root root 31905 06-28 14:57 myapp.pp

把模块加入 SELinux 核心，并验证：

引用

[root/secadm_r/SystemLow@myapp]# semodule -i myapp.pp

[root/secadm_r/SystemLow@myapp]# semodule -l|grep myapp

myapp 1.0

[root/secadm_r/SystemLow@myapp]# sesearch -s myapp_t -t myapp_tmp_t --allow

Found 2 av rules:

allow myapp_t myapp_tmp_t : file { ioctl read write create getattr setattr lock append unlink link rename };

allow myapp_tmp_t myapp_tmp_t : filesystem associate ;

[root/secadm_r/SystemLow@tmp]# touch test

[root/sysadm_r/SystemLow@tmp]# ll -Z test

-rw-r--r-- root root secadm_u:object_r:secadm_tmp_t:SystemLow test

[root/secadm_r/SystemLow@tmp]# chcon -t myapp_exec_t test

[root/secadm_r/SystemLow@tmp]# ll -Z test

-rw-r--r-- root root secadm_u:object_r:myapp_exec_t:SystemLow test

可见，myapp_t 类别、域、规则都已生效。当然，实际的应用软件不会是这么简单的。

3.图形工具

网上有不少 GUI 的配置工具，红旗 RFSOS4 也提供了一个“安全策略生成工具”用于简化应用程序安全策略模块的编写工作。执行文件是 polgengui ，在桌面上可看到快捷方式：

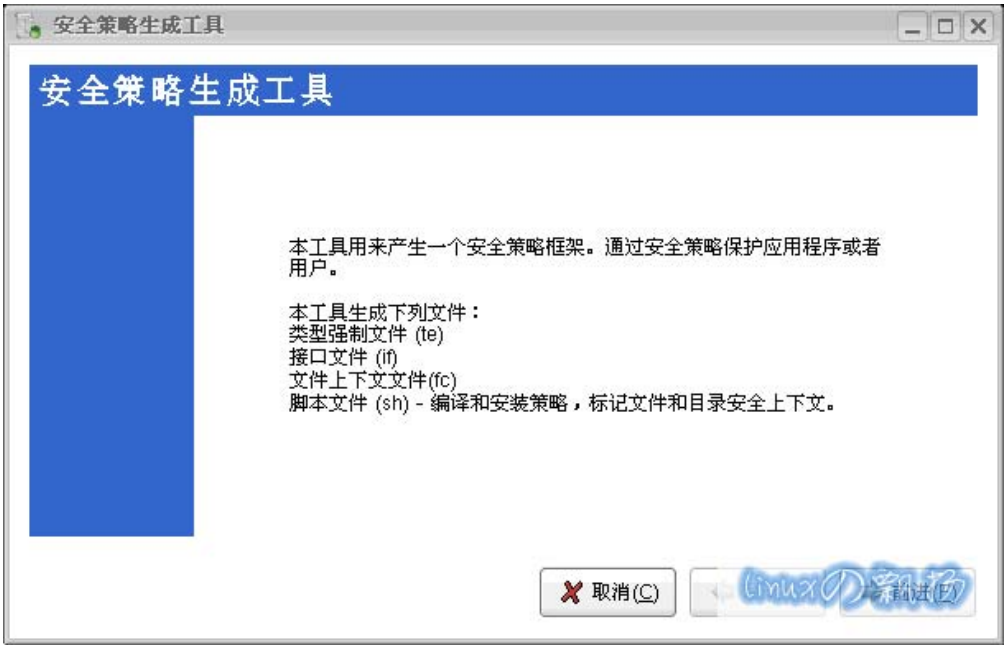
由于安全策略生成工具用于调试生成外来应用的安全策略，该工具被设计为只能运行于警报模式。使用前执行：

[root/secadm_r/SystemLow@~]# setenforce 0

※ 建议，应用程序最好按系统默认的路径存放执行和配置文件等信息，如执行文件放在/usr/bin，配置文

件放在/etc 下等，以便于利用工具提供的默认策略。

安全策略生成工具采用向导式操作



分七步完成安全策略生成过程：

1）选择应用程序类型

该对话框定义了用程序类型，这有助于设置对应守护进程正确地生成安全策略。

可设置的应用程序类型有：

引用

标准 daemon 守护进程，这些应用程序在 init 调用 rc.sysinit 或者/etc/init.d 目录下的脚本时启动。

inetd 服务守护进程，这些应用程序有 inetd 或者 xinetd 启动。

Web 应用程序/脚本（CGI），这些应用程序一般由 Apache 运行。

用户应用程序，通常由管理员或者用户直接在终端运行。

2）定义需要保护的应用程序名称

此对话框指定受保护的应用程序的域名和程序路径，策略生成工具会自动创建对应主体域名和对应可执行程序文件的安全上下文。例如，如果定义应用程序域名为 owcimomd，则自动创建 owcimomd_t 域和对应可执行文件上下文 owcimomd_exec_t。

※ 注意：如果定义的域名与系统内置安全策略中已有的域名重名，工具将提示并终止执行。

3）指定输入网络端口连接

该对话框允许管理员输入用空格分隔的一系列用于输入连接的应用程序绑定/监听网络端口。如果管理员无法确定端口信息，可以留空不填（将自动允许 1-65535 端口）。这与网络防火墙是不相关的，而是指 semanage port 属性的设置。

4）指定输出网络端口连接

该对话框允许管理员指定应用程序连接的 TCP 和 UDP 端口。

5）标识通用应用程序特征

该对话框允许管理员标识应用程序的运行特征，选中方框将相应的特征加入策略模板并允许应用程序使用这些功能。如果不确定应用程序具备哪些通用特征，可以留空不填，在调试阶段生成相应的策略。

例如：owcimomd 程序会访问 syslog 日志，并且在/tmp 创建有名管道，采用 pam 机制认证，并且有 setuid 操作，所以选择以上几种特征。

6）指定受保护的应用程序文件和目录和设置布尔值

该对话框允许指定应用程序需要操作的配置文件和数据目录，这些文件和目录将受到安全策略保护。工具将建立该文件和目录的安全上下文。例如：如果指定 owcimomd 程序配置文件 /etc/openwbem/openwbem.conf，则工具会自动产生该文件安全上下文 owcimomd_etc_t。**※ 注意：**输入

新的路径会产生新的文件安全上下文。产生的文件或者目录的安全上下文应该唯一性，不应该覆盖系统中已有文件上下文，否则后面的策略编译操作会失败。

由于应用程序执行文件对应的安全上下文已经在第一步操作生成，这里无须再次指定该执行文件。产生无效的安全上下文是在定制策略的时候常犯的错误。

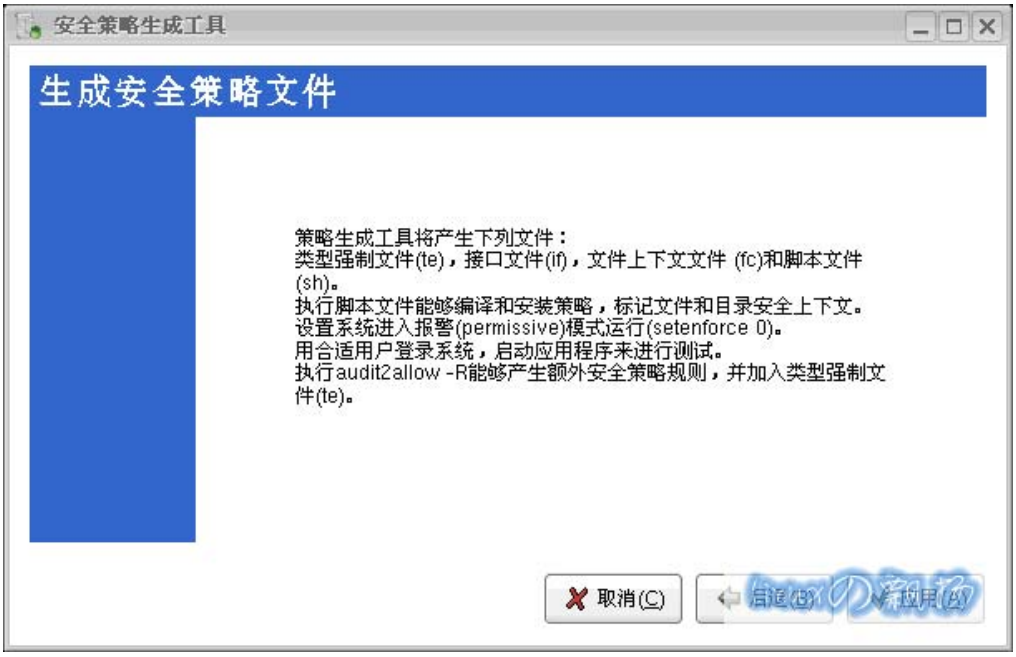
该对话框可定义布尔值变量，以方便实时生效的修改安全策略，是可选的。

7）选择安全策略存放目录

该对话框询问安全策略的输出存放目录，默认是当前工作目录，但最好指定一个单独的可读写目录，以便

于后续的调试工作。（编译为 PP 模块后不需再读取该目录）

8）生成策略文件



点击“应用”会生成相应的文件，工具会显示结果：

除之前提到的三个策略源码文件外，还有一个.sh 的脚本，这主要是用来在测试系统上编译，安装和调试模块的。

9）调试

利用.sh 脚本进行调试的工作。打开一个终端窗口，以 root 用户（secadm 已 su 到 root 即可）执行该脚本。安全策略模块会被加载到 SELinux 核心中，并自动执行 **restorecon** 命令把之前定义的应用程序使用文件（及配置文件等目录）的安全上下文修正。

```
[root/secadm_r/SystemLow@~]# cd openwbem
[root/secadm_r/SystemLow@openwbem]# sh owcimomd.sh
```

保持在警报（Permissive）模式下，以正确的用户启动应用程序进行测试。观察审计日志，由于把安全策略模块加入核心后，新的审计规则才会生效，所以，可利用 ausearch 命令搜索这段时间发生的审计日志，并通过 audit2allow -R 生成新的规则再导入.te 文件中，产生更完整的策略。.sh 脚本提供-update 参数自动完成这个步骤：

```
[root/secadm_r/SystemLow@openwbem]# sh owcimomd.sh -update
```

测试应用，重复这些步骤，直到没有新的阻止审计日志产生为止。最后，恢复为强制（Enforcing）模式，查看应用使用是否正常。

```
[root/secadm_r/SystemLow@~]# setenforce 1
```

这就是使用“安全策略生成工具”创建安全策略模块的过程。当然，你也可以自行编译安全策略。

4.手动编译安全策略文件

这可不是一个轻松活！如果你是应用开发者，那有时间熟悉一下还是不错的。SLIDE（基于Eclipse 开发的）、SEEDIT 是一些不错的GUI 编辑工具。这部分自问没有什么意见，就提供几份不错的学习资料供研究参考吧。

1）SELinux 学习笔记

下载地址：这里

本地下载：

下载文件

点击这里下载文件

2）SELinux详解

51CTO 提供的：这里

本地下载：

下载文件

点击这里下载文件

3）An SElinux module

这是国外equivocation.org提供的一份关于SELinux 的系列资料，英文的。虽然时间比较早，但内容挺全面的，值得看看。在关于selinux module 部分，以lighttpd 这个轻量级的HTTP Server 来描述安全策略模块各部分的编写要点，是快速入门的不错选择。分四部分：

An SElinux module (1): types and rules

An SElinux module (2): file contexts and labelling

An SElinux module (3): interfaces

An SElinux module (4): building and debugging

本地下载：

下载文件

点击这里下载文件

如果这些都不能满足要求，那你可以找 RFSOS、RHEL/CentOS、Fedora 的 selinux-policy-mls 源码。其中的 serefpolicy 压缩包有大量系统自带应用的安全策略模块可供参考。另外，红旗提供的 oracle-selinux 软件包也可以看看：

引用

/usr/share/doc/oracle-selinux-0.1/oracle.fc

/usr/share/doc/oracle-selinux-0.1/oracle.if

/usr/share/doc/oracle-selinux-0.1/oracle.te

5.补充说明

我们在[原]SELinux 简明学习指南——安全上下文曾提过，系统提供一个由非限制域进程（Unconfined Processes）运行的unconfined_t 域。这本来是用于一些特殊系统进程在系统启动时调用的。

但我们在安全策略模块中可使用unconfined_domain() 宏，把应用程序放入该域中。当然，这是很危险的，这是一个无限域，具有很大的访问权限，除非迫不得已，不要这样做！

另外，安全策略模块主要着重在TE 模型的类别规则上，如果规则与MLS/MCS 级别有冲突，需注意“上读下写”的问题。下面的宏可用于实现这部分的策略：

引用

mls_file_read_up()

mls_file_write_down()

12. 常用命令

audit2allow 通过分析审计日志信息，把原被禁止的操作转为允许的规则，并生成规则模块
audit2why 从审计日志得到一些解决建议
aureport 查看汇总的审计信息
ausearch 对审计日志进行检索
chcat 查看或修改文件的MCS Category类别标记
chcon 修改文件的安全上下文
getenforce 查看运行模式
getsebool 查看当前Booleans状态
matchpathcon 检查文件和目录的安全上下文是否与规则一致
newrole 切换用户的当前安全上下文
restorecon 恢复文件的默认安全上下文
seinfo 查询系统SELinux policy 所提供各属性值的情况
semanage 对SELinux 定义的属性进行配置
semodule 对SELinux 模块进行操作
sesearch 查询规则
sestatus 查看SELinux 状态
setenforce 修改运行模式
setsebool 修改当前Booleans状态