УДК 004.451
ББК 32.972.1
В34

Вермейлен С.
В34 Администрирование системы защиты SELinux / пер. с анг. В. Л. Верещагина, О. К. Севостьяновой. – М.: ДМК Пресс, 2020. – 300 с.: ил.

ISBN 978-5-97060-557-8

Эта книга показывает, как значительно усилить безопасность операционной системы Linux и устранить имеющиеся уязвимости установленных приложений.

Вы узнаете, как работает SELinux, как можно настроить ее под свои нужды и усилить с ее помощью защиту систем виртуализации, включающих технологию libvirt (sVirt) и контейнеризацию Docker. Также рассказывается об управляющих действиях, позволяющих улучшить безопасность конкретной системы с помощью принудительного контроля доступа – стратегии защиты, определяющей безопасность Linux уже много лет. Большинство возможностей системы защиты рассматривается на реальных примерах.

Книга предназначена для администраторов операционной системы Linux, в задачу которых входит управление ее защищенностью.

УДК 004.451
ББК 32.972.1

Authorized Russian translation of the English edition of SELinux System Administration ISBN 978-1-78712-695-4 © 2016 Packt Publishing.

This translation is published and sold by permission of Packt Publishing, which owns or controls all rights to publish and sell the same.

ISBN 978-1-78712-695-4 (анг.)
ISBN 978-5-97060-557-8 (рус.)

Содержание

Об авторе 13

О рецензентах 15

Предисловие 16

Глава 1. Фундаментальные концепции SELINUX 21

1.1. Предоставление большей безопасности в Linux 21
1.1.1. Использование модулей безопасности Linux 24
1.1.2. Расширение возможностей стандартного дискреционного разграничения доступа 25
1.1.3. Ограничение привилегий пользователя root 27
1.1.4. Сокращение воздействия уязвимостей 28
1.1.5. Включение возможностей SELinux в операционной системе 29
1.2. Маркировка всех ресурсов и объектов 31
1.2.1. Описание параметров безопасности 32
1.2.2. Принудительный доступ посредством типов функциональных ограничений 35
1.2.3. Распределение по ролям наборов функциональных ограничений 36
1.2.4. Разделение пользователей по ролям 38
1.2.5. Контроль информационных потоков посредством мандатного механизма 39
1.3. Формирование и распределение политик 40
1.3.1. Создание политик SELinux 41
1.3.2. Распределение политик в виде модулей 43
1.3.3. Комплектация модулей в хранилище политик 45
1.4. Различия между политиками 45
1.4.1. Поддержка многоуровневой защиты (MLS) 46
1.4.2. Манера поведения с неизвестными разрешениями 46
1.4.3. Поддержка неограниченных доменов 47
1.4.4. Ограничение межпользовательского обмена 48
1.4.5. Последовательные изменения версий политик 49
1.4.6. Качественное изменение версий политик 50
1.5. Заключение 51

Глава 2. Режимы работы и регистрация событий 53

2.1. Включение и выключение защиты SELinux 53
2.1.1. Установка глобального состояния защиты 54

2.1.2. Переключение в рекомендательный и принудительный режимы 55
2.1.3. Использование параметров загрузки ядра 57
2.1.4. Отключение защиты SELinux для отдельно взятого сервиса 58
2.1.5. Определение приложений, активно взаимодействующих с SELinux 61
2.2. Регистрация событий и аудит в SELinux 61
2.2.1. Последовательность контроля событий о нарушениях безопасности 62
2.2.2. Исключение конкретных отказов в доступе из числа регистрируемых 64
2.2.3. Конфигурирование подсистемы контроля событий безопасности Linux 65
2.2.4. Настройка локального системного регистратора событий 66
2.2.5. Разбор информации об отказах SELinux 67
2.2.6. Другие типы событий, связанные с SELinux 72
2.2.7. Использование команды ausearch 76
2.3. Получение помощи при отказах 77
2.3.1. Диагностика неисправности с помощью службы setroubleshoot 77
2.3.2. Отправка электронной почты, когда случился отказ SELinux 80
2.3.3. Использование утилиты audit2why 81
2.3.4. Взаимодействие с журналом system 82
2.3.5. Использование здравого смысла 83
2.4. Заключение 85

Глава 3. Управление учетными записями пользователей 86

3.1. Параметры безопасности пользователей 86
3.1.1. Сложность допустимого набора функций 87
3.1.2. Определение неограниченных доменов 89
3.2. Пользователи SELinux и их роли 90
3.2.1. Перечень сопоставлений пользователей с пользовательскими типами SELinux 90
3.2.2. Сопоставление учетных записей с пользовательскими типами 92
3.2.3. Настройка учетных записей относительно служб 93
3.2.4. Создание типов пользователей SELinux 94
3.2.5. Перечень типов допустимого набора функций у ролей 95
3.2.6. Управление категориями 96
3.3. Управление ролями SELinux 98
3.3.1. Настройки присвоения допустимых ролей пользователю 98
3.3.2. Проверка параметров безопасности при помощи утилиты getseuser 100
3.3.3. Подключение ролей с помощью команды newrole 100
3.3.4. Управление доступом к роли с помощью команды sudo 101
3.3.5. Переключение параметров безопасности посредством runcon 102

3.3.6. Переключение на системную роль 102
3.4. SELinux и PAM (подключаемые модули аутентификации) 104
3.4.1. Назначение параметров безопасности с помощью подключаемых модулей аутентификации 104
3.4.2. Запрещение доступа в рекомендательном режиме работы защиты 105
3.4.3. Многоэкземплярность каталогов 106
3.5. Заключение 107

Глава 4. Домены как допустимые наборы функций для процессов и контроль доступа на уровне файлов 109
4.1. О параметрах безопасности файлов 109
4.1.1. Получение информации о параметрах безопасности 110
4.1.2. Интерпретация наименований типов SELinux 111
4.2. Закрепление параметров безопасности за объектом и их игнорирование 112
4.2.1. Наследование параметров безопасности по умолчанию 113
4.2.2. Правила преобразования типов и их вывод 113
4.2.3. Копирование и перемещение файлов 115
4.2.4. Временное изменение параметров безопасности файла 117
4.2.5. Установка категорий для файлов и каталогов 118
4.2.6. Использование многоуровневой защиты для файлов 118
4.2.7. Резервное копирование и восстановление расширенных атрибутов 119
4.2.8. Использование опций монтирования для установки параметров SELinux 119
4.3. Формулировка параметров безопасности для файлов 121
4.3.1. Использование выражений, описывающих параметры безопасности 121
4.3.2. Регистрация изменений параметров безопасности файлов 123
4.3.3. Использование заказных типов 125
4.3.4. Различные виды файлов file_contexts и их компиляция 126
4.3.5. Обмен локальными изменениями 127
4.4. Изменение параметров безопасности у файлов 127
4.4.1. Использование команд setfiles, rlpkg и fixfiles 128
4.4.2. Изменение параметров безопасности на всей файловой системе 128
4.4.3. Автоматическое приведение к заданным значениям изменившихся параметров безопасности 129
4.5. Параметры безопасности процесса 130
4.5.1. Получение параметров безопасности процесса 130
4.5.2. Преобразование типа процесса 131
4.5.3. Проверка соответствия параметров безопасности 133
4.5.4. Другие способы преобразования типов 134

4.5.5. Изначально заданные параметры в структуре идентификатора безопасности ... 134
4.6. Определение границ возможных преобразований ... 135
4.6.1. Очистка переменных окружения во время преобразования к другому типу ... 135
4.6.2. Невыполнение преобразований, когда нет ограничивающего родительского типа ... 137
4.6.3. Использование флага, исключающего новые привилегии у процесса ... 138
4.7. Типы, разрешения и ограничения ... 139
4.7.1. Объяснение атрибутов типа ... 139
4.7.2. Запрос разрешений, предоставленных типу процесса ... 140
4.7.3. Рассмотрение наложенных ограничений ... 142
4.8. Заключение ... 143

Глава 5. Контроль сетевого взаимодействия ... 144

5.1. От контроля межпроцессного взаимодействия (IPC) до сокетов базовых протоколов (TCP/UDP) транспортного уровня ... 144
5.1.1. Использование разделяемой памяти ... 145
5.1.2. Локальное взаимодействие, осуществляемое по каналам ... 147
5.1.3. Обращение через сокеты домена UNIX ... 148
5.1.4. Рассмотрение сокетов netlink ... 149
5.1.5. Действия с сокетами протоколов TCP и UDP ... 150
5.1.6. Вывод списка сетевых соединений с параметрами безопасности ... 152
5.2. Межсетевой экран и маркировка сетевых пакетов ... 152
5.2.1. Вводные сведения о межсетевом экране netfilter ... 153
5.2.2. Реализация маркировки сетевых пакетов и соединений ... 154
5.2.3. Назначение меток пакетам ... 155
5.3. Промаркированные сети ... 157
5.3.1. Резервная маркировка в NetLabel ... 158
5.3.2. Ограничение потоков данных на уровне сетевого интерфейса ... 160
5.3.3. Ограничение потоков данных на уровне элементов сети ... 160
5.3.4. Проверка однорангового потока ... 161
5.3.5. Применение управления в старом стиле ... 162
5.4. Метки безопасности для IPsec ... 163
5.4.1. Установка стандартного IPsec ... 165
5.4.2. Подключение маркировки IPsec ... 166
5.4.3. Использование Libreswan ... 167
5.5. Технология маркировки сетей NetLabel с параметром CIPSO ... 168
5.5.1. Настройка сопоставлений потоков данных с доменами ... 169
5.5.2. Добавление сопоставлений для типов допустимого набора функций ... 170
5.5.3. Локальное использование параметра CIPSO ... 171

5.5.4. Поддержка опции безопасности для IPv6 172
5.6. Заключение 172

Глава 6. Поддержка sVirt и Docker 173

6.1. Виртуализация, защищенная SELinux 173
6.1.1. Представление о виртуализации 173
6.1.2. Обзор рисков виртуализации 175
6.1.3. Использование типов для объектов виртуальной инфраструктуры 176
6.1.4. Перенастраиваемое применение существующих типов виртуализации 177
6.1.5. Рассмотрение защиты различных категорий 179
6.2. Поддержка библиотеки libvirt 180
6.2.1. Различные случаи маркировки ресурсов 181
6.2.2. Оценка архитектуры libvirt 181
6.2.3. Настройка libvirt для работы с sVirt 182
6.2.4. Использование статических параметров безопасности 184
6.2.5. Гибкая настройка параметров безопасности 184
6.2.6. Использование разных мест хранения 185
6.2.7. Интерпретация информации в поле вывода данных о метке 185
6.2.8. Управление доступными категориями 186
6.2.9. Поддержка интерпретирующих доменов 186
6.2.10. Изменение параметров безопасности, установленных по умолчанию 187
6.3. Защищенные контейнеры Docker 188
6.3.1. Представление о защите контейнера 188
6.3.2. Интеграция системы защиты с контейнерами без sVirt 189
6.3.3. Перестраховка безопасности Docker средствами защиты sVirt 190
6.3.4. Ограничение привилегий контейнера 191
6.3.5. Применение различных параметров безопасности для контейнеров 193
6.3.6. Перемаркировка подключенного тома данных 194
6.3.7. Понижение контроля со стороны SELinux для специальных контейнеров 195
6.3.8. Изменение параметров безопасности, установленных по умолчанию 195
6.4. Заключение 196

Глава 7. D-Bus и systemd 197

7.1. Фоновый процесс системы (systemd) 197
7.2. Способ поддержки в systemd служб 198
7.2.1. Введение понятия модульных файлов 198

7.2.2. Установка параметров безопасности SELinux для какой-либо службы ... 199
7.2.3. Использование переходных служб ... 200
7.2.4. Требование включения или отключения SELinux для конкретной службы ... 201
7.2.5. Перемаркировка файлов во время запуска службы ... 202
7.2.6. Использование активизации, основанной на сокетах ... 204
7.2.7. Управление доступом к операциям с модулями ... 205
7.3. Регистрация событий с помощью systemd ... 206
7.3.1. Получение информации, относящейся к SELinux ... 206
7.3.2. Запрос событий, содержащих параметры безопасности SELinux ... 207
7.3.3. Интеграция диагностики неисправностей с журналом ... 207
7.4. Использование контейнеров systemd ... 209
7.4.1. Инициализация контейнеров systemd ... 209
7.4.2. Использование специальных параметров безопасности SELinux ... 209
7.5. Управление файлами устройств ... 210
7.5.1. Использование правил udev ... 210
7.5.2. Назначение метки SELinux на узле устройства ... 212
7.6. Взаимодействие с шиной сообщений D-Bus ... 212
7.6.1. Представление о взаимодействии между процессами D-Bus ... 212
7.6.2. Контроль получения доступа к службам с помощью SELinux ... 215
7.6.3. Управление потоками сообщений ... 216
7.7. Заключение ... 217

Работа с политиками SELinux ... 218

8.1. Логические параметры SELinux ... 218
8.1.1. Вывод списка логических параметров ... 219
8.1.2. Изменение значений логических параметров ... 220
8.1.3. Проверка влияния логического параметра ... 221
8.2. Усиление политик SELinux ... 222
8.2.1. Список модулей политики ... 222
8.2.2. Загрузка и удаление модулей политики ... 223
8.2.3. Создание политик с использованием программы audit2allow ... 224
8.2.4. Использование говорящих за себя наименований для модулей политики ... 226
8.2.5. Использование макрокоманд посреднической политики с программой audit2allow ... 227
8.2.6. Использование скрипта selocal ... 228
8.3. Создание модулей политик по специальным требованиям ... 229
8.3.1. Создание модулей SELinux с помощью исходного языка описания политик ... 230
8.3.2. Создание модулей SELinux с помощью посреднического стиля описания политик ... 231

8.3.3. Создание модулей SELinux с помощью обобщенно-промежуточного языка ... 232
8.3.4. Добавление описаний для параметров безопасности файла ... 232
8.4. Создание ролей и пользовательских типов допустимого набора функций ... 233
8.4.1. Создание файла pgsql_admin.te ... 233
8.4.2. Создание прав пользователя ... 234
8.4.3. Предоставление доступа для взаимодействия с командным интерфейсом ... 235
8.4.4. Формирование структуры файлов пользовательской политики ... 236
8.5. Создание новых типов для приложений ... 237
8.5.1. Создание файлов mojomojo.* ... 238
8.5.2. Создание интерфейсов политик ... 239
8.5.3. Создание структуры файлов политики для приложений ... 240
8.6. Замена существующих политик ... 241
8.6.1. Замена политик Red Hat Enterprise Linux ... 241
8.6.2. Замена политик в Gentoo ... 243
8.7. Другие варианты усиления политики безопасности ... 244
8.7.1. Создание типов SECMARK по специальным требованиям ... 244
8.7.2. Регистрация попыток доступа в журнале событий ... 245
8.7.3. Создание типов, соответствующих специальным требованиям ... 245
8.8. Заключение ... 246

Глава 9. Анализ поведения политики ... 248

9.1. Одноступенчатый анализ ... 248
9.1.1. Использование различных файлов политик SELinux ... 249
9.1.2. Отображение информации об объектах политики ... 249
9.1.3. Применение утилиты sesearch ... 251
9.1.4. Запрос разрешающих правил ... 251
9.1.5. Запрос сведений о правилах преобразования типов ... 251
9.1.6. Запрос правил для других типов ... 252
9.1.7. Запрос правил, связанных с ролями ... 253
9.1.8. Отображение данных с помощью графической программы apol ... 253
9.2. Анализ преобразований типов процессов ... 257
9.2.1. Использование программы apol ... 258
9.2.2. Использование программы sedta ... 259
9.3. Анализ потоков информации ... 261
9.3.1. Использование программы apol для анализа потоков информации ... 262
9.3.2. Использование программы seinfoflow для анализа потоков информации ... 265
9.4. Другие виды анализа политик ... 266
9.4.1. Сравнение политик при помощи sediff ... 266

9.4.2. Анализ политик при помощи sepolicy 267
9.5. Заключение 268

Глава 10. Частные случаи настройки защиты 269
10.1. Усиление защиты веб-серверов 269
10.1.1. Описание условий работы 270
10.1.2. Настройка для установки нескольких экземпляров программ 271
10.1.3. Создание категорий SELinux 272
10.1.4. Выбор необходимых параметров безопасности 273
10.1.5. Включение администраторов в систему защиты 275
10.1.6. Управление работой веб-сервера 275
10.1.7. Работа с обновлением содержания 277
10.1.8. Настройка сети и правил межсетевого экрана 279
10.2. Защита командно-строчного интерфейса 279
10.2.1. Разделение SSH на несколько экземпляров 280
10.2.2. Обновление правил работы сети 281
10.2.3. Изменение корневого каталога для отдельной программы 282
10.2.4. Предоставление параметров безопасности пользователю в зависимости от способа доступа 283
10.2.5. Настройка правил для SSH 285
10.2.6. Включение многопользовательского режима использования 286
10.3. Общий доступ к файлам через сетевую файловую систему NFS 287
10.3.1. Базовая настройка службы NFS 287
10.3.2. Включение поддержки NFS на стороне защищенного клиента 288
10.3.3. Настройка правил безопасности для NFS на сервере 288
10.3.4. Подключение общих сетевых ресурсов с различными параметрами безопасности 289
10.3.5. Работа с промаркированной сетевой файловой системой 290
10.3.6. Сравнение файлового сервера Samba с сетевой файловой системой NFS 291
10.4. Заключение 292

Предметный указатель 293

Об авторе

Свен Вермейлен (Sven Vermeulen) – постоянный участник различных проектов свободного программного обеспечения и автор многочисленных руководств и ресурсов в интернете. Свой первый опыт в разработке свободного программного обеспечения он получил в 1997 году и с тех пор только развивал и совершенствовал свои навыки в этом направлении. В 2003 году он присоединился к проекту Gentoo Linux как разработчик документации и затем выступал в разных ролях, включая такие, как доверенное лицо фонда Gentoo, член совета, руководитель проекта по различным инициативам в области документирования, а также руководитель проектов по усилению системой защиты SELinux операционной системы Gentoo и системному интегрированию.

В течение этого времени Свен получил экспертные знания как на уровне операционной системы, так и на уровне серверного прикладного программного обеспечения. Он использовал свой интерес к безопасности, чтобы направлять свои проекты, связанные с формированием руководств, в область защиты информации. Для этой цели им стали применяться:

- языки описания политики безопасности, механизмов контроля и результаты оценки SCAP (*Security Content Automation Protocol* – Протокол автоматизации информационного обеспечения безопасности);
- средства контроля разграничения доступа, реализованные в SELinux;
- аутентифиакция при помощи средства обеспечения защиты PAM (*Pluggable Authentication Modules* – подключаемые модули аутентификации);
- программные межсетевые экраны
- и многое другое.

Для SELinux Свен внес несколько вариантов политик в проект Посреднической политики (Reference Policy project), и он является активным участником проектов по разработке политик и пользовательского пространства.

В своей ежедневной работе Свен – архитектор информационных технологий в одном из европейских финансовых институтов, а также самостоятельно действующий инженер и консультант. Создание безопасных инфраструктур (и сопутствующая архитектурная интеграция) является, конечно, важной частью его работы. Свое образование – степень магистра компьютерной инженерии – Свен Вермейлен получил в Бельгии, в университете города Гент. Вторая степень была получена в магистратуре организации INNOCOM (https://www.inno.com) по специальности информационно-коммуникационной архитектуры предприятия. Работал инженером инфраструктуры веб-приложений.

Свен является основным автором книги *Gentoo Handbook* (Справочник по Gentoo), которая охватывает вопросы установки и настройки операционной системы Gentoo на нескольких архитектурах. Он также автор публикации в интернете *Linux Sea* (Море Linux) – http://swift.siphos.be/linux_sea, которая является

базовым введением в операционную систему Linux для начинающих системных администраторов. Кроме того, он является автором таких книг издательства Packt Publishing, как *SELinux System Administration* (Администрирование системы защиты SELinux, 1-е изд.) и *SELinux Cookbook* (Книга готовых рецептов для системы защиты SELinux).

Я хотел бы поблагодарить сообщество разработчиков ПО с открытым исходным кодом и свободного программного обеспечения за его бесконечное стремление создавать отличные программное обеспечение, документацию, настоящие произведения искусства и сервисы. Именно благодаря этому стремлению компании и организации во всем мире пользуются предоставляемыми средствами высокого качества со всей свободой, которую дает это программное обеспечение. В частности, я хотел бы поблагодарить сообщество Gentoo, поскольку оно предоставляет отличные метадистрибутив и операционную систему. Люди, которых я там встречаю, – все они очень высоко мотивированные, опытные и/или эксперты в определенных областях. Присутствие в сообществе заставляет меня стремиться узнать больше.

О рецензентах

Дэвид Куигли (David Quigley) начал свою карьеру исследователем компьютерных систем в Национальной исследовательской лаборатории по обеспечению информационной безопасности при Агентстве национальной безопасности США, где он работал в качестве члена команды SELinux. Дэвид возглавил работы по проектированию и реализации маркировки сетевой файловой системы NFS в SELinux. До этого участвовал в сообществе открытого программного обеспечения, поддерживая кодовую базу проекта вспомогательной файловой системы Unionfs 1.0, и вносил свой вклад в различные другие проекты. Дэвид выступал с докладами на таких конференциях, как Оттавский симпозиум по Linux, семинар по StorageSS, LinuxCon, а также на нескольких локальных собраниях групп пользователей Linux, где темы презентаций включали хранение, файловые системы и безопасность. В настоящее время Дэвид работает инженером по ядру файловой системы ZFS в отделе высокопроизводительных данных в Intel. И ранее рецензировал книгу *SELinux Cookbook*, опубликованную издательством Packt.

Я хотел бы поблагодарить мою замечательную жену Кэти за все, что она делает, чтобы у меня было время заняться такими вещами, как обзор этой книги, и поездками, связанными с презентациями о SELinux. Она – радость моей жизни и помогла мне стать тем, кем я являюсь сегодня. Я также хотел бы поблагодарить моих детей Зои Джейн и Кэролайн, которые напоминают нам о том, что нужно любить и ценить время, которое мы проводим вместе с семьей.

Сэм Уилсон (Sam Wilson) – старший инженер по системам и безопасности, недавно увлекся конструированием радиотехнического оборудования и специализируется на Red Hat Enterprise Linux. Благодаря обширным знаниям в области безопасности, охватывающим микросервисы, инфраструктуру, и в организации работы команды при обеспечении коллективных целей по обеспечению безопасности (*SecOps*) к Сэму регулярно обращаются за наставничеством и советами по SELinux организации, с которыми он сотрудничает и с которыми работает. Сэм активно участвует в сообществах GNU/Linux с начала 2007 года и добровольно посвятил себя работе над проектами NTFreenet, Darwin Community Arts, Ansible и Fedora.

Сэм является автором сайта https://www.cycloptivity.net, а также работает с командой интеллектуальной безопасности Atlassian Security Intelligence над визуализацией, эксплуатационной безопасностью и средствами управления для поддержки и защиты клиентов Atlassian в облаке.

Предисловие

Безопасное состояние операционной системы или какой-либо службы является результатом многоуровневого подхода к обеспечению безопасности. Системы могут быть защищены от внешнего мира при помощи межсетевых экранов, операционные системы должны регулярно получать обновления безопасности, работающие службы должны быть правильно настроены, необходимо разделять обязанности для конечных пользователей и т. д.

Контроль доступа – это еще один уровень обеспечения защиты, который администраторы должны учитывать. Благодаря системе защиты SELinux (Security Enhanced Linux – повышенная безопасность Linux) в экосистеме Linux появилась надежная и, что удобно, встраиваемая система принудительного контроля доступа. Некоторые дистрибутивы включают SELinux по умолчанию, другие позволяют администраторам включать ее самим. Android, одна из самых популярных операционных систем для мобильных устройств, также использует технологию SELinux под названием SEAndroid.

Но, в отличие от Android, где пользователи и приложения находятся под жестким контролем и где недопустимы отклонения в настройке и организации файлов и ресурсов, настольные компьютеры, рабочие станции и серверы, которые реализуют Linux, имеют большее разнообразие в возможностях управления защитой. В результате настройка SELinux в этих системах требует больше знаний о том, что такое SELinux, как он работает и как его можно использовать.

В этой книге мы обсуждаем, что такое SELinux и как он встроен в операционную систему Linux. Мы рассмотрим различные аспекты конфигурации SELinux и разберем несколько вариантов применения, которые используют сильные стороны SELinux для дальнейшего усиления безопасности системы и служб, размещенных на ней.

О чем эта книга

Глава 1 «*Фундаментальные концепции SELinux*» дает администраторам представление о том, что такое система защиты SELinux и как она взаимодействует на уровне ядра операционной системы Linux. Здесь объясняются различия в реализациях SELinux между дистрибутивами и описывается характерная для SELinux терминология, которая дальше будет часто использоваться по мере углубления в технологию SELinux.

Глава 2 «*Режимы работы и регистрация событий*» описывает различные состояния работы SELinux и показывает, где SELinux регистрирует свои события. Эта глава поможет администраторам разобраться с тем, как следует интерпретировать и анализировать эти события.

Глава 3 «*Управление учетными записями пользователей*» рассказывает администраторам, как управлять пользователями Linux и их правами, а также выполнять сопоставление этих пользователей с различными ролями, которые SELinux поддерживает с помощью собственной организации пользовательского пространства и подключаемых модулей аутентификации Linux. Кроме того, глава охватывает такую сущность, как категории защищаемой информации, реализованные в SELinux.

Глава 4 «*Домены как допустимые наборы функций для процессов и контроль доступа на уровне файлов*» знакомит администраторов с метками параметров безопасности SELinux. С тем, как эти метки хранятся в файловой системе или предоставляются другим ресурсам. В этой главе администраторы и конечные пользователи узнают, как устанавливать и обновлять метки параметров безопасности.

Глава 5 «*Контроль сетевого взаимодействия*» рассматривает стандартные службы сетевой безопасности, утилиту iptables и протокол IPSec с точки зрения их совместной работы с функциями защиты SELinux. Администраторы смогут научиться включать поддержку SELinux в этих службах безопасности и даже включать маркировку, выполняемую между распределенными по сети системами, при помощи таких методов, как Labeled IPSec и NetLabel/CIPSO.

Глава 6 «*Поддержка sVirt и Docker*» рассказывает, как компания Red Hat разработала технологию защищенной виртуализации (sVirt) и реализовала ее в двух системах виртуализации: библиотеке libvirt и контейнерах Docker. В этой главе объясняется, как настроить эти службы с помощью поддержки SELinux и контролировать перемещение ресурсов, используемых гостевыми системами виртуальной инфраструктуры или контейнерами.

Глава 7 «*D-Bus и systemd*» рассказывает о сферах влияния упомянутых системных служб уровня ядра и о том, как они используют правила SELinux для дальнейшего усиления безопасности своих собственных функциональных возможностей. Получив эти знания, администраторы смогут настроить защищенную работу службы межпроцессного взаимодействия D-Bus, а также управлять средствами доступа SELinux, применяемыми через подсистему инициализации systemd.

Глава 8 «*Работа с политиками SELinux*» посвящена настройке и управлению политиками SELinux. Она показывает, как можно создавать политики безопасности по заданным требованиям или даже заменять политику, официально предоставляемую в рамках дистрибутива.

Глава 9 «*Анализ поведения политики*» углубляется в инструменты анализа, которые позволяют инженерам и администраторам более детально рассматривать политику безопасности SELinux. С помощью этих инструментов можно получить наиболее полное представление о том, что политика в себя включает и как поведет себя в различных ситуациях.

Глава 10 «*Частные случаи настройки защиты*» описывает ряд распространенных случаев использования серверов, таких как веб-серверы и файловые серверы, и способы использования SELinux для их защиты. В этой главе рас-

сказывается, как можно изолировать пользовательское окружение с помощью SELinux и как администраторы могут построить защищенную многопользовательскую систему.

Что необходимо для понимания книги

Поскольку SELinux является компонентом для операционной системы Linux, то читателям желательно иметь ее в своем распоряжении вместе с установленной системой защиты SELinux. Процесс установки SELinux не входит в материал данной книги, по этому вопросу стоит обратиться к документации используемого дистрибутива. Кроме того, настройка системы защиты требует наличия привилегий администратора в системе.

Для кого предназначена эта книга

Эта книга предназначена для системных администраторов Linux, которые имеют достаточный опыт обслуживания систем Linux, хотят разбираться в технологии защиты SELinux и работать с ней. Кроме того, данная книга может быть полезна для архитекторов информационных систем, чтобы понять, как можно применять SELinux для повышения безопасности систем Linux и служб, работающих под управлением этой операционной системы для обеспечения нужд компании.

Соглашения

В этой книге вы найдете несколько стилей оформления текста, которые позволяют отличать одни виды информации от других. Вот примеры этих стилей и объяснение их значения.

Ключевые слова в тексте, имена таблиц базы данных, каталогов, файлов, расширения файлов, имена путей, адреса сайтов и данные, вводимые пользователем, выделены моноширинным шрифтом: «Мы выполняем это с помощью команды `semanage login`».

Блок текста, связанного с настройками, выводом команд и программами, выглядит в тексте книги так:

```
dbadm_r
  Dominated roles:
    dbadm_r
  Types:
    qmail_inject_t
    dbadm_t
    ...
    user_mail_t
```

Любая фраза командно-строчного интерфейса дополнительно выделяется жирным шрифтом:

```
# seinfo -amcs_constrained_type -x | grep virt_
```

Новые термины и **важные слова** выделены жирным шрифтом в тексте описания. Слова, которые пользователь может увидеть на экране (например, в меню или диалоговых окнах), отображаются в тексте книги примерно так: «После загрузки выберите пункт меню **Новый анализ** (New Analysis), для того чтобы начать работу с функциями анализа политики».

Предупреждения и важные замечания отмечаются таким значком.

Советы и подсказки выглядят так.

Тематические пояснения для русскоязычного издания.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Apress очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Глава 1

Фундаментальные концепции SELINUX

SELinux – система защиты, обеспечивающая повышенную безопасность в Linux. Она включает дополнительные меры защиты в операционную систему для большей целостности, доступности и конфиденциальности ее ресурсов.

В данном разделе будут рассмотрены следующие вопросы:

1) почему SELinux использует метки для идентификации ресурсов;
2) как качественно SELinux отличается от штатных систем контроля доступа Linux за счет наличия правил обеспечения безопасности;
3) каким образом правила контроля доступа повышают уровень защиты, распространяясь через файлы политик безопасности SELinux.

В конце главы будут рассмотрены различия между версиями SELinux, реализованными в дистрибутивах Linux.

1.1. Предоставление большей безопасности в Linux

Опытные администраторы систем, построенных на базе Linux, и проектировщики систем защиты знают, что им необходимо доверять пользователям и процессам в обслуживаемой системе. То есть администраторам требуется заручиться некоторой лояльностью со стороны пользователей и гарантиями для выполняемых процессов в системе, чтобы обеспечивался нужный уровень защищенности. Это частично связано с тем, что пользователи могут пытаться использовать уязвимости, найденные у запущенных программ, с которыми они должны работать «по долгу службы», но еще больший их вклад в снижение уровня угроз связан как раз с тем, что безопасное состояние системы зависит от непосредственного поведения пользователей. Дело в том, что пользователь с доступом к конфиденциальной информации (к такой, которая не должна быть предоставлена в общий доступ), сам же и управляет приложениями, в т. ч. запускает и совершает другие действия, которые оказывают влияние на защиту системы. Одним из механизмов стандартной защиты является дискреционный контроль доступа, который настраивается в соответствии с потребностями пользователя.

Механизм ***дискреционного контроля доступа*** DAC (*Discretionary Access Control*) в операционной системе Linux основывается на сопоставлении данных процесса, выполняемого пользователем (и/или группой пользователей), с информацией о разрешениях для пользователя (и/или группы), характерных для какого-либо файла, директории или другого управляемого ресурса. Рассмотрим файл `/etc/shadow`, который содержит информацию о паролях и именах пользователей некой локальной системы:

```
$ ls -l /etc/shadow
rw------- 1 root root 1010 Apr 25 22:05 /etc/shadow
```

Без дополнительных механизмов контроля доступа, имеющихся в наличии, этот файл является разрешенным для чтения и записи любым процессам, которыми владеет пользователь `root`, вне зависимости от целей процесса, преследуемых в системе. Файл `shadow` представляет типичный пример такого конфиденциального файла, который никто из пользователей не хотел бы видеть опубликованным или использованным недобросовестным образом. По сути, кто-то один имеет доступ к данному файлу и может скопировать его в какое-либо другое место, например в домашний каталог, по почте отправить на другой компьютер или попытаться выполнить атаку на защищенное криптографией значение пароля (*hash*) пароля, размещенное в этом файле.

Другой пример того, что стандартный механизм дискреционного контроля доступа требует доверия к пользователям, – это случай, когда в системе расположена база данных. Файлы базы данных сами по себе являются (очень хочется верить) доступными только для сеансовых пользователей, зарегистрированных в *системе управления базой данных* (Database management system – DBMS), и для пользователя операционной системы `root`.

Должным образом защищенные системы будут предоставлять доступ к этим файлам только хорошо проверенным пользователям (например, через команду `sudo`), позволяя им заменить свои эффективные пользовательские идентификаторы на идентификаторы пользователей базы данных или даже на пользователя с правами `root`, и это для строго заданного набора команд. Эти пользователи также могут анализировать файлы базы данных и получать доступ к потенциальной конфиденциальной информации в базе данных без входа через СУБД.

Однако пользователи операционной системы не являются единственной угрозой, из-за которой необходимо защитить систему. Многие фоновые программы (*software daemons*) запускаются с правами пользователя `root` или имеют необходимые для работы привилегии в системе. Ошибки внутри этих программ могут легко привести к утечке информации или даже к удаленному использованию уязвимостей. Программы резервирования, контроля выполнения, администрирования, планирования и им подобные программы: они все часто запускаются с высшими пользовательскими привилегиями, доступными в операционной системе Linux. Даже когда администратор не предоставляет привилегии пользователям, их взаимодействие со службами подразумевает потенциальный риск безопасности. А раз так, то пользователи продолжают

пользоваться своими расширенными правами для корректного взаимодействия с приложениями и обеспечения корректного функционирования системы в целом. Поэтому администратор вынужден строить безопасность системы на порядочности ее пользователей.

Рассмотрим теперь SELinux, который содержит дополнительный уровень контроля доступа, стоящий выше дискреционного механизма. SELinux предоставляет *мандатный контроль доступа* (Mandatory access control – MAC) системы, который нивелирует рассмотренные недостатки дискреционного контроля, предоставляя администратору полный контроль над тем, что является позволенным в системе, а что нет. Он достигает этого путем применения метода управления политиками, определяющими, являются или не являются процессы разрешенными к выполнению, а также путем приведения этих политик в жизнь через ядро операционной системы Linux.

Мандатные средства, которые контролируют доступ, приводятся в исполнение операционной системой и определяются исключительно правилами политики, задействованными системным администратором (или администратором безопасности). Пользователи и процессы не имеют прав на изменение правил безопасности, таким образом они не могут что-либо делать в части контроля доступа; защита больше не находится во власти свободы их действий.

Здесь слово «мандатный» так же, как и ранее слово «дискреционный», было выбрано для описания возможностей системы контроля доступа не случайно: оба термина являются известными в области информационной безопасности и имеют описания в других публикациях, включая стандарт «Критерии оценки доверенных компьютерных систем»[1] (известный также как «Оранжевая книга»), выпущенный Министерством обороны Соединенных Штатов Америки в 1985 году. Этот документ предшествовал стандарту «Общие критерии»[2], предназначенному для сертификации компьютерных систем.

Р В России данные термины по защите информации определяются и раскрываются в нижеприведенных и других документах:

1) Национальный стандарт Российской Федерации. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий;
2) руководящий документ ФСТЭК[3] «Защита от несанкционированного доступа к информации. Термины и определения». Утверждено решением председателя Гостехкомиссии России от 30 марта 1992 г.;
3) руководящий документ ФСТЭК «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации».

---

[1] Trusted Computer System Evaluation Criteria (TCSEC) 1985 [Электронный ресурс] // http://csrc.nist.gov/publications/history/dod85.pdf.

[2] Common Criteria (ISO/IEC 15408) [Электронный ресурс] // http://www.commoncriteriaportal.org/cc/.

[3] ФСТЭК – Федеральная служба технического и экспортного контроля.

1.1.1. Использование модулей безопасности Linux

Снова рассмотрим в качестве примера файл shadow. Система мандатного контроля доступа может быть сконфигурирована таким образом, чтобы позволять чтение и запись в файл только ограниченному набору процессов. В системах, настроенных подобным образом, пользователь, зашедший под учетной записью root, не имеет прямого доступа к файлу или даже к перемещению его. Он также не может изменить атрибуты файла:

```
# id
uid=0(root) gid=0(root)
# cat /etc/shadow
cat: /etc/shadow: Permission denied
# chmod a+r /etc/shadow
chmod: changing permissions of '/etc/shadow': Permission denied
```

P В данном примере автор выполняет проверку текущего пользователя и убеждается, что пользователь зашел под учетной записью root. После чего выполняет команду вывода на экран содержимого файла, содержащего защищенную форму представления паролей (hash) паролей, но в результате получает сообщение от системы, что доступ запрещен (Permission denied). Тогда, имитируя мышление взломщика, автор предполагает, что доступ запрещен стандартным механизмом дискреционного доступа, и выполняет команду изменения прав доступа к файлу, пытаясь получить разрешение на чтение для всех пользователей. В результате система снова пишет, что изменение прав доступа для данного файла запрещено.

Команда «id» выводит на экран некоторый используемый набор идентификационной информации для текущего пользователя. В данном случае приводится uid (сокр. от *user identifier* – идентификатор пользователя) и gid (сокр. от *group identifier* – идентификатор группы).

Команда cat выводит содержимое текстового файла /etc/shadow на экран.

Команда chmod выполняет изменение прав доступа к файлам и директориям. Параметр a (сокр. от *all* – все) определяет, что изменение прав будет для всех пользователей, а параметр r (сокр. от *read* – читать) указывает на то, что файл должен быть разрешен на чтение.

Невозможность изменения атрибутов файла достигается с помощью правил, которые описывают, при каких условиях содержимое файла может быть прочитано. Данные условия определяются в политике SELinux и загружаются, когда система проходит начальную загрузку. Само ядро Linux отвечает за соблюдение этих правил. Мандатная система контроля доступа легко включается в ядро Linux через его поддержку модулей безопасности Linux (Linux Security Modules – LSM):

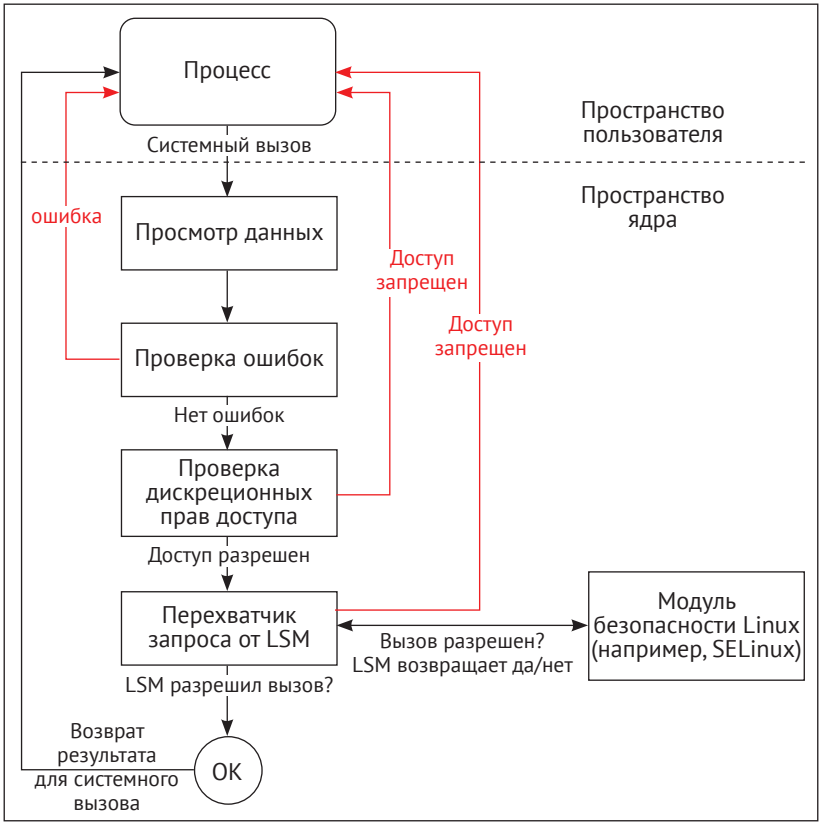

Рис. 1.1 ❖ Высокоуровневое описание процесса интеграции модулей безопасности в ядро Linux

1.1.2. Расширение возможностей стандартного дискреционного разграничения доступа

SELinux не изменяет реализацию механизма дискреционного контроля доступа, а также не может отменять правила, установленные механизмом контроля доступа самого Linux. Если стандартная система (без SELinux) исключает кому-то персональный доступ, тогда SELinux не может изменить это решение. Это связано с тем, что модули безопасности системы защиты (LSM[1]) срабатывают *после* включения стандартного дискреционного механизма, что специально запроектировано в рамках реализации проекта LSM.

[1] LSM [Linux Security Module] – модули безопасности операционной системы Linux.

Например, если необходимо предоставить дополнительному пользователю доступ к файлу, то нельзя добавить политику SELinux, которая решит эту задачу. Вместо нее необходимо найти среди других возможностей операционной системы такие, которые используют списки контроля доступа (ACLs[1]) стандарта, обеспечивающего переносимость приложений между различными версиями операционных систем (POSIX[2]). При помощи команд `setfacl` и `getfacl` (распространяются в составе пакета с утилитами для управления доступом ACL) пользователь может установить дополнительные разрешения на файлы или директории, предоставляя выбранный ресурс дополнительным пользователям или группам.

В качестве примера давайте предоставим пользователю `lisa` доступ к некоторому файлу с правом его чтения и записи, используя команду `setfacl`:

```
$ setfacl -m u:lisa:rw /path/to/file
```

Команда `setfacl` (сокращение от фразы «To set file access control list» – «Применить список управления доступом к файлу»), запускаемая из командной строки, предназначена для установки прав доступа к файлу или каталогу. В приведенном примере команда запущена с опцией `-m` (от слова modify – модифицировать), указывающей необходимость внесения изменений в некий файл, для которого приведен полный путь размещения в файловой системе `/path/to/file`. С одной стороны, данный путь является условным и образуется совокупностью слов, формирующих фразу «path to file» (путь к файлу), а с другой – является вполне допустимой последовательностью вложенных каталогов, где каталог `to` вложен в каталог `path`, а имя самого файла – `file`.

Так как команда распознает разные форматы записи списка контроля доступа (ACL), то в примере приведен тот, который необходим для решения задачи по добавлению пользователя: [u[ser]:]uid [:perms].

В данном формате:

1) `u` – сокращение от *user* (пользователь) определяет, что далее будет указан уникальный идентификатор конкретного пользователя – *uid* (user identifier – идентификатор пользователя, имеющий числовое значение). В примере идентификатор заменен символьным именем (логином) – lisa;
2) `perms` – сокращение от *permissions* (полномочия), определяющие права комбинациями обозначений из списка:
   - `r` (read) – иметь доступ на чтение файла или каталога;
   - `w` (write) – иметь доступ на запись в файл или каталог;
   - `x` (eXecute) – иметь доступ на запуск (исполнение) файла.

В примере указана комбинация параметров «rw», позволяющая прочитать содержимое файла и изменить его.

Для успешного выполнения команды необходимо, чтобы была включена поддержка в файловой системе списков контроля доступа (ACL) при подключении (монтировании)

---

[1] Access Control List [Список контроля доступа] – список контроля доступа (управления доступом).

[2] POSIX [Portable Operating System Interface for computer environments (for Unix)] – интерфейс переносимой операционной системы для компьютерного окружения (сред UNIX).

устройства. Например, в гипотетически возможном содержании файла /etc/fstab следует добавить запись `acl` следующим образом:

```
/dev/sda1   /   reiserfs  noatime,acl  0 2
```

В файловой системе *xfs* поддержка списков контроля доступа включена по умолчанию.

Подобным образом, для того чтобы увидеть примененный к файлу список управления доступом, в формате стандарта (POSIX) следует использовать такую команду:

```
$ getfacl /path/to/file
1: # file: file
2: # owner: swift
3: # group: swift
4: user::rw
5: user: lisa:rw
6: group::r--
7: mask::r--
8: other::r--
```

Команда `getfacl` (сокращение от фразы «To get file access control list» – «Получить список управления доступом к файлу»), запускаемая из командной строки, предназначена для получения данных об установленных правах доступа для файла или каталога. В приведенном примере команда запущена с указанием полного пути размещения в файловой системе `/path/to/file`.

Из полученных сведений определяется имя объекта (в данном случае `file`), владелец (`swift`), а также владеющая объектом группа (`swift`), каждому из которых предусмотрено соответствующее обозначение: `# file`, `# owner` и `# group`.

В 4-й строке вывода определяются права пользователя-владельца (`swift`), позволяющие читать файл и записывать в него.

В 5-й строке вывода отображаются права доступа пользователя `lisa` – чтение и запись файла – то, что было установлено при помощи утилиты `setfacl`.

В 6-й строке вывода указаны права на чтение для группы `swift`, владеющей файлом.

В 7-й строке вывода, обозначенной словом `mask` (маска), указываются права, которые накладываются на права всех владеющих объектом групп и отдельно указанных пользователей, не считая владельца и так называемых «остальных» пользователей, т. е. которые не указаны явно. Эффективными будут те разрешения, которые совпадают с указанными в маске. Маска рассчитывается автоматически, если не была конкретно задана в команде `setfacl`.

8-я строка определяет права всех остальных пользователей.

В приведенном примере у пользователя `lisa` фактическим (эффективным) правом является только чтение файла, согласно указанной маске.

1.1.3. Ограничение привилегий пользователя root

Стандартный механизм дискреционного контроля доступа в Linux доступен для всемогущего пользователя, имеющего имя `root`. В отличие от многих других пользователей системы, пользователь, вошедший под учетной записью `root`, имеет необходимые права для полного управления всей системой, начиная с важнейшего контроля доступа, управления системой регистрации событий и заканчивая изменениями пользовательских идентификаторов, на-

стройкой сети и многим другим. Это все обеспечивается благодаря концепции безопасности, называемой **capabilities**[1] (для общего представления об этих возможностях Linux посмотрите, что написано в руководстве: `man capabilities`). SELinux способен ограничить доступ к этим широким возможностям системы с большой точностью воздействия.

Благодаря этому избирательному разрешающему подходу SELinux даже пользователь `root` может быть ограничен без возникновения конфликтов в системе. Предыдущий пример неудавшегося обращения к файлу `/etc/shadow` является только одним примером деятельности, которую неограниченный в правах пользователь, такой как `root`, все-таки может не быть способным выполнить из-за того, что был применен механизм защиты SELinux.

Когда SELinux был добавлен в ядро операционной системы Linux, некоторые проектные замыслы по защите даже дошли до того, что предоставили общий доступ с правами пользователя `root` к командной оболочке системы, защищенной средствами SELinux, обращаясь с просьбой к хакерам и другим исследователям в области безопасности скомпрометировать такую систему. Умение ограничивать пользователя `root` было доброжелательно встречено системными администраторами, которым иногда приходилось временно передавать пароль от этого пользователя или доступ к командной оболочке, имеющей его права, другим пользователям (к примеру, администраторам базы данных), которым необходимы `root`-привилегии, когда их программное обеспечение выходит из строя. Благодаря SELinux администраторы могут теперь передавать командную командный интерфейс управления с правами пользователя `root` на время своего отсутствия с уверенностью в том, что пользователь имеет только необходимые ему права и неполные права системного администратора.

1.1.4. Сокращение воздействия уязвимостей

Если существует одно преимущество SELinux, на котором надо сделать акцент из-за частого неверного толкования, то им является именно способность снижения воздействий со стороны уязвимостей.

Хорошо написанная политика безопасности ограничивает прикладные программы таким образом, что разрешенные для них действия сведены к минимально необходимому набору функций. Такая модель наименьших прав (*least-privilege model*) гарантирует, что некорректное поведение приложения не только обнаруживается и регистрируется в журналах, но и предотвращается. Многие уязвимости приложений могут быть использованы для выполнения заданий, которые приложение не намеревалось выполнять. Когда такое случается, SELinux способен это предотвратить.

Однако существует два заблуждения о возможностях этой системы защиты препятствовать реализации таких угроз, и одно из них связано с реализацией политик, а другое – с контролем использования уязвимостей приложений.

---

[1] В русскоязычном варианте: «возможности», «способности», «мощность».

Если политика не написана с учетом модели наименьших прав, тогда система защиты может расценивать такое нестандартное поведение как нормальное и позволять действиям выполняться. Авторам такой политики следует иметь в виду, что их правила должны быть очень точно детализированы. К сожалению, такой подход к написанию политик очень затратный по времени, т. к. существует более 80 классов и более 200 разрешений, используемых в системе защиты, а в правилах политики необходимо учитывать все эти классы и разрешения для каждого взаимодействия между объектами и субъектами доступа. Вследствие чего тексты политик имеют тенденцию к запутанности и сложности поддержания.

Некоторые авторы создают более либеральные политики, чем это в действительности необходимо, которые могут допустить успешное использование уязвимости, реализованное через действие, являющееся неожиданным поведением даже с точки зрения самой программы. Есть такие политики приложений, которые прямо отмечены как *неограниченные* (unconfined), демонстрируя, что они очень либеральны в своих разрешениях. Разработчики операционной системы Red Hat Enterprise Linux даже исходно создают политики приложений как полностью разрешающие действия субъектов по отношению к объектам и включают усиление контроля доступа для приложений только после выхода нескольких версий (и дополнительного тестирования).

Второе заблуждение – это контроль такой защитой использования злоумышленниками имеющихся уязвимостей.

Если некая уязвимость приложения позволяет не прошедшему аутентификацию пользователю использовать возможности этого приложения, как если бы он был авторизован, тогда SELinux не будет принимать участия в снижении воздействия уязвимостей; система защиты только отмечает поведение самого приложения, но не внутренних действий приложения.

До тех пор, пока приложение ведет себя как ожидается (например, при доступе к своим собственным файлам и не шурует кругом в других файловых системах), система защиты будет успешно позволять действиям совершаться.

И только когда приложение начинает вести себя беспорядочно, система защиты останавливает эксплуатацию программы. Уязвимости, такие как удаленное выполнение команд (remote command execution – RCE) в связке с приложениями, которые не должны выполнять случайные команды (например, системы управления базами данных или веб-сервера, исключающие CGI-подобную функциональность), будут предотвращены, несмотря на то что атаки в стиле *sql-инъекций* или *перехвата управления* являются неконтролируемыми через политики SELinux.

1.1.5. Включение возможностей SELinux в операционной системе

Включение возможностей дополнительной системы защиты в операционной системе Linux – это не просто вопрос включения модулей безопасности (LSM) в ядро операционной системы.

Реализация системы защиты включает в себя следующее:

1) базовую подсистему, реализованную в ядре ОС через модули безопасности (LSM);
2) библиотеки, используемые приложениями, которые необходимы для взаимодействия с SELinux;
3) утилиты, используемые администраторами для взаимодействия с SELinux;
4) политики, которые, собственно, определяют контроль доступа.

Библиотеки и утилиты объединены посредством проекта сообщества SELinux (https://github.com/SELinuxProject/selinux/wiki).

P Это основной репозиторий (место хранения и поддержания) пользовательских библиотек и инструментов SELinux.

Функции программного обеспечения, предоставляемые данным проектом, дополняются возможностями SELinux, интегрированными в ядро ОС Linux, и используются дистрибутивами операционной системы.

Инструменты предоставляют следующие возможности:

1) компиляция политики – низкоуровневые инструменты, которые выполняют преобразование языка политики SELinux, основанного на тексте, в формат, используемый ядром для приведения политики в действие;
2) управление политикой – инструменты (такие как `semodule` и `semanage`) и библиотеки (такие как `libsemanage`), используемые для установки, удаления и обновления политик SELinux на работающих системах;
3) разработка политики – инструменты, нацеленные на создание и обновление политик (например, `audit2why` и `audit2allow`);
4) сервисы SELinux – библиотеки (например, `libselinux`) для приложений, которые должны быть осведомлены о SELinux или исполнять решения в части контроля доступа при помощи SELinux (например, Dbus);
5) утилиты SELinux – низкоуровневые утилиты (такие как `setenforce` и `restorecon`) для управляющей и SELinux-поддерживающей систем.

Программное обеспечение может быть получено как в виде протестированной версии продукта, так и в виде версий из рабочего репозитория.

Вслед за прикладными программами сообщества и библиотеками различные компоненты операционной системы Linux обновляются со специфическим SELinux-кодом, охватывающим подсистему инициализации и некоторые утилиты ядра.

Так как непросто выбрать, что именно надо включить в поставляемый SELinux, дистрибутивы операционных систем, которые поддерживают эту систему защиты, обычно поставляются с предопределенными и загруженными компонентами: операционные системы Fedora и Red Hat Enterprise Linux (с их ответвлениями типа CentOS и Oracle Linux) являются хорошо известными примерами. Некоторые дистрибутивы могут иметь SELinux, не автоматически включенный, но при этом поддерживать легкую установку дополнительных пакетов (как в случае с операционными системами Debian и Ubuntu), а другие могут иметь хорошо документированные способы приведения системы к защищенному варианту средствами SELinux (например, Gentoo и Arch Linux).

В этой книге примеры будут даваться для ОС Gentoo и ОС Red Hat Enterprise Linux версии 7.2. Будут рассматриваться обе операционные системы, потому что они имеют различия в деталях реализации, позволяющие нам продемонстрировать полный потенциал SELinux.

1.2. Маркировка всех ресурсов и объектов

Когда SELinux должен решить, следует ли позволить определенное действие или запретить его, он принимает его, основываясь на параметрах безопасности *context* как субъекта (инициирующего действие), так и объекта (являющегося целью действия). Эти параметры (или его части) указаны в правилах политики, которую применяет SELinux.

Параметры безопасности процесса – это то, что идентифицирует процесс в SELinux. SELinux не имеет представления о принадлежности процессов Linux и, будучи однажды запущенной, не заботится о том, как процесс называется, какой идентификационный номер (ID) имеет и с какой учетной записью связан. Все, что ей надо знать, – это каковы параметры безопасности у процесса, что и представлено пользователям и администраторам как **метка**. *Метка* и *параметры безопасности* часто используются как взаимозаменяемые, и хотя здесь есть небольшая разница (поскольку одно понятие является отображением другого), мы не будем долго на этом задерживаться.

Давайте посмотрим на пример метки: параметры безопасности обычного пользователя (вы можете попробовать это сами, если находитесь во включенном режиме SELinux):

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

Команда `id`, которая дает информацию о текущем пользователе, представлена здесь ключом `-Z` (принятый общий ключ для отображения дополнительной информации по безопасности, полученной из подсистем защиты, основанных на LSM). Он показывает нам параметры безопасности текущего пользователя (в действительности параметры самого процесса идентификации в момент его выполнения). Как мы можем видеть, параметры безопасности имеют линейную структуру строки и выглядят так, как если бы у них было пять полей (на самом деле это не так, у них четыре поля, последнее из них просто содержит знак двоеточия).

Разработчики SELinux решили использовать метки вместо метаданных реального процесса и файла (или другого ресурса) для своего механизма контроля доступа. Это отличается от других систем защиты, реализующих мандатное разграничение доступа, например таких, как AppArmor, которые используют путь исполняемого файла (и это имя процесса) и пути ресурсов для управления проверками доступа. Решение сделать SELinux системой, основанной на механизме мандатного контроля доступа, применяющем метки, было принято по следующим нескольким причинам:

- использование путей могло бы быть более легким для понимания администраторами, но оно не позволяет содержать информацию о параметрах безопасности, закрытой внутри ресурса. Если файл или каталог перемещается, или перемонтируется, или процесс имеет другой вид наименования, отличающийся от имени файла, тогда инструменты контроля доступа могли бы повести себя иначе, поскольку обращаются к пути вместо файла. При помощи параметров, основанных на метках, эта информация сохраняется, и система продолжает контролировать ресурс правильно;
- параметры безопасности очень хорошо раскрывают цель процесса. Приложение с одним и тем же бинарным кодом может работать с различными параметрами безопасности, в зависимости от того, как оно было запущено. Значения параметров (которые были выше показаны в результате работы команды `id -Z`) содержат именно то, что необходимо администратору. Вместе с этим он знает, что права есть у каждого из запущенных экземпляров, но он может также проследить, как конкретный процесс мог быть запущен и что является у него целью;
- параметры безопасности также создают обобщение самого объекта. Мы их используем для разговора о процессах и файлах, но параметры безопасности являются еще прикладными для таких небольших сущностей, как каналы межпроцессного взаимодействия или объектов баз данных. В то время как идентификация, основанная на адресе размещения в файловой системе, сможет быть полезна только тогда, когда можно этот адрес (путь) указать для защищаемого объекта.

В качестве примера рассмотрим следующие политики:

- предоставление фоновым процессам `httpd`, обеспечивающим работу с протоколом http, связывание с портом 80 протокола TCP;
- предоставление процессам, промаркированным как `httpd_t`, взаимодействия с портами протокола TCP, помеченными как http_port_t.

В первом примере мы не можем легко повторно использовать эту политику, когда процесс веб-сервера не использует исполняемый файл httpd (может быть, потому что он был переименован или не является Apache, а каким-нибудь другим веб-сервером) или когда мы хотим выполнить http-доступ на другой порт. В случае же метода маркировки исполняемые файлы могут носить названия вроде `apache2` или `MyWebServer.py`, в то время как процессы их могут иметь маркировку `httpd_t`, которой будет оперировать политика безопасности. Точно так же обстоит дело и с определениями портов: можно промаркировать порт `8080` с помощью типа `http_port_t` и таким образом позволить веб-серверам подключаться к этому порту.

1.2.1. Описание параметров безопасности

Перейдем к параметрам безопасности: в SELinux применяется по крайней мере три, а иногда и четыре значения. В качестве примера посмотрим на па-

раметры безопасности веб-сервера Apache:

```
$ ps -eZ | grep httpd
system_u:system_r:httpd_t:s0 511 ? 00:00:00 httpd
```

Как мы можем видеть, с процессом веб-сервера связан набор параметров безопасности, который содержит следующие поля:

- `system_u`: данное поле определяет *имя пользователя* (SELinux user), зарегистрированного в системе защиты;
- `system_r`: второе поле указывает название одной из разрешающих *ролей* (SELinux role), определенной в системе защиты и предоставленной субъекту/объекту доступа;
- `httpd_t`: поле представляет *тип* (SELinux type) (также поле известно как *домен* (domain) в случае, когда речь идет о процессе);
- `s0`: четвертое поле содержит сведения о *мандатной конфигурации* («sensitivity level»).

Понятие «домен» в данном случае применяется в качестве «*допустимого набора функций*». Далее будет сказано, что понятие домен распространяется не только на процесс, но и в принципе на субъект доступа. Само понятие «домен», как известно, имеет широкое распространение в информационных технологиях и может относиться как к адресному пространству интернета в качестве структурированного глобального адреса узла сети (см. документ РД 45.134–2000), так и к базам данных, серверам, ресурсам и другим объединениям. Учитывая, что слово имеет французские корни, означающие «владение», корректно провести параллель и сказать, что в данном случае домен – это те функции и возможности, которыми «владеет» пользователь, процесс, файл, ресурс с точки зрения системы защиты.

В русскоязычной технической литературе слово «sensitivity» практически не попадает под прямой перевод, подразумевая мандатный принцип разграничения доступа в соответствии с ГОСТ Р 50739–95 «Средства вычислительной техники. Защита от несанкционированного доступа информации. Общие технические требования», где мандатный принцип характеризуется назначением объектам и субъектам так называемых «уровней иерархической классификации» и «иерархических категорий». Далее в этой главе будет сказано, что параметр безопасности «sensitivity level» основывается на классической модели Белла-ЛаПадула, в которой рассматриваются уровни доступа, и на разделении информации по категориям. В данном же разделе стоит уточнить, что уровнями могут быть следующие наименования: «открыто», «конфиденциально», «строго конфиденциально» и др., в зависимости от того, как их определит правообладатель. А категории информации отделяют сведения, например, бухгалтерские от сведений отдела кадров. При этом бухгалтерские и сведения отдела кадров могут быть открытыми, конфиденциальными и т. п. Тогда бухгалтеры будут иметь доступ к своим материалам, а работники отдела кадров – к своим, и смешиваться эти данные не будут. В свою очередь, параметр безопасности «sensitivity level» объединяет в себе «уровни доступа» и «категории», определяя конкретные варианты комбинаций из их возможного множества. Например, на рис. 1.2 показана следующая запись «`s0-s0:c0.c1023`», которая подразумевает наличие доступа (например, у какого-то пользователя) ко всем 1024 категориям информации, в которых сама информация имеет низший уровень доступа `s0`. Но можно дать пользователю доступ и ко всем уровням и ограничить его в категориях, оставив, допустим, толь-

ко одну, и таким образом выполнять администратору высокоточную настройку доступа к данным. И от того, как он ее настроит, зависит способность получения (восприятия) тем же пользователем полного объема данных или какой-то большой/сокращенной части целого. Вот эта способность и отражена, по задумке авторов, в термине «sensitivity level», который дословно можно перевести как «уровень восприятия». Говоря о русскоязычном соответствии, следует учесть, что наиболее часто, когда речь идет о мандатном уровне, подразумевают только уровень доступа, но не категорию информации, характерную мандатному *принципу* тоже. В связи с этим наиболее корректно, говоря далее в тексте про 4-й параметр безопасности, использовать понятие «*мандатная конфигурация*», подразумевающее сочетание «уровней доступа» и «категорий информации».

Структура набора из четырех параметров может быть изображена, как показано на рис. 1.2.

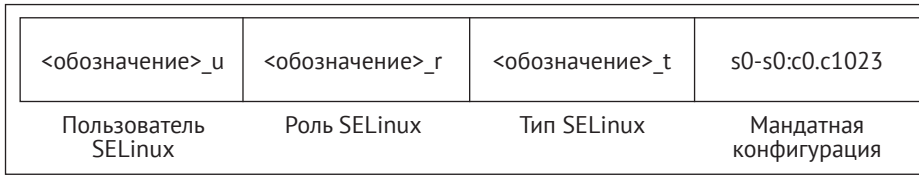

Рис. 1.2 ❖ Структура параметров безопасности SELinux, формируемая в результате использования команды `id -Z`, в качестве примера

Когда мы работаем с SELinux, то «параметры безопасности» – это практически все, что нам нужно. В большинстве случаев третий параметр (называемый «доменом», или [функциональным] «типом») является наиболее важным, т. к. многие правила политики системы защиты (порядка 99 %) состоят из правил, определяющих, каким образом осуществляется взаимодействие между двумя такими «типами» (без обращения внимания на «роли», «пользователей защиты» или «мандатные конфигурации»).

Параметры безопасности как и атрибуты модулей безопасности Linux (LSM) предъявляются в пользовательское пространство стандартизированным образом – совместимой с множественными реализациями модульной защитой LSM, позволяющей конечным пользователям и приложениям легко запросить конкретные параметры безопасности. Характерным местом, где эти атрибуты представлены, является псевдофайловая система `/proc`.

Внутри каждого места размещения процесса (`/proc/<pid>`) мы найдем поддиректорию, которая называется `attr`, внутри которой могут быть найдены следующие файлы:

```
$ ls /proc/$$/attr
current    fscreate    prev
exec       keycreate   sockcreate
```

Все эти файлы, если их открыть в режиме чтения, отобразят либо отсутствие данных, либо параметры безопасности SELinux. Если какой-то конкретный файл не содержит никаких данных, то это означает, что рассматриваемому

приложению явно не заданы параметры безопасности для этого конкретного аспекта работы и они будут получены либо из определенной политики, либо унаследованы от родительского процесса.

Назначение у этих файлов следующее:

- файл `current` содержит текущие параметры безопасности для данного процесса;
- файл `exec` содержит параметры безопасности, которые будут назначены дочерним процессам от текущего. Обычно данный файл является пустым;
- файл `fscreate` содержит параметры безопасности, которые будут назначены некоему файлу, после внесения в него изменений данным приложением. Обычно этот файл является пустым;
- файл `keycreate` содержит параметры безопасности, которые будут назначены ключам, временно хранимым (*cached*) данным приложением в ядре. Обычно этот файл является пустым;
- файл `prev` содержит *предшествующий* текущему набор параметров безопасности для определенного процесса. Обычно это параметры родительского приложения;
- файл `sockcreate` содержит параметры безопасности, которые будут назначены следующему сокету, созданному посредством данного приложения. Обычно данный файл является пустым.

Если приложение имеет множество подзадач, тогда такая же информация будет доступна в директории каждой подзадаче по следующему пути: `/proc/<pid>/task/<taskid>/attr`.

1.2.2. Принудительный доступ посредством типов функциональных ограничений

Тип функциональных ограничений (третья часть параметров безопасности) процесса (называемый **доменом**) является основой тщательного контроля доступа этого процесса по отношению как к нему самому, так и к другим типам (которые могут быть применимы к процессам, файлам, сокетам, сетевым интерфейсам и многому другому). В большинстве литературы о SELinux механизм контроля доступа, основанный на метках, является настолько хорошо настраиваемым, что можно было сказать, что сам SELinux – это система обязательного **соблюдения типов** допустимого набора функций. Когда какие-то действия являются запрещенными, наиболее вероятной причиной этому является тщательный контроль доступа на уровне типа допустимого набора функций (в т. ч. отсутствие его корректной настройки).

С помощью механизма соблюдения типа SELinux является способным контролировать, что позволено делать любому приложению, основываясь на том, как оно получает управление (команду на выполнение): веб-сервер, который запущен в диалоговом режиме пользователем, будет иметь тип, отличный от

веб-сервера, запущенного через систему init[1], даже если исполняемый файл процесса и путь размещения являются теми же самыми. Веб-сервер, запущенный системой init, является наиболее вероятно заслуживающим доверия (и, таким образом, веб-серверу позволено выполнять все функции, необходимые для его полноценной работы), в то время как веб-сервер, запущенный вручную, является наименее вероятно подлежащим рассмотрению в качестве образца нормального поведения, и раз так, то он будет иметь отличающиеся привилегии.

Большинство средств SELinux будут сосредоточены на типах допустимого набора функций. Даже когда тип – просто третий компонент параметров безопасности, он является наиболее важной сущностью для многих администраторов. Многие документы будут даже просто рассказывать о типах, таких как http_t, охотнее, чем обо всех параметрах безопасности в целом.

Взгляните на следующий процесс фоновой программы обеспечения межпроцессного взаимодействия dbus-daemon:

```
# ps -eZ | grep dbus-daemon
system_u:system_r:system_dbusd_t 4531 ?       00:00:00 dbus-daemon
staff_u:staff_r:staff_dbusd_t    5266 ?       00:00:00 dbus-daemon
```

В данном примере первый процесс dbus-daemon является фоновой программой системы D-Bus, запущенной с типом, соответственно названным system_dbusd_t, тогда как другой процесс является запущенным с назначенным ему типом staff_dbusd_t. Будучи полностью одинаковыми, даже на уровне бинарного кода, они оба преследуют различные цели в системе и таким образом имеют разные назначения типов защиты. Затем SELinux использует этот тип для ограничения действий, разрешенных процессу по отношению к другим типам, включая и то, как system_dbusd_t взаимодействует с staff_dbusd_t.

Типы SELinux обусловлены в обозначении окончанием _t, хотя это и необязательно.

1.2.3. Распределение по ролям наборов функциональных ограничений

Роли SELinux (второй компонент параметров безопасности) позволяют системе защиты поддерживать контроль доступа. Несмотря на то что тип допустимого набора функций (*type enforcement*) является наиболее используемым (и наиболее известным) компонентом SELinux, контроль доступа, основанный на роли, является важнейшим методом, для того чтобы обеспечивать надежную систему защиты, в особенности от вредоносных пользовательских покушений. Роли SELinux определяют, какие разрешенные наборы функций (домены) процессов могут быть запущены вместе. Эти типы типы допустимого набора функций, назначаемые процессам (домены) в своей области параметров безопасности

[1] init – система инициализации, запускающая все остальные процессы после инициализации ядра. Является первым процессом пользовательского режима и отвечает за дальнейшую загрузку системы. – *Прим. перев.*

определяют разрешения. По существу, роли помогают определить, что именно некий пользователь (который имеет доступ к одной или более ролям) может и не может делать.

Роли SELinux обусловлены в обозначении окончанием _r. В большинстве систем, включающих SELinux, следующие роли исходно доступны для сопоставления с пользователями:

Таблица 1.1. Описание предустановленных ролей

Роли	Описание
user_r	Эта роль предназначена для ограниченных в действиях обычных пользователей: она позволяет использовать приложения, чти процессы запускаются только со специфическими типами. Привилегированные типы, включающие возможности для выбора другого Linux-пользователя, являются непозволительными для этой роли
staff_r	Эта роль предназначена для некритичных операций: она в основном ограничивает те же приложения, что и в первой роли, но с возможностью поменять роль. Эта роль предназначена для операторов, находящихся на работе (для того чтобы предоставлять этим пользователям роли с наименьшими привилегиями до тех пор, пока это возможно)
sysadm_r	Эта роль предназначена для системных администраторов: она является очень привилегированной, включающей различные системные задачи по администрированию. Однако определенные типы приложений для конечного пользователя могут не быть поддержаны (особенно если эти типы являются предназначенными для потенциально уязвимого или непроверенного программного обеспечения) в целях сохранения системы, защищаемой от зловредных воздействий
secadm_r	Эта роль предназначена для администраторов безопасности: позволяет менять политику SELinux и выполнять действия по управлению данной системой защиты. Роль в основном используется, когда разделение обязанностей является необходимым между системным администратором и службой управления системными политиками
system_r	Данная роль предназначена для фоновых системных процессов: является наиболее полно привилегированной для поддержания различных типов защиты скрытых от пользователя процессов (*daemon*) и процессов операционной системы. При этом типы защиты приложений, предназначенные для программ конечных пользователей, а также типы, обеспечивающие администрирование, являются недоступными из этой роли
unconfined_r	Данная роль (в переводе «неограниченная») предназначена для конечных пользователей: она предоставляет конечное количество типов, но эти типы являются очень привилегированными, так как ориентированы на выполнение любых приложений, запущенных каким-либо пользователем в более или менее свободной манере (без ограничений правилами SELinux). Данная роль как таковая является доступной, исключительно если системный администратор хочет защитить конкретные процессы (часто выполняемые в фоновом режиме), оставляя в покое остальные системные операции почти не затронутыми защитой SELinux

В зависимости от дистрибутива другие роли могут также поддерживаться, например `guest_r` и `xguest_r`. Для того чтобы узнать больше о поддерживаемых ролях, лучше всего обратиться к документации конкретного дистрибутива. Некий обзор доступных ролей может быть предоставлен посредством команды `seinfo`, являющейся частью пакета `setools-console` в RHEL или `app-admin/setools` в Gentoo.

```
# seinfo --role
Roles: 14
  auditadm_r
  dbadm_r
  ...
  unconfined_r
```

1.2.4. Разделение пользователей по ролям

Пользователи, зарегистрированные в системе защиты SELinux (определяемые в первой части параметров безопасности), отличаются от штатных пользователей операционной системы Linux. В противовес тому, что информация о пользователе Linux может быть изменена даже в то время, когда пользователь работает в системе (при помощи таких инструментов, как `sudo` или `su`), политика SELinux может обеспечить (и обеспечивает), чтобы пользователь SELinux оставался таким же и тогда, когда пользователь Linux сам изменился. На основе неизменного состояния пользователя SELinux можно реализовать особый контроль доступа, дающий уверенность в том, что пользователи не смогут работать в обход набора установленных им разрешений даже в том случае, когда они имеют привилегированный доступ.

В качестве примера такого контроля доступа является **пользователь-ориентированный контроль доступа** (User-based access control – UBAC), возможности которого некоторые дистрибутивы Linux поддерживают (по желанию) и который оберегает пользователей от доступности файлов различных SELinux-пользователей, даже если те пользователи пытаются применять дискреционный механизм Linux для открытия доступа к файлам друг друга.

Наиболее важной характерной чертой пользователей SELinux, однако, является то, что они формулируют ограничения в части того, какие роли Linux-пользователи принимают, а какие нет. Какой-либо Linux-пользователь сначала назначается какому-то пользователю SELinux – множество Linux-пользователей может быть назначено этому же пользователю SELinux. После назначения этот пользователь не может выбрать какую-либо роль SELinux, т. к. он не является способным в этом участвовать.

Роль-ориентированный контроль доступа, реализованный в SELinux, представлен на рис. 1.3.

Пользователи SELinux по договоренности создаются с дописыванием в конце `_u`, хотя это и не обязательно. Те пользователи SELinux, которые доступны во многих дистрибутивах, являются названными так же, как и пред-

ставляющие их роли, но вместо окончания _r они имеют окончание _u. Например, для роли sysadm_r существует пользователь SELinux sysadm_u.

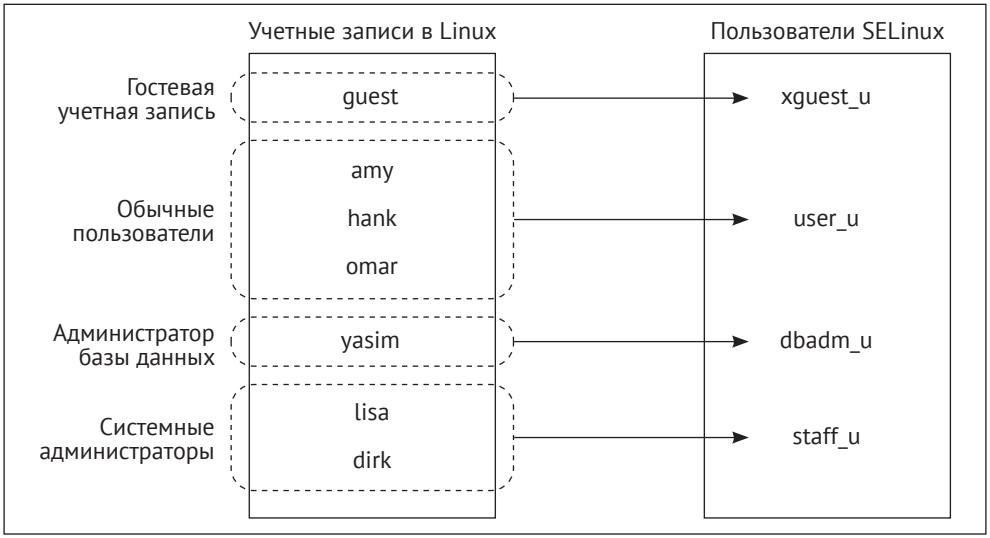

Рис. 1.3 ❖ Соотнесение учетных записей Linux с пользователями SELinux

1.2.5. Контроль информационных потоков посредством мандатного механизма

Четвертая часть параметров безопасности SELinux – мандатная конфигурация – не всегда присутствует (некоторые дистрибутивы Linux по умолчанию не включают мандатные метки). Если же она все-таки есть, тогда эта часть маркировки требуется для обеспечения **многоуровневой защиты MLS** (*multilevel security*), поддерживаемой в SELinux. Мандатные метки позволяют разделить защищаемые ресурсы на классы и ограничить доступ к этим ресурсам на основе соответствующих разрешений. Эти метки состоят из двух частей: значения конфиденциальности (обозначаются префиксом s) и значения категории (обозначаются префиксом c).

Во многих больших организациях и компаниях документы подразделяются на «общедоступные», «конфиденциальные» и «строго конфиденциальные». SELinux может назначить процессам определенный уровень доступа по отношению к этим ресурсам. С помощью механизма многоуровневой защиты система SELinux может быть сконфигурирована в соответствии с моделью Белла–ЛаПадула – моделью защиты, которая может быть охарактеризована как «не читать все, что выше, и не записывать во все, что ниже» (англ. *no read up and no write down*). Данная модель основана на уровне дозволенности процесса: этот процесс не может читать что-либо с более высоким уровнем конфиденци-

альности и не может писать в (или обмениваться информацией как-то иначе с) любой(ым) ресурс(ом) с более низким уровнем конфиденциальности. SELinux не использует метки с названиями вроде «общедоступно», «конфиденциально» и т. п. Вместо этого он применяет числовые значения от 0 (для самого низкого уровня конфиденциальности) до того числа, которое администратор определит как предельное значение (оно конфигурируется и устанавливается после сборки политики SELinux).

Часть метки, связанная с категориями, позволяет ресурсам быть отнесенными к одной категории или нескольким, на которые контроль доступа также распространяется. Одной функциональной возможностью из их общего числа, появляющихся от использования категорий, является поддержка разделения объектов доступа по принадлежности разным владельцам (например, в рамках системы размещения и предоставления доступа к приложениям для нескольких клиентов) в системе Linux, когда процессам и ресурсам, принадлежащим одному владельцу, назначается собственный набор категорий, в то время как процессы и ресурсы другого владельца получают отличающийся набор категорий. Когда процесс не имеет должных категорий, он не может выполнять какие-либо операции с ресурсами (или другими процессами), которые имеют другие присвоенные категории.

Неписаное соглашение в мире SELinux состоит в том, что (по крайней мере) две категории применяются для формирования различий между двумя участниками (владельцами). Имея возможность случайного выбора двух категорий для некоего участника не из предопределенного набора категорий, мы убеждаемся, что каждый участник имеет уникальную комбинацию и таким образом обеспечивается необходимая изоляция. Использование двух категорий не является обязательным, но реализуется такими сервисами, как **sVirt** и **Docker**.

В этом смысле категории можно рассматривать как особый признак, позволяющий предоставить доступ, только когда он совпадает и у субъекта, и у объекта доступа. Поскольку многоуровневая защита является не часто используемой, преимущества исключительного использования категорий представляются в том, что называется **многокатегорийная защита** (Multi-category security – MCS). Это частный случай многоуровневой защиты, где присутствует только один уровень конфиденциальности (`s0`).

1.3. Формирование и распределение политик

При включении SELinux принудительный режим (*enforcement*) контроля доступа автоматически не запускается. Если SELinux является включенным, но он не может найти политику, тогда запуск будет отменен. Это связано с тем, что политики определяют поведение системы (которое SELinux должен позволять). Политики в основном распространяются в откомпилированном виде (подобно программам) и представляют собой некие модули политик. Эти модули затем объединяются в единое хранилище политик и загружаются в па-

мять, для того чтобы позволить SELinux претворять в жизнь правила политик на данной системе.

Операционная система «Gentoo», будучи метадистрибутивной, полностью основанной на исходных текстах, с тем же успехом распространяет политики SELinux как (исходный) код, который является компилируемым и собираемым во время установки, подобно тому, как это делается с другими программами.

Под метадистрибутивом, в контексте создателей системы, понимается качественно новая сущность, появившаяся в связи с основным процессом создания Linux-систем (а именно после него), позволяющая наиболее гибко настраивать конечную систему и оптимизировать производительность.

Следующее изображение показывает взаимосвязи между **правилами политики** (*policy rules*), **модулями политики** (*policy modules*) и **пакетом политики** (*policy package*) (который часто является один к одному отражением **хранилища политики** (*policy store*)).

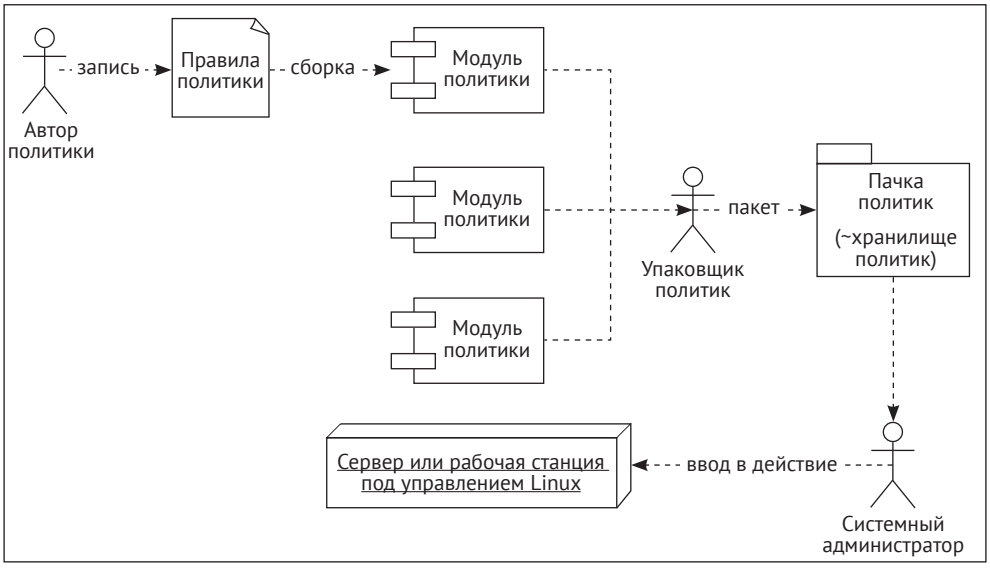

Рис. 1.4 ❖ Взаимосвязь между правилами политики, модулями политик и хранилищем политик

1.3.1. Создание политик SELinux

Специалист, формирующий политики, может записывать правила на трех возможных языках:

1) в стандартном для SELinux формате, воспринимаемом человеком («format-a human-readable») при помощи хорошо устоявшегося языка для написания политик SELinux;

2) в посредническом стиле политики («reference policy style») – он расширяет стандартный формат SELinux макросами «M4» для способствования разработке политик;
3) на обобщенно-промежуточном языке SELinux («common intermediate language» – «CIL») – формат для политик, воспринимаемый компьютером (и с большей долей напряжения человеком тоже).

Наибольшее число дистрибутивов, поддерживающих SELinux, основывается на посреднической политике (https://github.com/TresysTechnology/refpolicy/wiki). В этом случае полностью функциональный набор политик SELinux управляется как проект свободного программного обеспечения. Такой подход позволяет поставлять дистрибутивы с набором функциональных политик, а не писать их самим. Многие участники проекта являются разработчиками дистрибутивов, пытающимися включить изменения их дистрибутива в сам проект посреднической политики, где изменения рецензируются специалистами, чтобы уверенно избежать правил, привносящих в проект то, что может подвергнуть опасности любую платформу. Без широкого набора макросов `M4`, предложенного проектом посреднической политики, писать повторно используемые модули политики быстро становится очень беспокойным занятием.

Формат обобщенно-промежуточного языка является довольно новым, несмотря на это, он уже много где используется (новое пользовательское пространство SELinux преобразует все в данный формат (CIL) в фоновом режиме), но при этом он еще не является общепринятым для создателей политик, чтобы применять его безоговорочно.

В качестве примера рассмотрим правило для веб-сервера, которое мы обсуждали ранее, но повторим здесь еще раз для вашего удобства: оно позволяет процессам, промаркированным как `httpd_t`, связываться с TCP-портами, промаркированными как `httpd_port_t`.

В стандартном формате это записывается следующим образом:

```
allow httpd_t http_port_t : tcp_socket { name_bind };
```

Использование посреднического стиля записи политик приводит к тому, что данное правило становится частью следующего вызова макроса:

```
corenet_tcp_bind_http_port(httpd_t)
```

В обобщенно-промежуточном языке это правило должно быть изображено следующим образом:

```
(allow httpd_t http_port_t (tcp_socket (name_bind)))
```

В большинстве представлений мы можем видеть, что данное правило отражает следующее:

1) субъект (тот, кто является выполняющим действие); в данном случае это набор процессов, промаркированных как имеющие тип `httpd_t`;

2) целевой ресурс или объект (целевой для воздействия); в данном случае он является набором TCP-сокетов (`tcp_socket`), имеющих маркер `httpd_port_t`. В посредническом стиле записи политики данный объект реализован в имени функции;
3) конкретное действие или разрешения; в данном случае оно является действием по связыванию с портом (`name_bind`). В посредническом стиле записи политики данный объект реализован в имени функции;
4) результат, который данная политика будет претворять в жизнь; в данном случае это то, что действие разрешено (`allow`). В посредническом стиле записи политики данный объект реализован в имени функции.

Любая политика в основном пишется для какого-либо приложения, или набора приложений. Так, предыдущий пример будет частью политики, написанной для веб-сервера.

Создатели политик обычно формируют три файла на приложение или на набор приложений:

1) файл с расширением `.te`, который содержит тип претворяемых в жизнь правил;
2) файл с расширением `.if`, который содержит интерфейс и шаблон определений, позволяющих авторам политик проще использовать вновь сгенерированные правила политик для улучшения других политик совместным образом. Вы можете сравнить их с заголовочными файлами в других языках программирования;
3) файл с расширением `.fc`, который содержит выражения, определяющие параметры безопасности файла. Эти выражения являются правилами, которые назначают маркировку для ресурсов, расположенных в файловой системе.

Завершенная политика упаковывается в модуль политики SELinux.

1.3.2. Распределение политик в виде модулей

Исходно SELinux использовал единственный монолитный метод работы с политиками: все правила, контролирующие возможный доступ, хранились в одном файле политик. Вскоре стало ясно, что он не поддается управлению при долгосрочном использовании, и в результате родилась идея разработки модульного подхода к политикам.

С модульным подходом разработчики политик могут писать наборы обособленных политик для отдельных приложений (или групп приложений), ролей и т. п. Эти политики затем создаются и распространяются как модули политик. Платформы, которые нуждаются в контроле доступа для отдельных приложений, загружают модуль политики SELinux, который определяет правила доступа для этого приложения.

Процесс построения модулей политик показан на рис. 1.5. На нем также показывается, где обобщенно-промежуточный язык (CIL) вступает в игру, даже когда правила политики сами не написаны на нем. Для дистрибутивов, кото-

рые еще не поддерживают обобщенно-промежуточный язык, модуль политики (`semodule`) будет напрямую перемещен из состояния `.pp` файла к файлу хранилища политик «`policy.##`».

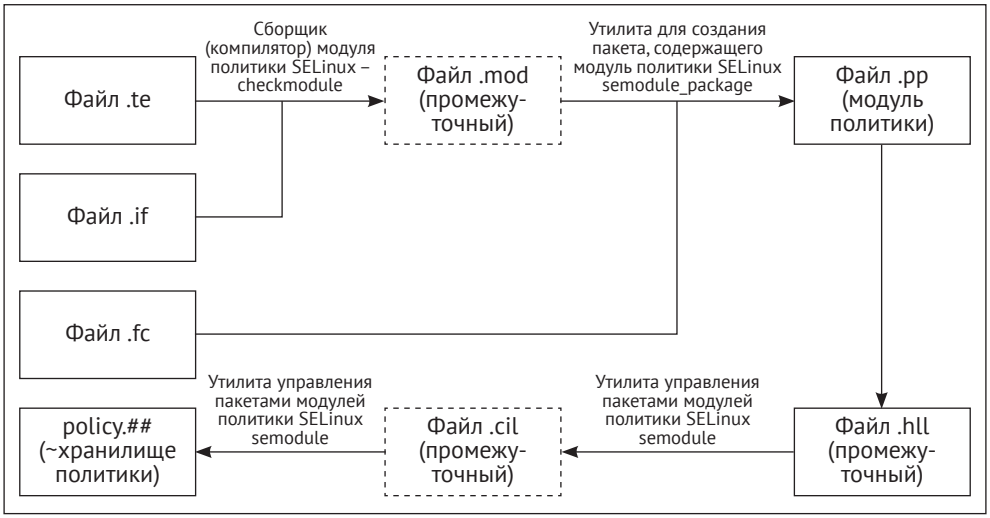

Рис. 1.5 ❖ Процесс преобразования:
от правила политики в хранилище готовых политик

Вместе с новыми пользовательскими программами системы SELinux файлы *.pp (которые являются модулями политики SELinux) продуманы для того, чтобы быть записанными на языке высокого уровня (high-level language – HLL). Не следует ожидать, что это означает большую читаемость их человеком: данные файлы являются двоичными. В подобном случае подразумевается, что разработчики SELinux хотят поддерживать создаваемые политики SELinux в разных форматах, среди которых этот относят к высокоуровневым языкам, раз уж включают в состав средство грамматического разбора (*parser*), которое может конвертировать эти файлы в обобщенно-промежуточный формат (CIL). Маркировка форматов двоичного модуля как высокоуровневых позволяет в проект SELinux вводить различие между языками высокого уровня (HLL) и обобщенно-промежуточным форматом (CIL) в манере обратной совместимости.

При распространении политик многие дистрибутивы Linux помещают модули *.pp внутри каталога `/usr/share/selinux`, обычно в подкаталог, называемый позже хранилищем политик (такой как `targeted`). Здесь эти модули являются готовыми для активации администраторами.

Когда активируется модуль, команда `semodule` (часть пакета `policycoreutils`) будет копировать его в предназначенную для этого директорию:

- `/etc/selinux/targeted/modules/active/modules` (для Red Hat Enterprise Linux)

или

- `/var/lib/selinux/mcs/active/modules` (для Gentoo).

Данное размещение определяется версией пользовательского пространства SELinux – более ранние версии используют каталог /var/lib. Когда все модули, наконец, сгруппированы в одном месте, собирается окончательная бинарная политика, размещается в файле /etc/selinux/targeted/policy/policy.30 (последняя цифра может быть иной) и загружается в память.

В операционной системе Red Hat Enterprise Linux политики предоставляются пакетом selinux-policy-targeted (или -minimum, или -mls). В Gentoo – различными пакетами sec-policy/selinux-* (Gentoo использует раздельные пакеты для каждого модуля, сокращая число политик, загруженных на некоторую усредненную систему).

1.3.3. Комплектация модулей в хранилище политик

Хранилище политик (policy store) содержит единственную, но всеобъемлющую политику, и только одна политика может быть активна в системе в любой момент времени. Администраторы могут выбирать хранилища политик, хотя это часто требует от системы перезагрузки и может даже обязывать выполнять перемаркировку записей в системе (перемаркировка – это действие по перенастройке параметров безопасности на всех файлах и ресурсах, используемых в данной системе).

Активная политика, применяемая в системе, может быть определена при помощи команды sestatus (статус SELinux, предоставляемый средствами пакета policycoreutils) следующим образом:

```
# sestatus | grep "Loaded policy name"
Loaded policy name:             targeted
```

В этом примере текущая загруженная политика (хранилище) называется targeted. Имя политики, которое SELinux будет использовать после следующей перезагрузки, определяется в конфигурационном файле /etc/selinux/config параметром SELINUXTYPE.

Это система инициализации на уровне ОС (будь это **SysV-совместимая** или systemd), которая является в основном ответственной за загрузку политики SELinux, эффективно активизирующей поддержку функций SELinux в системе. Система инициализации считывает конфигурацию, размещенную в хранилище политики, и загружает файл политики в память. Если система инициализации не поддерживает такие возможности (другими словами, не SELinux-ориентированная), тогда политика может быть загружена при помощи команды load_policy, являющейся частью пакета policycoreutils.

1.4. Различия между политиками

Наиболее общие хранилища политик носят названия strict, target, mcs и mls. Ни одно из этих имен, связанных с хранилищами политик, не является жестко закрепленным, несмотря на то что это и считается предметом соглашения. Поэтому их рекомендуется уточнять в документации к дистрибутиву, для того

чтобы проверить, какое предпочитаемое имя политики должно быть. Тем не менее имя часто предоставляет некоторую информацию о параметрах SELinux, которые включены в политике.

1.4.1. Поддержка многоуровневой защиты (MLS)

Одна из опций, которая может быть включена, – это поддержка многоуровневой защиты (MLS). Если она отключена, тогда параметры безопасности SELinux не будут содержать четвертый компонент с мандатной конфигурацией, определяя параметры безопасности для процессов и файлов, как показано ниже:

```
staff_u:sysadm_r:sysadm_t
```

Для того чтобы проверить, выключена ли поддержка многоуровневой защиты, достаточно посмотреть только на отсутствие четвертого компонента среди параметров безопасности, но это также можно узнать из строки «`Policy MLS status`»[1], выводимой командой `sestatus`:

```
# sestatus | grep MLS
Policy MLS Status:        disabled
```

Третий способ – это посмотреть в псевдофайле `/sys/fs/selinux/mls`. Значение `0` говорит, что многоуровневая защита выключена, значение `1` – что включена:

```
# cat /sys/fs/selinux/mls
0
```

Многоуровневая защита включена в основном в такие хранилища политик, как `targeted`, `mcs` и `mls`, тогда как в `strict` обычно выключена.

1.4.2. Манера поведения с неизвестными разрешениями

Разрешения (такие как «чтение», «открытие» и «блокирование») являются определенными одновременно в ядре Linux и в самой политике. Однако иногда более новые ядра поддерживают разрешения, которые текущая политика пока что еще не понимает.

Возьмем разрешение `block_suspend` (быть способным блокировать системные зависания) в качестве примера. Если ядро Linux поддерживает (и проверяет) данный доступ, а загруженная политика SELinux не понимает пока эти разрешения, тогда SELinux должен решить, как нужно иметь дело с ними.

SELinux может быть сконфигурирован так, чтобы выполнять при этом одно из следующих действий:

1) `allow`: принимать все, что является неизвестным, как позволенное;
2) `deny`: принять, что нет никаких разрешений для выполнения данных действий;
3) `reject`: остановить систему и дать команду на прекращение выполнений всех функций процессора.

---

[1] В переводе с англ. «Статус политики многоуровневой защиты».

Данная конфигурация настраивается через значение deny_unknown. Для того чтобы определить заданное состояние для неизвестных разрешений, надо обратиться к строчке sestatus для Policy deny_unknown status:

```
# sestatus | grep deny_unknown
Policy deny_unknown status:   denied
```

В данном примере автор выполняет команду просмотра состояния SELinux sestatus и указывает через вертикальную черту, что необходимо выполнить поиск и выборку всех строк, содержащих фразу deny_unknow, при помощи утилиты grep.
Результатом этой команды является вторая строчка, показывающая, что состояние политики запрет_неизвестного (Policy deny_unknown status) находится в статусе отказа (denied) для выполнения любых действий, разрешения которых неизвестны в SELinux.

Администраторы могут устанавливать данный параметр для себя в файле /etc/selinux/semanage.conf через переменную handle-unknown, указав ей одно из трех значений: allow, deny или reject.

Операционная система «Red Hat Enterprise Linux» по умолчанию принимает все неизвестные разрешения как дозволенные, тогда как операционная система «Gentoo» по умолчанию запрещает их.

Название политики block_suspend можно в русскоязычном варианте представить как воспрепятствование_зависаниям.
Названия конфигураций в переводе на русский выглядят в данном контексте следующим образом:

- allow – разрешать,
- deny – отказывать;
- reject – подавлять.

Конфигурационный параметр deny_unknown – запрет_неизвестного.
Здесь и далее составное название sestatus имеет смысл рассматривать как сокращение от SELinux status – состояние SELinux.
Policy deny_unknown status – состояние политики запрет_неизвестного.
Переменная handle-unknown, определяемая в файле контроля конфигурации и действий утилит semanage (управление политикой SELinux) и semodule (управление пакетами модулей политики SELinux), может переводиться как определитель неизвестного.

1.4.3. Поддержка неограниченных доменов

Политика SELinux может быть очень строгой, ограничивающей приложения как можно ближе к их необходимой функциональности, но также может быть и весьма свободной, в которой приложениям разрешено работать. Одна из концепций, доступных во многих политиках, – это идея неограниченных доменов (unconfined domains). Когда она включена, то означает, что некоторые SELinux-домены (параметры безопасности, свойственные процессу) позволяют делать едва ли не все, что захочется (конечно, в пределах разрешений прав дискреционного разграничения доступа, который по-прежнему сохраняется), и только избранное число доменов являются в полной мере ограниченными в своих действиях. Неограниченные домены были привнесены для того, чтобы

позволить защите SELinux быть активной на рабочих станциях и серверах, где администраторы не хотят полностью ограничивать всю систему, а только некоторые приложения, запущенные на ней. В основном такие реализации имеют прикладное значение для ограничения сервисов, имеющих сетевой интерфейс (такие как веб-сервер и система управления базами данных), позволяя конечным пользователям и администраторам неограниченно перемещаться повсюду.

В случаях, когда речь идет о других защитах мандатного разграничения доступа, таких как AppArmor, *неограниченность* (`unconfinement`) является, по существу, частью дизайна системы, поскольку системы защиты контролируют действия только для определенных приложений или пользователей. Однако SELinux был спроектирован как система полного контроля мандатного разграничения доступа, и таким образом необходимо предоставить правила разграничения доступа даже для тех приложений, для которых не нужно никаких правил. Помечая эти приложения как неограничиваемые, мы почти не имеем дополнительных ограничений, налагаемых средствами SELinux.

Мы можем видеть, являются ли включенными неограниченные домены или нет, в системе с помощью команды `seinfo`, которую используем для запроса поддержкой текущей политикой типа защиты с названием `unconfined_t`. В системах, где неограниченные домены поддерживаются, этот тип будет доступен:

```
# seinfo -tunconfined_t
  unconfined_t
```

Для систем, где не поддерживаются, этот тип не будет частью политики:

```
# seinfo -tunconfined_t
ERROR: could not find datum for type unconfined_t
```

Многие дистрибутивы, которые включают неограниченные домены, называют свои политики `targeted`, но это только соглашение, которое не всегда выполняется. Поэтому всегда лучше это уточнить, используя команду `seinfo`. RHEL включает неограниченные домены, тогда как в Gentoo это конфигурируемая настройка через флаг `unconfined USE`.

1.4.4. Ограничение межпользовательского обмена

Когда включен пользователь-ориентированный контроль доступа (UBAC), некоторые типы SELinux будут защищены дополнительными ограничениями. Это будет гарантировать, что один пользователь SELinux не получит доступ к файлу (или другому специфическому ресурсу) другого пользователя, даже когда эти пользователи предоставят доступ к своим данным через стандартные средства разрешения Linux. Пользователь-ориентированный контроль доступа предоставляет некоторый дополнительный контроль доступа над информационными потоками между ресурсами, но он весьма далек от совершенства. В сущности, он позволяет изолировать пользователей SELinux одного от другого.

Ограничение в SELinux представлено в виде некоего правила ограничения доступа, которое использует все части параметра безопасности, для того чтобы принять решение. В отличие от правил принудительного исполнения, которые являются основанными исключительно на типах, ограничения могут учитывать пользователя SELinux, роль SELinux или мандатную конфигурацию. Ограничения обычно разрабатываются один раз и остаются неизменными, иначе многие авторы политик не будут касаться ограничений в своих разработках.

Многие дистрибутивы Linux, включая Red Hat Enterprise Linux, исключают пользователь-ориентированный контроль доступа. Gentoo позволяет пользователям выбрать, хотят ли они его использовать или нет, с помощью флага `ubac USE` (который является включенным по умолчанию).

1.4.5. Последовательные изменения версий политик

Изучая выданные сведения командой sestatus, мы видим, что имеется следующая запись о версии политики:

```
# sestatus | grep version
Max kernel policy version:      28
```

Приведенная здесь версия не имеет ничего общего с обозначением версии правил политики, но при этом определяет функции SELinux, которые поддерживаются в настоящее время запущенным ядром. В приведенном выводе число 28 определяет наиболее позднюю версию политики, которая поддерживается ядром. Постоянно новые возможности добавляются в SELinux, и номер версии увеличивается. Сам файл политики (содержащий все правила SELinux, загруженные в момент запуска системы) может быть найден в файле `/etc/selinux/targeted/policy` (где `targeted` относится к используемому хранилищу политики, поэтому если система применяет хранилище с именем `strict`, то путь будет `/etc/selinux/strict/policy`).

Если существует множество файлов с политиками, тогда мы можем использовать команду seinfo, для того чтобы найти в выведенной ею информации сведения о том, какой файл политики используется:

```
# seinfo
Statistics for policy file: /etc/selinux/targeted/policy/policy.30
Policy Version & Type: v.30 (binary, mls)
...
```

В табл. 1.2 предоставлен текущий список последовательных изменений политик в плане реализованных в них возможностей, соотнесенных с версией ядра Linux, в которых эти политики были реализованы. Многие функции представляют интерес только для разработчиков политик, но приведенная здесь эволюция функциональных возможностей дает нам хорошее представление о развитии самого SELinux.

Таблица 1.2. История развития возможностей SELinux

Версия	Ядро Linux	Описание
12		Старый API для SELinux, сейчас устарела
15	2.6.0	Введены новые API для SELinux
16	2.6.5	Добавлена поддержка условных расширений политики
17	2.6.6	Добавлена поддержка протокола IPv6
18	2.6.8	Добавлена поддержка детализированных разрешений для сокетов взаимодействия приложений пользователя с процессами ядра (netlink-сокеты)
19	2.6.12	Добавлена поддержка многоуровневой защиты (MLS)
20	2.6.14	Сокращен размер таблицы векторов доступа
21	2.6.19	Добавлена поддержка для диапазона переходов многоуровневой защиты
22	2.6.25	Введены возможности политики (policy capabilities)
23	2.6.26	Добавлена возможность для каждого домена рекомендательного режима (permissive mode)
24	2.6.28	Добавлена поддержка четкой иерархии (границы типов)
25	2.6.39	Добавлена поддержка переходов, основанных на именах файлов
26	3.0	Добавлена поддержка для ролевых переходов у классов, не относящихся к процессам. Добавлена поддержка атрибутов ролей
27	3.5	Добавлена поддержка гибкого наследования пользователей и ролей для вновь созданных объектов
28	3.5	Добавлена поддержка гибкого наследования типов для вновь созданных объектов
29	3.14	Добавлена поддержка атрибутов в ограничениях (constraints) SELinux
30	4.3	Добавлена поддержка для расширенных разрешений и реализация их в первую очередь на устройствах контроля ввода/вывода (IOCTL). Улучшение поддержки XEN

По умолчанию, когда собирается политика SELinux, используется новейшая версия, поддерживаемая ядром и библиотекой `linsepol` – ответственной за сборку бинарного вида политики. Администратор может сам изменить версию на более низкую, используя параметр `policy-version` в файле `/etc/selinux/semanage.conf`.

1.4.6. Качественное изменение версий политик

По сравнению с возможностями политик, описанных выше, основным различием между политиками (и дистрибутивами) является собственно их содержание. Мы уже описывали, что многие дистрибутивы основывают свою политику на проекте посреднической политики (reference policy). Но хотя этот

проект рассматривается как главный (master) во многих дистрибутивах, при этом каждый дистрибутив имеет свои собственные отклонения от основного набора политики.

Многие дистрибутивы создают широкие дополнения к политике без прямого интегрирования их в исходную посредническую политику. Существует несколько возможных причин, почему это так не делается:

1) конкретная политика, модернизированная или расширенная, является еще достаточно незрелой: Red Hat изначально работает с активными политиками, но в рекомендательном режиме (permissive), преднамеренно политики не являются принудительными (enforced). Взамен SELinux регистрирует то, что должно быть предотвращено, и, основываясь на этих журналах, политики затем улучшаются. Это означает, что политики являются готовыми для фактического применения только после нескольких выпусков;
2) конкретная политика, модернизированная или расширенная, является также специфичной для конкретного дистрибутива: если набор политик не носит характера многократного использования в других дистрибутивах, то некоторые поставщики предпочтут оставить эти политики при себе, т. к. действия по размещению изменений в отслеживаемом (upstream) проекте требуют довольно много усилий;
3) конкретная политика, модернизированная или расширенная, не следовала следующим правилам и рекомендациям для внесения изменений в репозиторий проекта: посредническая политика имеет набор рекомендаций, которым должны следовать политики. Если набор правил политики не удовлетворяет этим требованиям, тогда он не будет принят;
4) конкретная политика, модернизированная или расширенная, не реализовала прежнюю модель защиты, как того требует проект посреднической политики: раз SELinux является весьма обширной системой принудительного контроля доступа, то вполне возможно написать полностью отличные от нее политики;
5) распространитель не имеет времени или ресурсов для внесения изменений в отслеживаемую ветку проекта в репозитории.

Все это означает, что политики SELinux между дистрибутивами (и даже конкретными версиями одного и того же дистрибутива) могут, являясь корректными по содержанию, быть совершенно различными. Например, Gentoo стремится внимательно отслеживать проекты посреднических политик и вносить изменения, соединяя содержание, в течение нескольких недель.

1.5. Заключение

В этой главе мы увидели, что SELinux представляет собой некий дополнительный высокоточный механизм контроля доступа верхнего уровня в иерархии систем контроля доступа операционной системы Linux. SELinux реализуется

через модульную систему безопасности Linux (LSM) и использует метки для идентификации защищаемых ресурсов и процессов, основанных на принадлежности (пользователе), роли, типе защиты и даже на мандатном уровне и категории ресурса. Было рассмотрено, как SELinux-политики обрабатываются в системах, поддерживающих SELinux, и мы кратко коснулись того, как авторы политик их структурируют.

Дистрибутивы Linux реализуют политики SELinux, которые могут иметь минимальные различия от других, основанных на поддерживаемых функциях, таких как мандатные метки, поведение при появлении неизвестных разрешений, поддержка ограничивающих уровней или специфические ограничения, приложенные конкретно, такие как пользователь-ориентированный контроль доступа (UBAC). Однако большинство правил политики сами по себе являются подобными и даже основаны на одном и том же проекте посреднической политики.

Переключение между режимами работы SELinux и рассмотрение журналов событий, которые SELinux создает, когда запрещает определенный доступ, являются предметом рассмотрения следующей главы. В ней мы также разберем, как подступиться к часто встречающемуся требованию отключения SELinux и почему это решение неудачное для реализации.