

Глава 6. Модули безопасности Linux.

6.1 Linux Security Module (LSM).

Linux Security Module (LSM)



- LSM – фреймворк, позволяющий добавить в ядро дополнительные проверки доступа
- LSM не привязан к какому-либо способу проверки
- Используется для реализации отличных от DAC моделей доступа:
 - MAC – Mandatory Access Control
 - RBAC – Role Based Access Control

В большинстве операционных систем имеются средства управления доступом, которые определяют, может ли определенный объект (пользователь или программа) получить доступ к определенному ресурсу.

В системах UNIX® применяется разграничительный контроль доступа (discretionary access control, DAC). Этот метод позволяет ограничить доступ к объектам на основе групп, к которым они принадлежат.

Такое разграничение прав доступа может привести к возникновению ряда проблем из-за того, что программа, в которой может быть обнаружена уязвимость, наследует все права доступа пользователя. Следовательно, она может выполнять действия с тем же уровнем привилегий, какой есть у пользователя (что нежелательно).

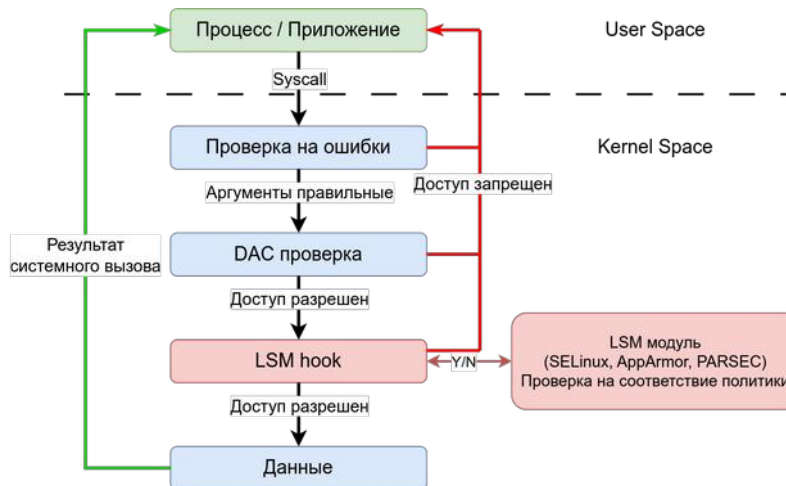
Вместо того чтобы определять ограничения подобным образом, более безопасно использовать принцип наименьшего уровня привилегий (principle of least privilege), согласно которому программы могут делать только то, что им необходимо для выполнения своих задач, и не более того.

Разработчики ядра Linux не стали реализовывать в своем проекте конкретную модель доступа основанную на принципе наименьших привилегий, вместо этого создали фреймворк LSM (Linux Security Module), который позволяет применять любые существующие и будущие модели управления доступом.

Глава 6. Модули безопасности Linux.

LSM позволяет добавить в систему поддержку таких модулей управления доступом как MAC (Mandatory Access Control) или RBAC (Role Based Access Control).

Принцип работы LSM



В 2001 году было предложено добавить в ядро Линукс новый механизм обеспечения безопасности — SELinux, но идея была отвергнута, т. к. существовали и альтернативные проекты в области обеспечения безопасности и не было консенсуса, какая модель лучше.

Поэтому в конце 2003 была реализована концепция LSM, которая позволяла добавлять дополнительные модули ядра, которые реализуют мандатный контроль доступа.

Все LSM выполняют дополнительные проверки внутри соответствующего модуля, а запрос на доступ от приложения в виде системного вызова передается модулю через LSM hook. При чем вызывается LSM hook после того, как было получено разрешение от дискреционного механизма контроля доступа. Таки образом LSM не подменяет, но дополняет традиционные механизмы управления доступом.

Как при этом оценивается запрос от приложения внутри модуля не важно. Модуль может вернуть только два ответа: Да — разрешить действие или Нет — запретить действие.

Как правило внутри модулей имеется некоторая политика, в соответствии с которой, модуль принимает решение о разрешении доступа.

Существуют несколько реализаций LSM модулей:

- SELinux (Security Enhanced Linux)
- AppArmor (Application Armor)
- PARSEC
- Smack
- TOMOYO
- Landlock

6.2 SELinux.

6.2.1 Механизм работы политик SELinux.

Что такое SELinux



- SELinux реализует систему mandatory access control — MAC
- SELinux устанавливает правила для файлов и процессов на основе политик
- SELinux — модуль LSM

Linux с улучшенной безопасностью (SELinux) - это реализация принудительного управления доступом (mandatory access control — MAC) в ядре Linux, проверяющего разрешение на выполнение операций после проверки стандартного разграничительного управления доступом DAC.

SELinux создан Агентством Национальной Безопасности и вводит в действие правила для файлов и процессов в системе Linux, для совершаемых над ними действий, основываясь на установленной политике.

Что такое SELinux

- Каждый файл, каталог или устройство это объект
- Каждый процесс — субъект
- У объектов и субъектов имеются метки
- Политика устанавливает правила взаимодействия объектов и субъектов на основе меток

При использовании SELinux, файлы, включая директории и устройства являются объектами. Процессы, такие как, выполнение команды пользователем или приложение Mozilla® Firefox®, являются субъектами.

Основное назначение архитектуры MAC - это возможность принудительного назначения административно-установленной политики безопасности над всеми процессами и файлами системы, при этом решение основывается на метках, содержащих множество значимой информации по безопасности. Когда механизм SELinux реализован, он переводит систему в состоянии достаточной защищенности и предоставляет критичную поддержку приложениям, защищая приложения от взлома или обхода безопасности.

MAC предоставляет строгое разделение приложений и позволяет безопасное исполнение не доверенных приложений. Обладая способностью ограничивать привилегии, связанные с исполнением процессов, MAC ограничивает рамки потенциальной угрозы, таким образом ограничивая взлом уязвимостей в приложениях и системных службах. MAC включает защиту информации от пользователей корректно авторизованных в системе с ограниченными правами также как и от авторизованных пользователей, которые неосознанно исполняют вредоносный код.

Что такое SELinux

- Пользователи Linux сопоставляются (маппируются) с пользователями SELinux
- Пользователи SELinux - это часть политики SELinux

В операционных системах Linux с запущенным SELinux, существуют пользователи Linux и пользователи SELinux. Пользователи SELinux - это часть политики SELinux. Пользователи Linux сопоставляются (маппируются) с пользователями SELinux. Для того, чтобы избежать путаницы, в данном руководстве используются два термина "пользователь Linux" и "пользователь SELinux" для различия двух разных понятий.

Модели управления доступом SELinux

- **Type Enforcement (TE):** Основной механизм ограничения, используемый в целевых политиках
- **Role-Based Access Control (RBAC):** в этой модели права доступа реализуются в качестве ролей
- **Multi-Level Security (MLS):** многоуровневая модель безопасности, в которой всем объектам системы присваивается определенный уровень доступа
- **Multi-Category Security(MCS):** Расширение MLS, используется в целевой политике для ограничения виртуальных машин и контейнеров через sVirt

В дополнение к DAC SELinux (Security Enhanced Linux) предлагает несколько вспомогательных моделей управления доступом:

- **Type Enforcement (TE):** Основной механизм ограничения, используемый в целевых политиках. Позволяет детально, на самом низком уровне управлять разрешениями. Самый гибкий, но и самый трудоемкий для системного администратора механизм.
- **Role-Based Access Control (RBAC):** в этой модели права доступа реализуются в качестве ролей. Ролью называется разрешения на выполнение определенных действий одним или несколькими элементами системы над другими частями системы. По-сути, RBAC является дальнейшим развитием TE.
- **Multi-Level Security (MLS):** многоуровневая модель безопасности, в которой всем объектам системы присваивается определенный уровень доступа. Разрешение или запрет доступа определяется только соотношением этих уровней.
- **Multi-Category Security(MCS):** Расширение MLS, используется в целевой политике для ограничения виртуальных машин и контейнеров через sVirt.

Что НЕ МОЖЕТ SELinux

- Не антивирус
- Не замена паролям, брандмауэрам, или разрешениям на доступ и т.д.
- Не система типа все-в-одном

SELinux не является:

- антивирусным программным обеспечением.
- заменой паролям, межсетевым экранам или другим системам безопасности.
- решением безопасности "всё в одном".

SELinux разработан, для усовершенствования существующих решений по безопасности. Даже с запущенным SELinux, необходимо использование практик по безопасности, таких как обновление программного обеспечения последними обновлениями, использование сложных паролей, межсетевых экранов и прочего.

Терминология SELinux

- **Сущность (identity)** - этот термин схож с понятием "пользователь" в классической схеме доступа. Сущность может иметь такое же название, как и логин пользователя, но в отличие от логина, сущность не меняется после выполнения команды su.
- **Домен (domain)** - это список того, что может делать отдельный процесс. Фактически домен - это действия, минимально необходимые одному процессу для выполнения его задачи.
- **Роль (role)** - это список доменов, которые могут быть использованы. Если некоего домена нет в списке, то роль не может выполнить действия из этого домена.
- **Тип (type)** - это набор действий (операция) применительно к объекту. Важно понять отличие от домена. Домен относится к процессам, а тип - к объектам.

Основные понятия SELinux

- **Сущность (identity)** — этот термин схож с понятием "пользователь" в классической схеме доступа. Сущность может иметь такое же название, как и логин пользователя, но в отличие от логина, сущность не меняется после выполнения команды su. Если провести аналогию, то сущность - это конкретный человек, Вася Пупкин, Петя Смирнов и т.д.
- **Домен (domain)** — это список того, что может делать отдельный процесс. Фактически домен - это действия, минимально необходимые одному процессу для выполнения его задачи. По аналогии из реальной жизни, доменом можно назвать набор действий для совершения какой-либо операции.
- **Роль (role)** — это список доменов, которые могут быть использованы. Если некоего домена нет в списке, то роль не может выполнить действия из этого домена. В данном случае можно провести аналогию с должностью. То есть роль - это фактически должность (или должностная инструкция), которая может выполнять определённые наборы операций, или, в понятии SELinux, домены.
- **Тип (type)** — это набор действий (операция) применительно к объекту. Важно понять отличие от домена. Домен относится к процессам, а тип - к объектам, таким как файлы, каталоги, пайпы(pipes), сокеты и т. д.



Терминология SELinux

- **Уровень (level)** - состоит из чувствительности и категории. Используется в системах MLS/MCS.
- **Контекст безопасности (context)** - это набор всех атрибутов, связанных с объектами и субъектами. Контекст безопасности для субъектов (процессов) состоит из сущности, роли, домена, чувствительности и категории

`user:role:type:sensitivity:category`

- **Переход (transition)** - это смена контекста безопасности. Есть два основных типа переходов:
 - Переход домена процесса - процесс меняет контекст;
 - Переход типа файла - создание файлов в определённых подкаталогах.
- **Политика (policy)** - это набор правил, контролирующих взаимодействие ролей, доменов, типов и т.д.

- **Уровень (level)** — это атрибут многоуровневого управления доступом MLS и MCS. Пространство MLS - это пара уровней, записанных в виде `lowlevel-highlevel`, если уровни в данной паре отличаются или, если не отличаются, то `lowlevel`. То есть (`s0-s0` то же самое, что и `s0`). Если дополнительно определены категории, то уровень записывается как `sensitivity:category-set`. Если категории не определены, то запись выглядит как `sensitivity`.
- **Контекст безопасности (context)** — это набор всех атрибутов, связанных с объектами и субъектами. Контекст безопасности для субъектов (процессов) состоит из сущности, роли, домена, чувствительности и категории. Обычно используется только сущность-роль-домен(или тип), а, например, целевая политика от Fedora использует только домены и типы.
- **Переход (transition)** — это смена контекста безопасности. Есть два основных типа переходов:
 - Переход домена процесса — процесс меняет контекст; Например, запускается из-под пользователя некий демон. Selinux, на основе метки исполняемого файла, меняет его контекст.
 - Переход типа файла — создание файлов в определённых подкаталогах. Например, пользователь создаёт html-страничку в каталоге WEB-сервера. Чтобы WEB-сервер получил доступ к этой страничке, необходимо сменить контекст безопасности файла (WEB-сервер не имеет доступа к контексту пользователя).
- **Политика (policy)** — это набор правил, контролирующих взаимодействие ролей, доменов, типов и т. д. Политики работают на уровне системных вызовов и

Глава 6. Модули безопасности Linux.

обрабатываются ядром, но можно реализовать и на уровне приложения. Политики описываются при помощи специального языка описания правил доступа.

Политики SELinux

- Целевая (targeted)
- Минимальная (minimum)
- Многоуровневая (MLS)
- Строгая (strict)

В настоящий момент уже разработано несколько готовых политик безопасности, которые можно использовать по умолчанию на серверах и на домашних компьютерах. Всё, что требуется от системного администратора - выбрать используемую политику и перезагрузить компьютер с включённым SELinux.

В среднем, политика безопасности SELinux для всей системы содержит более ста тысяч правил, так что её создание и отладка занимает значительное время.

Наиболее распространены следующие политики:

1. **Целевая (targeted)**. Эта политика разработана компанией Red Hat и является наиболее используемой;
2. **Минимальная (minimum)**. Является модификацией целевой политики, в которой только выбранные процессы защищаются.
3. **Многоуровневая (MLS)**. Позволяет обеспечивать уровни безопасности и может использоваться госструктурами для хранения информации различных уровней секретности;
4. **Строгая (strict)**. Этот вариант политики подразумевает правило "Что не разрешено, то запрещено".

Установка SELinux

- RedHat – устанавливается по умолчанию
- Debian (см. <https://wiki.debian.org/SELinux/Setup>):
 - Установить пакеты: `selinux-basics`, `selinux-policy-default`, `selinux-utils`, `auditd`
 - Выполнить команду `selinux-activate`
 - Перезагрузить ОС
 - Проверить установку командой `check-selinux-installation`

В RedHat подобных дистрибутивах SELinux устанавливается и включается по умолчанию. Никаких специальных действий по установке не требуется.

В Debian подобных дистрибутивах для использования SELinux его сначала требуется установить.

Пример: Установка SELinux на Debian, см. подробности тут:

<https://wiki.debian.org/SELinux/Setup>

1. Проверьте предварительные требования: файловые системы должны поддерживать расширенные атрибуты, ядро скомпилировано с поддержкой аудита и собственно SELinux.

2. Установите нужные пакеты:

```
root@sl10:~# apt install selinux-basics selinux-policy-default \
selinux-utils auditd
```

3. Выполнить команду `selinux-activate`, которая настроит GRUB, PAM и создаст файл `/.autorelabel`.
4. Перезагрузите ОС.
5. Проверьте установку командой `check-selinux-installation`.

Режимы работы SELinux

- Три режима работы:
 - 1) Enforcing
 - 2) Permissive
 - 3) Disabled
- Команда `sestatus` — показывает состояние SELinux
- Команда `getenforce` — режим работы
- Команда `setenforce` — переключение режима работы

SELinux имеет три основных режим работы, при этом иногда по умолчанию установлен режим Enforcing. Это довольно жесткий режим, и в случае необходимости он может быть изменен на более удобный для конечного пользователя.

1. **Enforcing:** Режим по-умолчанию. При выборе этого режима все действия, которые каким-то образом нарушают текущую политику безопасности, будут блокироваться, а попытка нарушения будет зафиксирована в журнале.
2. **Permissive:** В случае использования этого режима, информация о всех действиях, которые нарушают текущую политику безопасности, будут зафиксированы в журнале, но сами действия не будут заблокированы.
3. **Disabled:** Полное отключение системы принудительного контроля доступа.

Команда `sestatus` показывает состояние SELinux.

Пример:

```
root@sl0:~# sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:       /etc/selinux
Loaded policy name:            default
Current mode:                permissive
Mode from config file:         permissive
Policy MLS status:             enabled
Policy deny_unknown status:    allowed
Memory protection checking:    actual (secure)
Max kernel policy version:     33
```

Также вы можете узнать статус SELinux при помощи команды `getenforce`.

Пример:

```
root@sl0:~# getenforce
```

Глава 6. Модули безопасности Linux.

Permissive

Команда `setenforce` позволяет быстро переключаться между режимами *Enforcing* и *Permissive*, изменения вступают в силу без перезагрузки. Но если вы включаете или отключаете SELinux, требуется перезагрузка, ведь нужно заново устанавливать метки безопасности в файловой системе.

Для установки режима по-умолчанию, который будет применяться при каждой загрузке системы, задайте значение строки `SELINUX=` в файле `/etc/selinux/config`, задав один из режимов — `enforcing`, `permissive` или `disabled`.

Пример:

```
root@sl0:~# grep SELINUX=[a-z] /etc/selinux/config
SELINUX=permissive
```

```
root@sl0:~# apt-cache show selinux-policy-default | grep -A5 Description-ru
Description-ru: ограниченный и специализированный варианты правил SELinux
В пакете содержится образцовый набор правил для SE Linux. В стандартной
конфигурации он предоставляет набор правил ранее известный как
"специализированный" (targeted). Если удалить модуль unconfined, то будет
предоставляться набор правил ранее известный как "ограниченный" (strict).
```

Параметр `SELINUXTYPE=` в файле `/etc/selinux/config` указывает тип применяемой политики.

Ограниченные и неограниченные процессы

- В целевой политике все процессы делятся на две категории
 - Ограниченные т. е. защищаются (ограничиваются) SELinux
 - Неограниченные те что используют DAC механизм в своей работе

Когда целевая политика `targeted` используется, процессы, которые являются целевыми, запускаются в ограниченном домене, остальные процессы запускаются в неограниченном домене. Например, по умолчанию пользователи, прошедшие авторизацию, работают в домене `unconfined_t` и системные процессы запущенные `init`-ом запускаются в домене `initrc_t` - оба домена неограниченные.

Неограниченные домены (так же, как и ограниченные) - это субъекты для операций выполнения и записи в память. По умолчанию, субъекты запущенные в неограниченном домене не могут выделить память для записи и запустить ее. Это уменьшает степень угрозы атаки переполнения буфера `buffer overflow attacks`. Эти проверки памяти отключаются установкой Булевых переключателей, что позволяет изменять политику SELinux "на ходу". Настройка Булевых значений рассматривается позже.

Почти каждая сетевая служба ограничена. Также большинство процессов, которые запускаются в Linux с привилегиями пользователя `root` и выполняют задачи для пользователей, такие как приложение `passwd`, ограничены. Когда процесс ограничен, он запускается в своём собственном домене, например процесс `httpd` запускается в домене `httpd_t`. Если ограниченный процесс скомпрометирован атакующим, в зависимости от конфигурации SELinux, доступ атакующего к ресурсам и вред, который он может нанести ограничен.

Неограниченные (`unconfined`) процессы выполняются в неограниченных (`unconfined`) доменах, программы запускаемые `init` выполняются в неограниченном `unconfined initrc_t` домене, неограниченные процессы ядра запускаются в домене `kernel_t`. Для неограниченных процессов правила политики SELinux также применяются, но правила политики существуют для разрешения практически всех доступов для процессов,

Глава 6. Модули безопасности Linux.

запущенных в неограниченных доменах. Процессы запущенные в неограниченных доменах откатываются к использованию только правил DAC. Если неограниченный процесс скомпрометирован, SELinux не ограничивает атакующего от получения доступа к системным ресурсам и информации, но, конечно, правила DAC всё равно используются. SELinux это улучшение механизмов безопасности над DAC, но SELinux не заменяет его.

Определение контекста

- Опция `-Z` позволяет определить контекст объекта или субъекта
 - `ls -Z`
 - `ps -Z`
- В целевых политиках для предоставления доступа к ресурсам домен субъекта должен иметь права доступа к типу объекта
- Правила доступа описываются в политике
- Для стандартных сервисов уже имеются политики, разработанные майтейнерами дистрибутивов

В целевых (targeted) политиках предоставление доступа основано на анализе меток. Политика проверяет может ли домен процесса (субъекта) получить доступ к ресурсу (объекту). SELinux перехватывает системные вызовы и разрешает доступ, если политика это позволяет.

Информация о метках находится в контексте.

Основная опция получения информации о контекстах `-Z`.

Пример: Мы проверим контекст файлов, которые использует демон `apache2`. И контекст процесса веб сервера. В политиках имеется разрешение для домена `httpd_t` получать доступ к типам `httpd_sys_content_t`. Чтобы процесс получил домен `httpd_t` программа имеет тип `httpd_exec_t`.

```
root@s10:~# apt install apache2
```

```
root@s10:~# ls -Z /var/www/html/index.html
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

```
root@s10:~# ps -eZ | grep apache2
system_u:system_r:httpd_t:s0      1609 ?        00:00:00 apache2
system_u:system_r:httpd_t:s0      1610 ?        00:00:00 apache2
system_u:system_r:httpd_t:s0      1639 ?        00:00:00 apache2
```

```
root@s10:~# ls -Z /usr/sbin/apache2
system_u:object_r:httpd_exec_t:s0 /usr/sbin/apache2
```

В примере выше мы не видим самой политики. Политика представляет из себя бинарный файл, который во время старта системы загружается в ядро. Правила написания политик мы обсудим ниже.

Булевы значения (переключатели)

- Политика может предусматривать настройку разрешения или запрета на выполнение каких-либо особых действий
- Такие разрешения могут находиться в двух состояниях да или нет, поэтому называются булевыми
- Команда `semanage boolean -l` или `getsebool -a` выводят список всех булевых значений
- Команде `setsebool` устанавливает нужное значение

Переключатели позволяют изменять части политики SELinux во время работы (без перезапуска и остановки), не обладая глубоким пониманием создания политики SELinux. Это позволяет вносить изменения, такие как: разрешение доступа службам к файловым системам NFS, без перезагрузки или recompilation политики SELinux.

Для получения списка переключателей, объяснения, за что отвечает каждый переключатель, включен или выключен, необходимо выполнить команду `semanage boolean -l` от имени пользователя `root`.

Пример:

```
root@s10:~# semanage boolean -l | egrep '(Переключатель|httpd.*connect_db)'
```

Переключатель SELinux	Состояние	По умолчанию	Описание
httpd_can_network_connect_db	(выкл.,выкл.)	Determine whether scripts and	modules can connect to databases over the network.

Команда `getsebool -a` выводит список переключателей, показывает выключены они или нет, но не даёт описания, за что они отвечают.

Пример:

```
root@s10:~# getsebool -a | grep 'httpd.*connect_db'
```

httpd_can_network_connect_db --> off

Для получения статуса одного конкретного Булева значения (переключателя) `boolean-name` используется команда `getsebool boolean-name`

Пример:

```
root@s10:~# getsebool httpd_can_network_connect_db
```

httpd_can_network_connect_db --> off

Команда `setsebool boolean-name x` переводит переключатели в состояние включено или выключено, где *boolean-name* - название переключателя, а *x* - `on` для включения или `off` для выключения.

Пример:

```
root@sl0:~# setsebool httpd_can_network_connect_db on
root@sl0:~# getsebool httpd_can_network_connect_db
httpd_can_network_connect_db --> on
```

Изменения контекста файлов

- Для предоставления доступа к файлам необходимо иметь правильный контекст
- Временное назначение контекста выполняется командой `chcon`
- Восстановление контекста — `restorecon`
- Постоянный контекст управляется командой `semanage fcontext`

Чтобы некоторый ограниченный процесс получил доступ к файлу, последний, в свою очередь, должен иметь нужный тип.

Команда `chcon` устанавливает временный контекст.

Пример: Переключаем в режим Enforcing. Создаем собственный ресурс и получаем запрет доступа. После установки правильного контекста доступ появляется.

```
root@sl0:~# setenforce 1
root@sl0:~# getenforce
Enforcing

root@sl0:~# mkdir /home/myweb
root@sl0:~# echo 'My test Web Page' > /home/myweb/index.html
root@sl0:~# ls -Z /home/myweb/
unconfined_u:object_r:home_root_t:s0 index.html

root@sl0:~# cat /etc/apache2/sites-enabled/myweb.conf
Alias /myweb /home/myweb
<Directory /home/myweb>
    AllowOverride none
    Require all granted
</Directory>

root@sl0:~# curl http://127.0.0.1/myweb/index.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at 127.0.0.1 Port 80</address>
</body></html>
```

Глава 6. Модули безопасности Linux.

```
root@sl0:~# setenforce 0
root@sl0:~# curl http://127.0.0.1/myweb/index.html
My test Web Page

root@sl0:~# setenforce 1
root@sl0:~# curl http://127.0.0.1/myweb/index.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at 127.0.0.1 Port 80</address>
</body></html>

root@sl0:~# chcon -t httpd_sys_content_t /home/myweb/index.html
root@sl0:~# curl http://127.0.0.1/myweb/index.html
My test Web Page
```

Команда `restorecon` восстанавливает контекст по умолчанию.

Пример: Восстановление контекста приводит к запрету доступа.

```
root@sl0:~# restorecon -R /home/
root@sl0:~# curl http://127.0.0.1/myweb/index.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at 127.0.0.1 Port 80</address>
</body></html>

root@sl0:~# ls -Z /home/myweb/index.html
unconfined_u:object_r:user_home_t:s0 /home/myweb/index.html
```

Команда `semanage fcontext` изменяет контекст SELinux для файлов. При использовании целевой политики `targeted`, изменения вносимые данной командой, добавляются в файл `/etc/selinux/targeted/contexts/files/file_contexts`, если изменения вносятся для существующих файлов, то они добавляются в файл `file_contexts`, или добавляются файл `file_contexts.local` для новых файлов и каталогов, например при создании каталога `/web/.setfiles`, использующаяся при маркировке файловой системы и `restorecon`, использующаяся для восстановления контекста SELinux по умолчанию, читают эти файлы. Это значит, что изменения вносимые командой `semanage fcontext` постоянно, даже если файловая система будет перемаркирована. Политика SELinux контролирует возможность пользователей изменять контекст файлов.

Для внесения изменений в контекст SELinux изменений, которые сохраняются при перемаркировании файловой системы надо:

Глава 6. Модули безопасности Linux.

1. Выполнить команду `semanage fcontext -a options file-name|directory-name` Помните, что необходимо использовать полные пути к файлам и каталогам.
2. Выполнить команду `restorecon -v file-name|directory-name` для применения изменений контекста.

Пример: Применение контекста к каталогу.

```
root@sl0:~# semanage fcontext -a -t httpd_sys_content_t /home/myweb
libsemanage.add_user: user sddm not in password file
root@sl0:~# restorecon -R /home/
root@sl0:~# ls -Z /home/myweb/index.html
unconfined_u:object_r:user_home_t:s0 /home/myweb/index.html

root@sl0:~# ls -dZ /home/myweb
unconfined_u:object_r:httpd_sys_content_t:s0 /home/myweb

root@sl0:~# semanage fcontext -a -t httpd_sys_content_t '/home/myweb(/.*)?'
libsemanage.add_user: user sddm not in password file
root@sl0:~# restorecon -vR /home/myweb/
Relabeled /home/myweb/index.html from unconfined_u:object_r:user_home_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0

root@sl0:~# !curl
curl http://127.0.0.1/myweb/index.html
My test Web Page

root@sl0:~# tail -2 /etc/selinux/default/contexts/files/file_contexts.local
/home/myweb      system_u:object_r:httpd_sys_content_t:s0
/home/myweb(/.*)? system_u:object_r:httpd_sys_content_t:s0
```

Первое из показанных правил маркировки лишнее, т. к. второе правило (`/home/myweb(/.*)?`) применяется как к каталогу `/home/myweb` так и ко всему его содержимому.

Удаление перманентной маркировки файлов производится командой `semanage fcontext -d options file-name|directory-name`

Пример: Удаление лишнего правила маркировки.

```
root@sl0:~# semanage fcontext -d -t httpd_sys_content_t /home/myweb
libsemanage.add_user: user sddm not in password file
root@sl0:~# tail -2 /etc/selinux/default/contexts/files/file_contexts.local
/usr/bin/VBoxClient system_u:object_r:unconfined_execmem_exec_t:s0
/home/myweb(/.*)?   system_u:object_r:httpd_sys_content_t:s0
```

6.2.2 Язык описания правил доступа.

Зачем создавать свои политики



- Вы используете ограниченный сервис нестандартным образом
- Вы хотите создать собственный ограниченный сервис
- Расширение или изменение существующих политик для собственных нужд

SELinux это гибкий и мощный инструмент обеспечения безопасности. Основой для принятия решений в SELinux служит политика. Политику по умолчанию создают люди, которые поддерживают дистрибутив Линукса. Основу для этой политики создают разработчики программного обеспечения.

Разработчики софта и дистрибутивов, как правило, придерживаются следующих целей:

- Программы должны работать. Пользы от программы, которая абсолютна защищена и при этом не может работать, нет никакой.
- Обеспечение безопасности основано на общих принципах и соображениях. Это означает, что в конкретно вашей установленной системе, не учитываются все нюансы функционирования даже стандартных пакетов.
- В каждом дистрибутиве имеется набор предусмотренного этим дистрибутивом пакетов. Если вы используете какой-то пакет не предусмотренный разработчиками, то для него не будет политики. В этом случае пакет или не работает или попадает в категорию неограниченных, т. е. не защищается с помощью SELinux.

Исходя из выше изложенного можем сделать вывод, что рано или поздно у вас может возникнуть потребность в создании собственной политики. Основными причинами для этого могут служить следующие соображения:

- Вы используете стандартный ограниченный сервис не так как это предусматривал разработчик. В этом случае вам придется расширять стандартную политику своими правилами.

Глава 6. Модули безопасности Linux.

- Вы используете пакет, которого нет в стандартной поставке, или создаете собственный пакет. В таком случае вам потребуется создать свой модуль политики SELinux для данного программного обеспечения.
- Вы, после оценки, существующих дефолтных политик пришли к выводу, что ограничения в ней слишком либеральные и необходимо ужесточение политик.

Методология создания политик

- Реактивная — как реакция на проблему, связанную с существующей политикой
- Проактивная — политика создается на основе представлений о том, как должна работать программа

Перед созданием собственных политик, нужно понимать, что имеется два подхода для создания политик.

- Реактивный. В этом случае вы решаете какую-то конкретную проблему, связанную с неправильной работой ограниченной программы. Как результат решения проблемы вы создаете политику SELinux и загружаете ее.
- Проактивный. Вы создаете собственный ограниченный сервис и у вас имеются представления о том, что и как этот сервис должен делать. Результатом этих представлений будет созданная вами политика.

Конечно же имеется и гибридный подход, в котором, например, сначала создается политика, применяется, тестируется, на основе тестов модифицируется и снова применяется, анализируется и т.д.

Компоненты модуля политики

- Модуль бинарный файл, который создается из текстовых
 - Модуль TE — декларирование типов, описание правил.
 - Модуль FC — правила маркировки файлов
 - Модуль IF — описание интерфейса взаимодействия с этим модулем

<https://selinuxproject.org/page/RefpolicyWriteModule>

Модуль SELinux это бинарный файл. Создание этого файла основано на использовании трех текстовых файлов. Не все три файла требуется создавать для модуля.

1. Файл `.te` предназначен для описания используемых типов и описания правил.
2. Файл `.fc` описывает маркировку файлом этим модулем.
3. Файл `.if` создает интерфейс взаимодействия с модулем, на который могут ссылаться другие модули. В файле описываются макросы.

На сайте <https://selinuxproject.org/page/RefpolicyWriteModule> можно посмотреть принципы создания модулей.

Для создания собственных модулей вам потребуются пакеты `selinux-policy-dev` (в Debian) или `selinux-policy-devel` (для RedHat). Так же полезно будет установить пакет и `selinux-policy-doc`.

Синтаксис TE модуля

- TE модуль может состоять из следующих частей:
 - 1)Заголовок с названием и версией модуля (обязательный)
 - 2)Описание типов
 - 3)Политики
 - 4)Макросы

Каждый модуль `.te` может состоять из четырех составных частей:

1. Заголовок. Обязательная часть, которая задает уникальное имя модуля и версию. Версия позволяет отслеживать изменения в модуле, а так же ядру понимать, что модуль изменился и надо загрузить его новую версию.
2. Описание типов. Эти строки описывают все возможные типы или атрибуты нашего ограниченного сервиса. Необязательный компонент, если у сервиса нет собственных типов.
3. Политики. Описание правил работы сервиса. Необязательный если нет особых правил.
4. Макросы. Используют политики определенные в других модулях (через файл `.if`). Необязательный.

Пример: Модуль некоего приложения с названием `myapp`. В файле определяется название модуля — `myapp` и его версия — `1.0.0`. Затем идет декларирование типов, как напрямую строками типа `type`, так и макросами. Последняя часть — политика. Политика описывается напрямую и через макросы.

```
root@sl0:~# cat /usr/share/doc/selinux-policy/example.te
```

```
policy_module(myapp,1.0.0)
```

```
#####  
#  
# Declarations  
#
```

```
type myapp_t;  
type myapp_exec_t;
```

Глава 6. Модули безопасности Linux.

```
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t)

type myapp_log_t;
logging_log_file(myapp_log_t)

type myapp_tmp_t;
files_tmp_file(myapp_tmp_t)

#####
#
# MyApp local policy
#

allow myapp_t myapp_log_t:file { read_file_perms append_file_perms };

allow myapp_t myapp_tmp_t:file manage_file_perms;
files_tmp_filetrans(myapp_t,myapp_tmp_t,file)
```

Описание макросов можно посмотреть в каталоге

`/usr/share/selinux/devel/include/`, если установлен пакет `selinux-policy-devel`

Синтаксис FC файла

- Файл определяет контексты для маркировки файлов
- Контексты определяются макросом `gen_context`

Файл `.fc` определяет контексты, которые будут использованы для маркировки файлов, необходимых для работы сервиса.

Пример: Назначение контекста на файл программы.

```
root@sl0:~# cat /usr/share/doc/selinux-policy/example.fc
# myapp executable will have:
# label: system_u:object_r:myapp_exec_t
# MLS sensitivity: s0
# MCS categories: <none>

/usr/sbin/myapp      --      gen_context(system_u:object_r:myapp_exec_t,s0)
```

Синтаксис IF файла

- IF файл состоит из двух частей
 - 1)Описания в виде html структуры
 - 2)Описание макросов, которые будут доступны другим модулям

В файле `.if` описываются макросы, которые другие модули могут использовать для получения доступа к сервису.

Пример: Описывается два макроса: `myapp_domtrans`, который дает право переходить в домен `myapp_t` и `myapp_read_log`, который дает возможность просматривать журналы приложения. Обратите внимание, что в каждом макросе явно описываются используемые типы.

```
root@sl10:~# cat /usr/share/doc/selinux-policy/example.if
## <summary>Myapp example policy</summary>
## <desc>
##   <p>
##       More descriptive text about myapp.  The desc
##       tag can also use p, ul, and ol
##       html tags for formatting.
##   </p>
##   <p>
##       This policy supports the following myapp features:
##       <ul>
##         <li>Feature A</li>
##         <li>Feature B</li>
##         <li>Feature C</li>
##       </ul>
##   </p>
## </desc>
#

#####
## <summary>
##   Execute a domain transition to run myapp.
## </summary>
## <param name="domain">
##   <summary>
```

Глава 6. Модули безопасности Linux.

```
##      Domain allowed to transition.
##      </summary>
## </param>
#
interface(`myapp_domtrans', `
    gen_require(`
        type myapp_t, myapp_exec_t;
    ')

    domtrans_pattern($1,myapp_exec_t,myapp_t)
')

#####
## <summary>
##      Read myapp log files.
## </summary>
## <param name="domain">
##      <summary>
##          Domain allowed to read the log files.
##      </summary>
## </param>
#
interface(`myapp_read_log', `
    gen_require(`
        type myapp_log_t;
    ')

    logging_search_logs($1)
    allow $1 myapp_log_t:file read_file_perms;
')
```


Компиляция и установка модуля

- Процедура установки нового модуля
 - 1) Создаете каталог
 - 2) В это каталог копируете файлы .te, .fc и .if
 - 3) Выполняете команду
`make -f /usr/share/selinux/devel/Makefile`
 - 4) Устанавливаете полученный модуль (файл .pp) в ядро командой
`semodule -i файл.pp`
 - 5) Включаем модуль командой
`semodule -e модуль`
 - 6) Выполняем перемаркировку файлов командой `restoreconn`

После создания модуля. Его необходимо скомпилировать и установить. Процедура установки модуля следующая:

1. Создаете каталог.
2. В это каталог копируете файлы .te, .fc и .if
3. Выполняете команду
4. `make -f /usr/share/selinux/devel/Makefile`
5. Устанавливаете полученный модуль (файл .pp) в ядро командой
`semodule -i файл.pp`
6. Включаем модуль командой `semodule -e модуль`
7. Выполняем перемаркировку файлов командой `restoreconn`

Пример: Установка нового модуля.

```
root@sl0:~# mkdir myapp
root@sl0:~# cp /usr/share/doc/selinux-policy/example.* myapp/
root@sl0:~# cd myapp/
root@sl0:myapp# rename example myapp example.*
root@sl0:myapp# ls
myapp.fc myapp.if myapp.te
root@sl0:myapp# make -f /usr/share/selinux/devel/Makefile
Compiling targeted myapp module
/usr/bin/checkmodule: loading policy configuration from tmp/myapp.tmp
/usr/bin/checkmodule: policy configuration loaded
/usr/bin/checkmodule: writing binary representation (version 17) to
tmp/myapp.mod
Creating targeted myapp.pp policy package
rm tmp/myapp.mod.fc tmp/myapp.mod
root@sl0:myapp# semodule -i myapp.pp
root@sl0:myapp# semodule -e myapp
```

Удаление ненужного модуля производится командой `semodule -d модуль`

Пример:

```
root@sl0:myapp# semodule -l | grep myapp
myapp 1.0.0
root@sl0:myapp# semodule -d myapp
root@sl0:myapp# semodule -l | grep myapp
```

Реактивная политика

- SELinux проводит аудит событий, результат которого записывается в файл
`/var/log/audit/audit.log`
- Команда `audit2allow` анализирует события и предлагает создать политику для устранения запретов
- Вывод команды `audit2allow` это только информация к размышлению. Его необходимо проанализировать, и принять решение, что делать с этими запретами
- Для анализа можно использовать команду `audit2why`

При использовании SELinux производится аудит событий. В результате вы можете увидеть, как блокируется доступ к тем или иным функциям ОС. События записываются в журнал `/var/log/audit/audit.log`.

Выбрав из журнала интересующие вас события мы можем сформировать политику, которая устранил блокировки. Для этого можно воспользоваться командой `audit2allow`, которая анализирует события типа `denied` и на их основе формирует предлагаемую политику.

Пример: Формирование политики на основе анализа событий по метке `httpd_t`.

```
root@sl10:~# grep httpd_t /var/log/audit/audit.log | audit2allow
```

```
#===== httpd_t =====  
allow httpd_t home_root_t:file { getattr map open read };  
  
#!!!! This avc can be allowed using the boolean 'httpd_read_user_content'  
allow httpd_t user_home_t:file { getattr map open read };
```

В общем случае применять такие политики **ПЛОХАЯ ИДЕЯ!!!**

Лучше проанализировать этот вывод и принять решение что необходимо сделать. Для подробного анализа событий можно воспользоваться командой `audit2why`, которая показывает почему возникла ошибка и предлагает возможные решения.

Пример: Разбор событий командой `audit2why`

```
root@sl10:~# grep httpd_t /var/log/audit/audit.log | audit2why | tail -9  
type=AVC msg=audit(1738673843.372:305): avc: denied { getattr } for pid=2137  
comm="apache2" path="/home/myweb/index.html" dev="sda1" ino=131308  
scontext=system_u:system_r:httpd_t:s0  
tcontext=unconfined_u:object_r:user_home_t:s0 tclass=file permissive=0
```

Анализ подразумевает что вы отвечаете на следующие вопросы:

1. Мешает ли блокировка правильной работе ограниченного сервиса?
2. Если сервис работает некорректно, то в чем причина?
 1. Неправильная политика?
 2. Неправильная маркировка файлов?
 3. Некорректные действия пользователей?

По результатам анализа мы можем принять следующие решения:

1. Принять события и ничего не делать. Это означает, что вы не просто блокируете доступ, но и отслеживаете блокировку. Например, чтобы отслеживать атаки на службу.
2. Модифицировать существующую политику, если решили, что политика не верна.
3. Создать новый модуль и установить его. Не создавайте модули автоматически, это может привести к брешу в системе безопасности. В примере выше создавать политику с правилом `allow httpd_t user_home_t:file { getattr map open read };` не лучшее решение. Правильным решением будет установить нужную маркировку файлов.
4. Запретить аудит блокировки таких событий. Тогда вместо `allow` необходимо создать модуль с параметром `dontaudit` или использовать опцию `-D` в команде `audit2allow`

6.2.3 Реализация других форм контроля доступа с помощью SELinux.

RBAC



- Role Based Access Control использует понятие роль для которой назначаются привилегии
- Привилегии для роли назначаются политикой
- Пользователь связывается с одной или несколькими ролями

При работе с пользователями использование только меток для процессов и файлов является недостаточным. Причин несколько:

1. Как правило мы не можем заранее предусмотреть какие программы и файлы будет использовать конкретный пользователь.
2. Если мы знаем какую-то программу, которую может использовать пользователь, например nano, то не можем с достаточной долей уверенности сказать, как пользователь будет использовать эту программу. Что он будет редактировать?
3. Запуская одну и ту же программу разным пользователям надо разрешать разный тип доступа. Задачи у каждого пользователя свои.
4. В целевой политике процесс получает метку от файла программы, но это совершенно не подходит под принцип разделения доступа на основе пользовательской информации.

Для решения задач по управлению доступом пользователей мы можем воспользоваться системой RBAC (Role Based Access Control). В этой системе каждый пользователь Unix связывается с SELinux пользователем. SELinux пользователь получает список своих ролей.

Роль определяет полномочия ее обладателя. Управление ролями настраивается посредством политики.

Принципы работы RBAC

- Связь между Unix пользователем и SELinux пользователем можно посмотреть командой

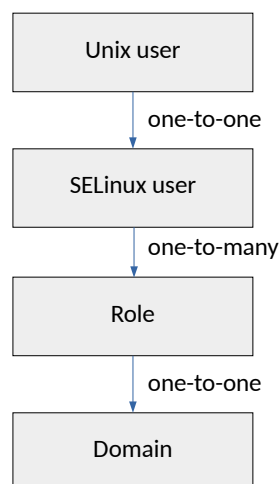
```
semanage login -l
```

- Каждый SELinux пользователь получает роли, которые можно отследить командой

```
semanage user -l
```

- Политика в отношении роли определяется модулем. Список модулей определяется командой

```
semanage module -l
```



По умолчанию система RBAC не используется. Все пользователи являются неограниченными, или, другими словами используют систему DAC.

Пример: Пользователю root назначен пользователь `unconfined_u` (неограниченный) у которого назначена роль `unconfined_r`. Метка домена пользователя `unconfined_t`.

```

root@sl0:myapp# id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
root@sl0:myapp# ps -Z
LABEL                                PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 1055 ttyS0 00:00:07 bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 3074 ttyS0 00:00:00 ps

```

Связь Unix и SELinux пользователей можно посмотреть командой `semanage login -l`. Эта команда так же показывает SELinux пользователя по умолчанию, это назначение используется для Unix пользователей, которым не привязан SELinux пользователь.

Пример: SELinux пользователь по умолчанию `unconfined_u`

```
root@sl0:myapp# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
<code>__default__</code>	<code>unconfined_u</code>	<code>s0-s0:c0.c1023</code>	*
<code>root</code>	<code>unconfined_u</code>	<code>s0-s0:c0.c1023</code>	*
<code>system_u</code>	<code>system_u</code>	<code>s0-s0:c0.c1023</code>	*

Назначенные роли SELinux пользователям можно выяснить командой `semanage user -l`.

Глава 6. Модули безопасности Linux.

Пример: В примере пользователю `staff_u` назначены роли `staff_r` `sysadm_r` `system_r` `unconfined_r`.

```
root@sl0:myapp# semanage user -l
```

SELinux User Roles	Labeling Prefix	MLS/MCS Level	MLS/MCS Range	SELinux
guest_u	user	s0	s0	guest_r
root	user	s0	s0-s0:c0.c1023	staff_r
sysadm_r	system_r unconfined_r			
staff_u	user	s0	s0-s0:c0.c1023	staff_r
sysadm_r	system_r unconfined_r			
sysadm_u	user	s0	s0-s0:c0.c1023	sysadm_r
system_u	user	s0	s0-s0:c0.c1023	system_r
unconfined_r				
unconfined_u	user	s0	s0-s0:c0.c1023	system_r
unconfined_r				
user_u	user	s0	s0	user_r
xguest_u	user	s0	s0	xguest_r

Политика для каждой роли определяется в соответствующем модуле. Команда `semanage module -l` показывает список модуле.

Пример:

```
root@sl0:myapp# semanage module -l | grep staff
staff                100                pp
```

Роли по умолчанию

Роль	Модуль	Описание
user_r	unprivuser	Базовая роль для пользователей. Может выполнять большинство операций непривилегированного пользователя.
staff_r	staff	Администраторская непривилегированная роль.
sysadm_r	sysadm	Стандартный администратор.
secadm_r	secadm	Администратор политик безопасности.
auditadm_r	auditadm	Может настраивать политику аудита и проводить аудит событий.
logadm_r	logadm	Настройка и управление журналами.
webadm_r	webadm	Настройка веб-сервера apache и, опционально, управление контентом веб-сервера.
guest_r	guest	Сильно ограниченный пользователь. Без поддержки GUI.
xguest_r	xguest	Сильно ограниченный пользователь с поддержкой GUI.
unconfined_r	unconfined	Неограниченный пользователь, за исключением защиты памяти.

В SELinux определено несколько стандартных ролей.

Каждая роль имеет индивидуальный модуль.

Вы можете использовать эти роли для выполнения стандартных системных действий.

Создание собственных ролей

- Макросы `userdom_unpriv_user_template(myrole)` и `userdom_admin_user_template(myrole)` могут использоваться как шаблоны при создании собственных ролей
- При создании собственных ролей вам, помимо создания модуля, возможно, потребуется внести изменения в файлы в каталоге `/etc/selinux/targeted/contexts` :
 - `default_type` — контекст по умолчанию для роли
 - `default_contexts` — поведение SELinux программ для выбора контекста пользователю

Если существующие роли не удовлетворяют ваших потребностей, то вы можете создать свою собственную роль. Для этого вам потребуется:

1. Создать модуль SELinux с описанием политики для вашей роли. При создании модуля вы можете воспользоваться макросами-шаблонами
 1. `userdom_unpriv_user_template(myrole)`: макрос для определения роли похожей на `user_r` и `staff_r`.
 2. `userdom_admin_user_template(myrole)`: макрос для определения роли похожей на `sysadm_r`.
2. Внести изменения в файлы в каталоге `/etc/selinux/targeted/contexts`:
 1. `default_type` — контекст по умолчанию для роли
 2. `default_contexts` — поведение SELinux программ для выбора контекста пользователю.

Использование RBAC

- Создайте Unix пользователя с привязкой к SELinux пользователю командой `useradd -Z` или свяжите существующего пользователя командой `semanage login -a`
- Вы можете изменить роль пользователей по умолчанию командой `semanage login -m -s "пользователь_u" -r "s0" __default__`

Создать Unix пользователя с желаемой привязкой к SELinux пользователю можно командой `useradd` с опцией `-Z`.

Если вы хотите сопоставить существующих пользователей, то нужно использовать команду `semanage login -a`.

Пример: Создание Unix пользователя с привязкой к SELinux пользователю.

```
root@sl0:myapp# useradd -Z staff_u staffuser
root@sl0:myapp# passwd staffuser
Changing password for user staffuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
root@sl0:myapp# gpasswd -a staffuser wheel
Adding user staffuser to group wheel
root@sl0:myapp# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
__default__	unconfined_u	s0-s0:c0.c1023	*
root	unconfined_u	s0-s0:c0.c1023	*
staffuser	staff_u	s0-s0:c0.c1023	*
system_u	system_u	s0-s0:c0.c1023	*

Пример: Проверка возможностей ограниченного пользователя.

```
root@sl0:myapp# ssh staffuser@127.0.0.1
staffuser@127.0.0.1's password:
[staffuser@sl0 ~]$ id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
[staffuser@sl0 ~]$ ps -Z
LABEL                                PID TTY          TIME CMD
staff_u:staff_r:staff_t:s0-s0:c0.c1023 4059 pts/0    00:00:00 bash
staff_u:staff_r:staff_t:s0-s0:c0.c1023 4080 pts/0    00:00:00 ps
```

Глава 6. Модули безопасности Linux.

```
[staffuser@sl0 ~]$ su -
Password: Правильный пароль
su: Authentication failure
[staffuser@sl0 ~]$ sudo su -
[sudo] password for staffuser:
su: avc.c:74: avc_context_to_sid_raw: Assertion `avc_running' failed.
[staffuser@sl0 ~]$ sudo -i
-bash: /root/.bash_profile: Permission denied
-bash-4.2# id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
bash-4.2# passwd root
passwd: SELinux denying access due to security policy.
bash-4.2# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=30.5 ms
#
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1001ms
rtt min/avg/max/mdev = 30.590/30.590/30.590/0.000 ms
bash-4.2# wget http://www.google.com -O /dev/null
--2017-08-02 20:53:56-- http://www.google.com/
Resolving www.google.com (www.google.com)... 173.194.221.103, 173.194.221.105,
173.194.221.106, ...
Connecting to www.google.com (www.google.com)|173.194.221.103|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://www.google.ru/?gfe_rd=cr&ei=lfWBWbe4DsyG7gTYn5u4Aw [following]
--2017-08-02 20:53:56-- http://www.google.ru/?
gfe_rd=cr&ei=lfWBWbe4DsyG7gTYn5u4Aw
Resolving www.google.ru (www.google.ru)... 173.194.222.94,
2a00:1450:4010:c0b::5e
Connecting to www.google.ru (www.google.ru)|173.194.222.94|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: '/dev/null'

[ <=> ] 10,960 --.-K/s in 0.001s

2017-08-02 20:53:56 (7.81 MB/s) - '/dev/null' saved [10960]
-bash-4.2# logout
```

Переключение на новую роль

```
[staffuser@sl0 ~]$ sudo -r sysadm_r -i
root@sl0:~# id -Z
staff_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
root@sl0:~# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
passwd: all authentication tokens updated successfully.
root@sl0:~# ping 8.8.8.8 -c1
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=30.8 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

Глава 6. Модули безопасности Linux.

```
rtt min/avg/max/mdev = 30.802/30.802/30.802/0.000 ms
root@sl0:~# wget http://www.google.com -O /dev/null
--2017-08-02 20:53:31-- http://www.google.com/
Resolving www.google.com (www.google.com)... 173.194.222.105, 173.194.222.103,
173.194.222.147, ...
Connecting to www.google.com (www.google.com)|173.194.222.105|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://www.google.ru/?gfe_rd=cr&ei=fPWBWZPSE6rG7gTF0re4Dg [following]
--2017-08-02 20:53:31-- http://www.google.ru/?
gfe_rd=cr&ei=fPWBWZPSE6rG7gTF0re4Dg
Resolving www.google.ru (www.google.ru)... 173.194.221.94,
2a00:1450:4010:c0b::5e
Connecting to www.google.ru (www.google.ru)|173.194.221.94|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: '/dev/null'

[ <=> ] 10,944 --.-K/s in 0s

2017-08-02 20:53:31 (98.7 MB/s) - '/dev/null' saved [10944]

root@sl0:~# logout
```

Для сопоставления существующего Unix пользователя с SELinux пользователем можно использовать команду `semanage login -m`:

```
root@sl0:myapp# semanage login -m -s "user_u" -r "s0" __default__
root@sl0:myapp# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
__default__	user_u	s0	*
root	unconfined_u	s0-s0:c0.c1023	*
staffuser	staff_u	s0-s0:c0.c1023	*
system_u	system_u	s0-s0:c0.c1023	*

MLS/MCS

- В системе MLS или MCS используются дополнительные целочисленные атрибуты контекста:
 - sensitivity (чувствительность) — уровень доступа
 - category (категория) — категория
- Чувствительность — число определяющее уровень доступа. При анализе чувствительности оценивается величина чувствительности объекта и субъекта.
- Категория — число, которое дает право на взаимодействие с другими категориями, вне зависимости от величины.

Системы MLS или MCS используют, в дополнение к меткам, еще и числовой анализ при предоставлении доступа.

Метка чувствительность (sensitivity) задает уровень доступа. Общая идея при использовании чувствительности: некий уровень может прочитать информацию уровнем ниже своего, но не может ее изменить; на своем уровне пользователь может читать и изменять данные; на уровень выше своего может только изменять (предоставлять) информацию, но не читать.

Использование категорий показывает какой диапазон категорий может получить доступ к другим категориям.