OTUS

# SELinux – когда всё запрещено

otus.ru


**Проверить, идет ли запись**

# Меня хорошо видно && слышно?

# Преподаватель

## Лавлинский Николай

Технический директор «Метод Лаб»

Более 15 лет в веб-разработке

Преподавал в ВУЗе более 10 лет
Более 4 лет в онлайн-образовании

Специализация: оптимизация производительности, ускорение сайтов и веб-приложений

https://www.methodlab.ru/
https://www.youtube.com/c/NickLavlinsky
https://www.youtube.com/@site_support
https://vk.com/nick.lavlinsky

# Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Slack #general

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

## Условные обозначения

Индивидуально

Время, необходимое
на активность

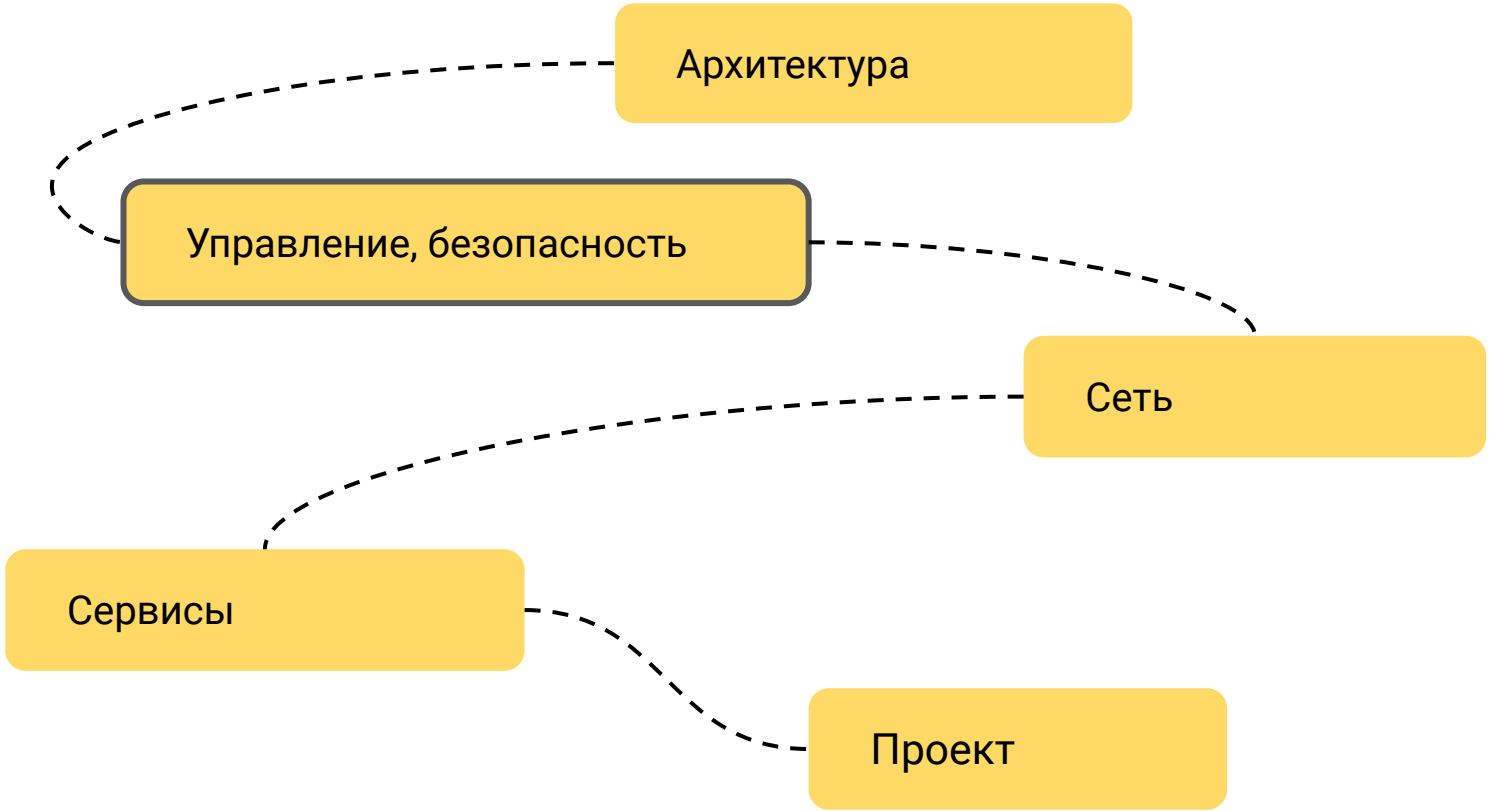
Пишем в чат

Говорим голосом

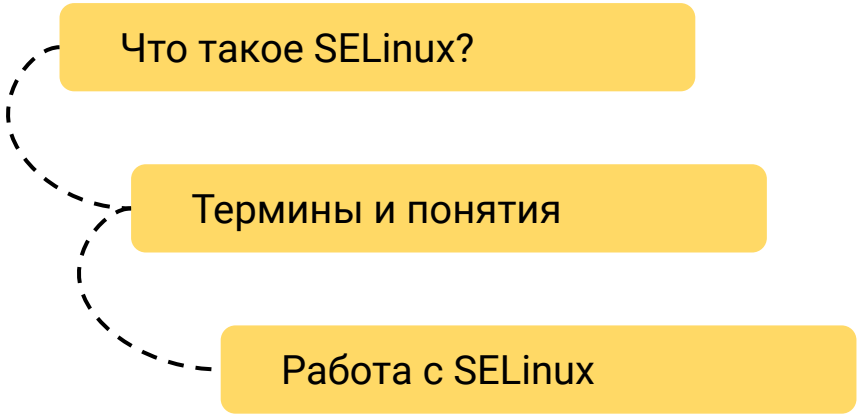
Документ

Ответьте себе или
задайте вопрос

# Карта курса


Архитектура
Управление, безопасность
Сеть
Сервисы
Проект

# Маршрут вебинара



# Цели вебинара

**После занятия вы сможете**

1. Понять, что такое системы принудительного контроля доступа
2. Узнать SELinux поближе и познакомиться с инструментами для работы с ним
3. Перестать бояться SELinux и научиться управлять его политиками

# Смысл

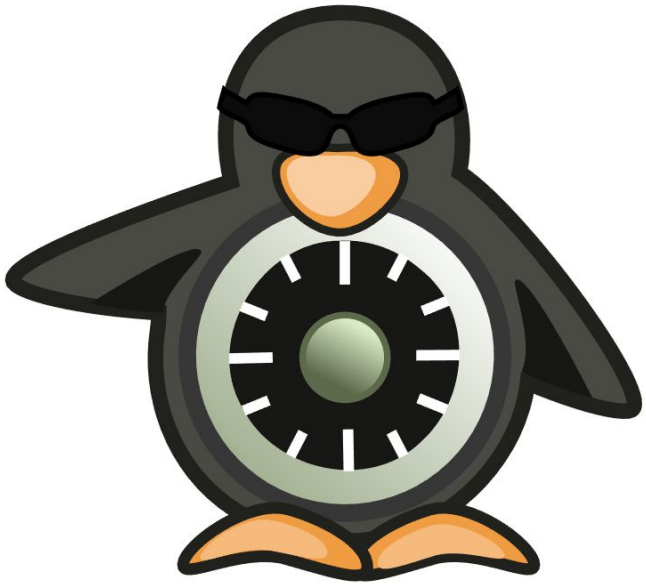
## Зачем вам это уметь

1. Чтобы понимать, как работает система SELinux и на что она влияет
2. Чтобы отлаживать процесс работы сервисов и приложений с учетом взаимодействия с SELinux
3. Чтобы усилить меры безопасности вашей инфраструктуры

# История создания SELinux

# История SELinux

- **SELinux** (англ. Security Enhanced Linux) — система принудительного (мандатного) контроля доступа (Mandatory Access Control)
- Разработана в АНБ (NSA – National Security Agency)
- Первый релиз – 1998
- Фреймворк **LSM** (Linux Security Modules — модули безопасности Linux) – 2003
- Первый релиз AppArmor – 2009



# Mandatory Access Control

**Что такое, зачем?**

# MAC и DAC в Linux

- Дискреционный механизм доступа (DAC, Discretionary Access Control)
  - Каждый файл имеет владельца
  - Владелец может передавать права
  - Владелец, группа, остальные
  - Расширение: Posix ACL
- Мандатный механизм доступа (MAC, Mandatory Access Control, матричное управление доступом)
  - Явные права на объекты (файлы, устройства, сокеты, порты, процессы)
  - Права определяются политиками, а не владельцем
  - Модель управления правами домен-тип (домен процесса, тип данных)
  - Вариант в Debian: `AppArmor`

# SELinux — реализация MAC

- Гибкое ограничение прав пользователей и процессов на уровне ядра
- Работа совместно с DAC
- Снижение риска эксплуатации уязвимостей приложений
- Ограничение прав опасных процессов
- Протоколирование действий

# SELinux — особенности

- Сложная система политик
- Непонятное разделение ответственности за создание политик
- Каждый ресурс должен быть связан с сервисом
- Высокая вероятность ошибки конфигурации
- Нет инструментов для работы с политиками “из коробки”
- Низкие накладные расходы
- Безопасность никогда не бывает удобной

# Термины SELinux

# Термины SELinux

- **Субъект** — пользователь или процесс, то есть то, что выполняет действия в системе
- **Объект** — то над чем выполняются действия, то есть файлы, порты, сокеты и прочее
- Режимы (policy) SELinux
  - **`targeted`** – набор политик по умолчанию (включает MCS)
  - `minimum` – вариант `targeted`, минимальный набор политик
  - `mls` – MLS (уровни секретности)

# Механизмы мандатного доступа

- **MLS** (Multi-Level Security, многоуровневая система безопасности)
  - Модель Белла-Лападулы
  - Уровни доступа (секретности)
  - Объекты маркируются уровнями доступа
- **MCS** (Multi-Category Security, мультикатегорийная система безопасности)
  - Данные разбиты на категории
  - Объектам назначаются метки категорий
- **RBAC (Roles Based Access Control)** — управление доступом на основе ролей
- **TE** (type Enforcement) — принудительная типизация доступа

# MLS – уровни секретности

- Все субъекты и объекты имеют свой уровень допуска
- Субъект с определенным уровнем допуска имеет право читать и создавать (писать/обновлять) объекты с тем же уровнем допуска
- Кроме того, он имеет право читать менее секретные объекты и создавать объекты с более высоким уровнем
- Субъект никогда не сможет создавать объекты с уровнем допуска ниже, чем он сам имеет, а также прочесть объект более высокого уровня допуска
- Краткая формулировка: «write up, read down» и «no write down, no read up»
- Применяется при повышенных требованиях к безопасности (гос. и военные)
- Работает в режиме `mls`

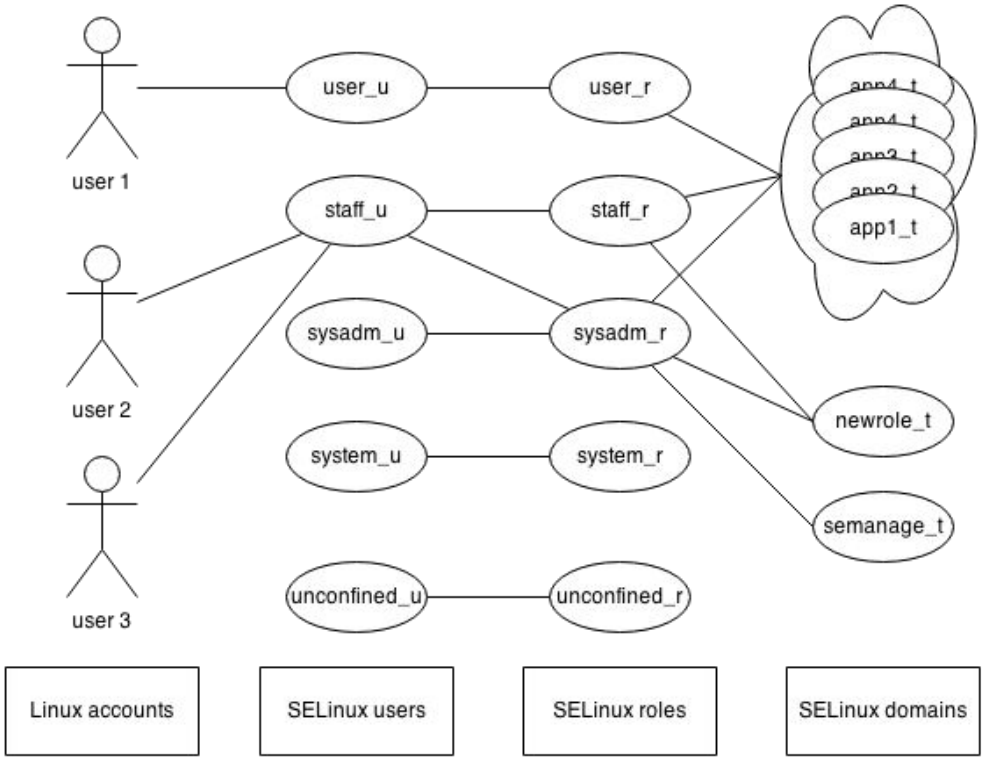
# MCS – категории объектов

- Все субъекты и объекты имеют свою категорию
- Субъект получает доступ к своим разрешенным категориям
- Метки категорий расставляются по объектам и субъектам
- Работает в режиме `targeted`

# RBAC — управление по ролям

- Контроль доступа к объектам файловой системы через роли, созданные на основании требований бизнеса или других критериев
- Роли могут быть разных типов и уровней доступа к объектам
- Пользователи по умолчанию:
  - `system_u` - системные процессы
  - `root` - системный администратор
  - `user_u` - все логины пользователей

# RBAC — управление по ролям



# RBAC — роли по умолчанию

- `user_r` — роль обычного пользователя, разрешает запуск пользовательских приложений и других непривилегированных доменов
- `staff_r` — похожа на роль `user_r`, но позволяет получать больше системной информации, чем обычный пользователь, эта роль выдается пользователям, которым следует разрешить переключение на другие роли
- `sysadm_r` — административная роль, разрешает использование большинства доменов, включая привилегированные
- `system_r` — системная роль, не предназначенная для непосредственного переключения

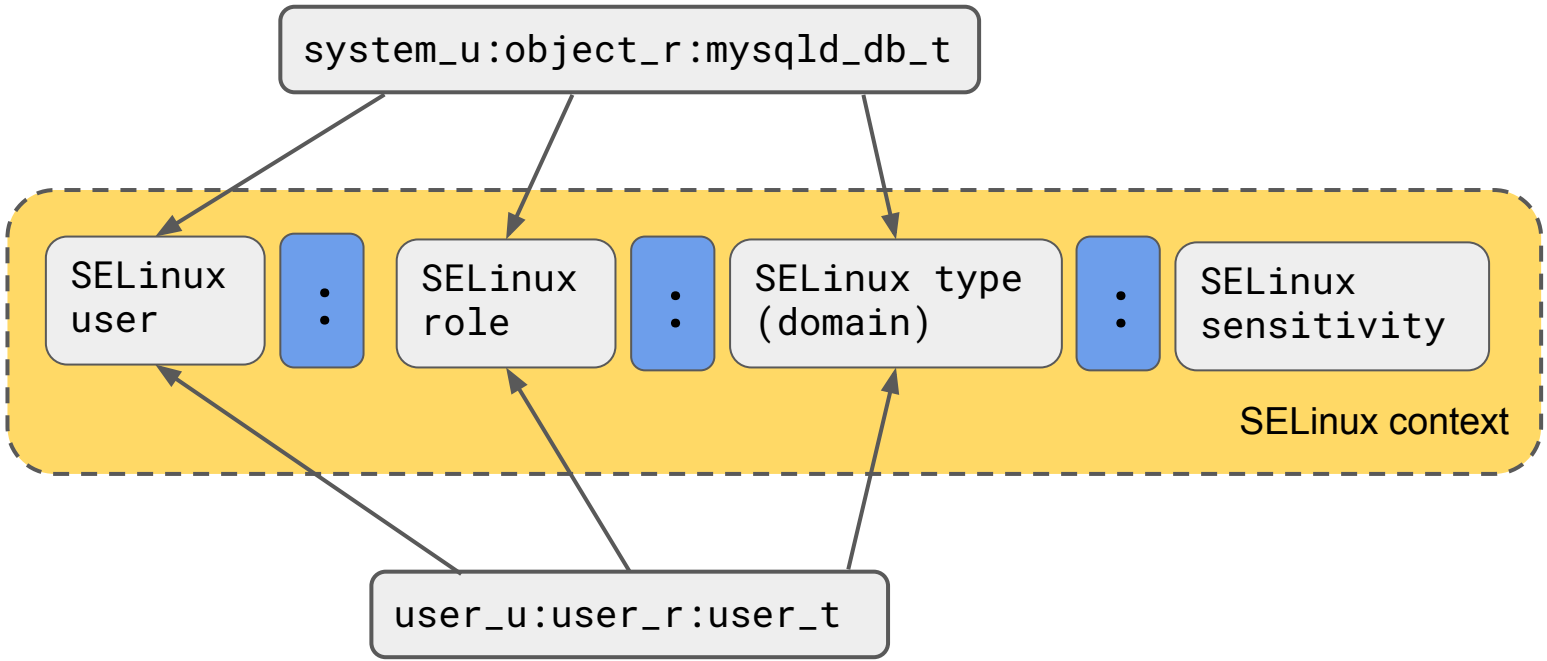
# SELinux — термины MAC

- **TE** (Type Enforcement, принудительная типизация доступа)
- **Контекст безопасности** (`context`) - по сути та же метка, выглядит как строка переменной длины и хранится в расширенных атрибутах файловой системы. Объединяет в себе роли, типы и домены
- **Домен** (`domain`) - список действий, которые может выполнять процесс по отношению к различным объектам
- **Тип** (`type`) - атрибут объекта, который определяет, кто может получить к нему доступ
- **Роль** - атрибут, который определяет, в какие домены может входить пользователь, то есть какие домены пользователь имеет право запускать

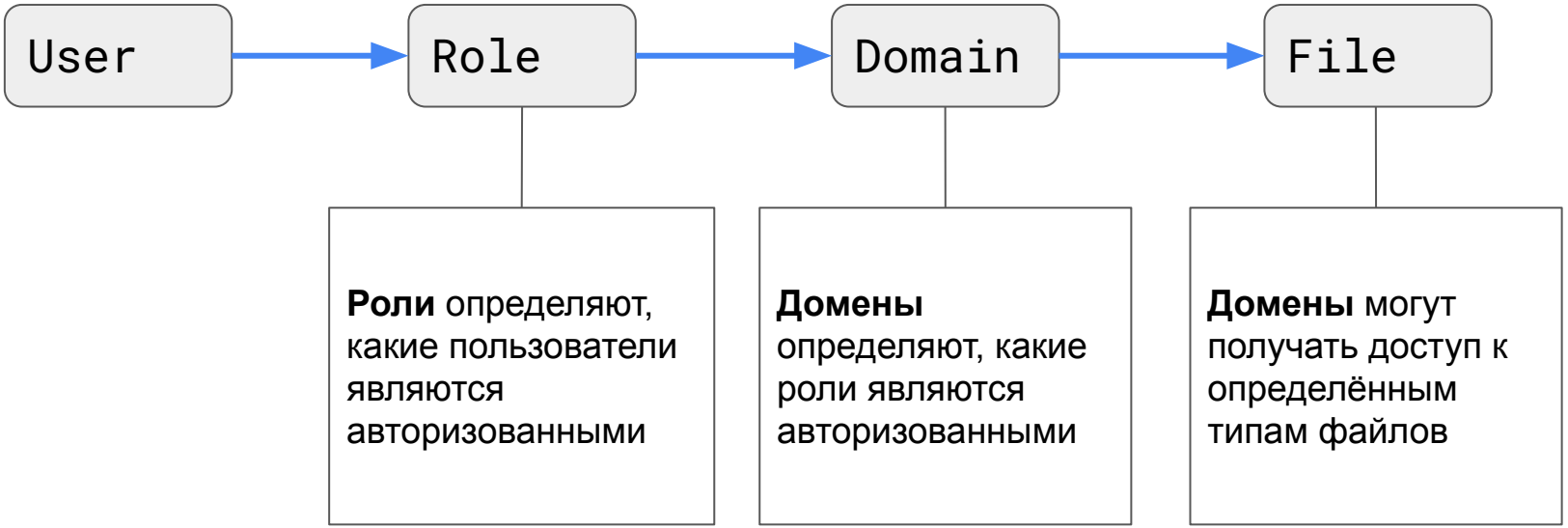
# TE – Type Enforcement

- **TE** (Type Enforcement, принудительная типизация доступа)
- **Контекст безопасности** (`context`) - по сути та же метка, выглядит как строка переменной длины и хранится в расширенных атрибутах файловой системы. Объединяет в себе роли, типы и домены
- **Домен** (`domain`) - список действий, которые может выполнять процесс по отношению к различным объектам
- **Тип** (`type`) - атрибут объекта, который определяет, кто может получить к нему доступ
- **Роль** - атрибут, который определяет, в какие домены может входить пользователь, то есть какие домены пользователь имеет право запускать

# TE – Type Enforcement



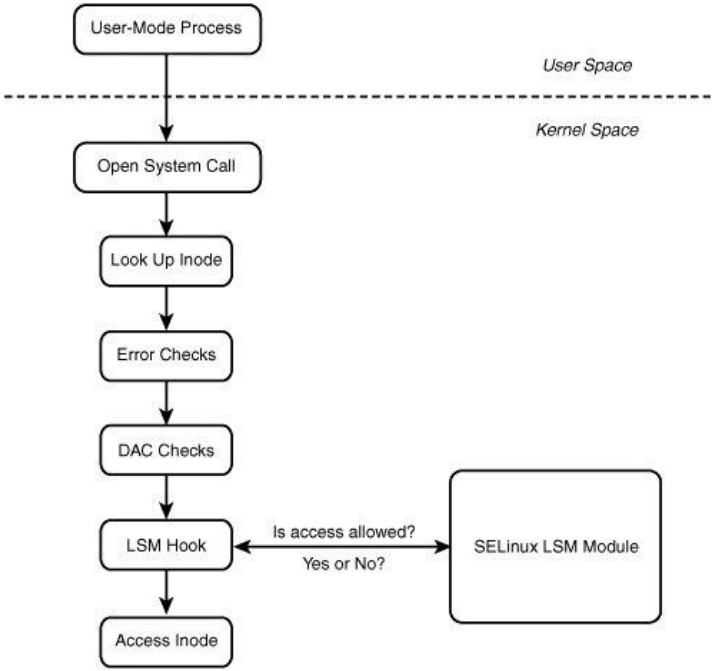
# TE – Type Enforcement



# TE – Type Enforcement

- **Роль** — набор правил
- **Домен** — то, что разрешено процессу (субъекту)
- **Тип** — набор правил для файла (объекта)
- **Суть работы** — сопоставление домена с типом через роль

# Совместная работа DAC и MAC (LSM)



# Инструменты управления SELinux

# SELinux – инструменты

- Пакет `setools-console`
  - `sesearch`
  - `seinfo`
  - `findcon`
  - `getsebool`
  - `setsebool`
- Пакет `policycoreutils-python`
  - `audit2allow`
  - `audit2why`
- Пакет `policycoreutils-newrole`
  - `newrole`
- Пакет `selinux-policy-mls`
  - `selinux-policy-mls`

# Наследование типов в SELinux

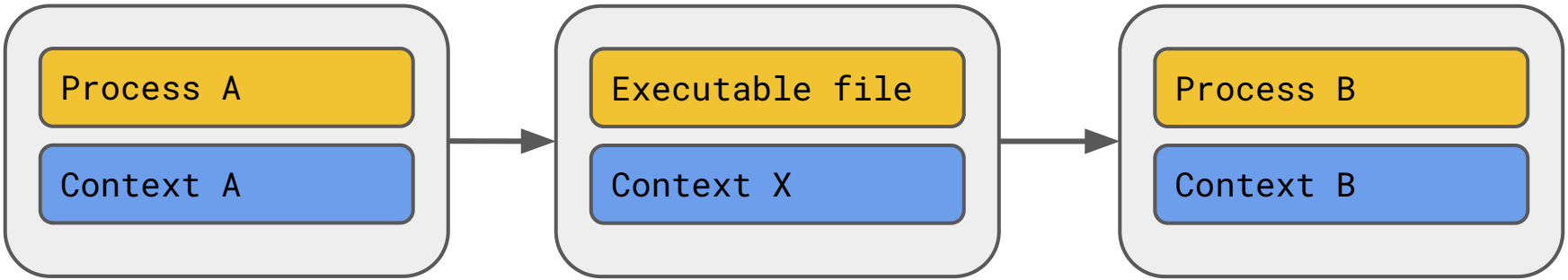
- При создании файла в каталоге он наследует его тип
- Переход контекста (`context transition`) происходит при наличии условий
    1. Целевой контекст файла является исполняемым для исходного домена
    2. Целевой контекст файла помечен как точка входа для целевого домена
    3. Исходный домен разрешен для перехода в целевой домен

# Переход контекста

Процесс A с контекстом A запускает файл

Точка входа для контекста B

Запущенный код создаёт процесс B

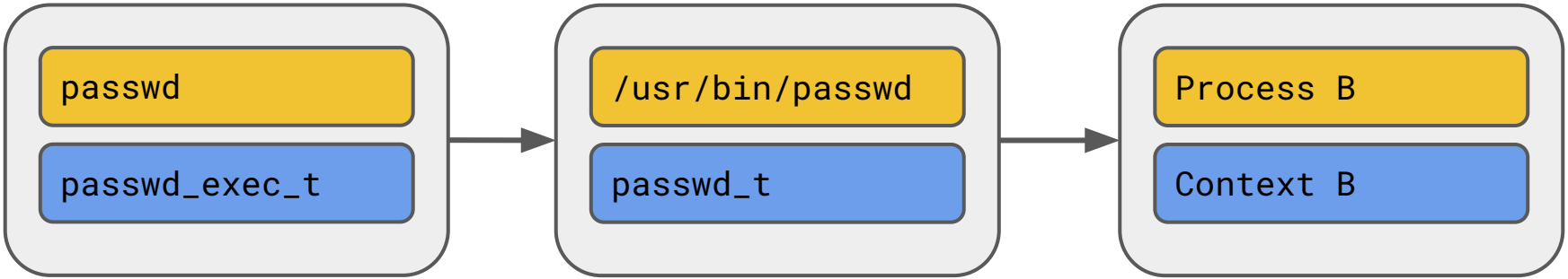


# Переход контекста – пример

Пользователь запускает **/usr/bin/passwd**

Оболочка переходит в домен **passwd_t**

Процессы в домене **passwd_t** получают доступ к файлам в домене **shadow_t**



# Описание примера

- В домен passwd_t можно войти выполнением приложений, маркированных типа passwd_exec_t
- Только авторизованные домены passwd_t, могут записывать в файлы маркированные типом shadow_t
- Только авторизованные домены могут выполнять переход в домен passwd_t. Например, процесс sendmail запускается в домене sendmail_t и не имеет законных причин для запуска passwd, таким образом он не может выполнить переход в домен passwd_t
- Процессы запущенные в домене passwd_t могут читать и записывать только в авторизованные файлы промаркированные типом etc_t или shadow_t. Это предотвращает приложение passwd от записи или чтения в другие файлы

# Вопросы по SELinux

- **Если нужно запустить самосборное приложение?**
  - Если нужно запустить несговорчивое или сомнительное приложение - запускать его надо из каталога /opt, в нем SELinux не работает
- **А где же лежат все эти контексты и как они выглядят?**
  - Контексты лежат вот по этому пути: /etc/selinux/targeted/contexts/files
- **Контексты (и политики) уже есть в системе?**
  - Да, об этом позаботились создатели дистрибутива и разработчики приложений

# Команды SELinux

- **`ls -Z /root`** — просмотр контекста безопасности каталога
- **`semanage login -l`** – информация о правах пользователей
- **`ls -Z /usr/sbin/nginx`** — контекст безопасности объекта
- **`ps -Z 12345`** — контекст безопасности процесса
- **`sesearch -A -s httpd_t | grep 'allow httpd_t'`** – разрешающие правила для типа `httpd_t`
- **`sesearch -s httpd_t -t httpd_exec_t -c file -p execute -Ad`** – ищем правила преобразования по типам

# Режимы работы SELinux

# Режимы работы

- Конфиг: `/etc/selinux/config`
- Статус: `sestatus` или `getenforce`
- Отключение: `setenforce 0`
- Включение: `setenforce 1`

# Изменение контекстов

- Меняем (временно) тип в контексте каталога: `chcon -R -t type /home/user`
- Проверяем контекст каталога: `ls -Z /home/user`
- Восстанавливаем контекст каталога: `restorecon -v /home/user`
- Постоянное изменение контекста:
    - `semanage fcontext -a options file-name|directory-name`
    - `restorecon -v file-name|directory-name`

# Создание модуля на основе лога

1. Очищаем `audit.log: echo > /var/log/auditd/audit.log`
2. Включаем в SELinux режим permissive: `setenforce 0`
3. Запускаем приложение и получаем ошибки в `audit.log`
4. Смотрим ошибки и рекомендации в audit.log:

   ```
   audit2why < /var/log/audit/audit.log
   ```
5. Формируем модуль с правилами для SELinux из данных лога

   ```
   audit2allow -M httpd_add --debug < /var/log/audit/audit.log
   ```
6. Загружаем модуль

   ```
   semodule -i httpd_add.pp
   ```

# Параметризованные политики

# Параметризованные политики

- Представляют из себя политики, которые описаны переменными с булевым типом (`on/off`)
- Управляются утилитами: `getsebool` и `setsebool`
- Просмотр политик в отношении сервиса samba:
  `getsebool -a | grep samba`
- Меняем значение выбранной политики (постоянно):
  `setsebool -P samba_share_fusefs on`

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Практика

# Домашнее задание 1

1. Запустить nginx на нестандартном порту 3-мя разными способами:
   - переключатели setsebool;
   - добавление нестандартного порта в имеющийся тип;
   - формирование и установка модуля SELinux.
2. Описать все решения в README репозитория.

В чат ДЗ отправьте ссылку на ваш git-репозиторий.

**Сроки выполнения: указаны в личном кабинете**

# Домашнее задание 2

1. Обеспечить работоспособность приложения при включенном selinux.
   - развернуть приложенный стенд https://github.com/mbfx/otus-linux-adm/tree/master/selinux_dns_problems ;
   - выяснить причину неработоспособности механизма обновления зоны (см. README);
   - предложить решение (или решения) для данной проблемы;
   - выбрать одно из решений для реализации, предварительно обосновав выбор;
   - реализовать выбранное решение и продемонстрировать его работоспособность.

2. Составить README с анализом причины неработоспособности, возможными способами решения и обоснованием выбора одного из них;

3. Прислать исправленный стенд или демонстрацию работоспособной системы скриншотами и описанием

# Что мы изучили?

## Подведем итоги

1. Системы мандатного контроля доступа
2. Особенности работы SELinux
3. Рассмотрели инструменты работы с политиками

# Список материалов для изучения

1. https://freedesktop.org/wiki/Software/systemd/
2. https://www.freedesktop.org/software/systemd/man/systemd.exec.html
3. https://www.freedesktop.org/software/systemd/man/index.html

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Рефлексия

# Цели вебинара

## Проверка достижения целей

1. Понять, что такое системы принудительного контроля доступа
2. Узнать SELinux поближе и познакомиться с инструментами для работы с ним
3. Перестать бояться SELinux и научиться управлять его политиками

# Рефлексия

Что было самым полезным на занятии?

Как будете применять на практике то, что узнали на вебинаре?

# Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

# Приходите на следующие вебинары

**Лавлинский Николай**

Технический директор “Метод Лаб”

https://www.methodlab.ru/
https://www.youtube.com/c/NickLavlinsky
https://www.youtube.com/@site_support
https://vk.com/nick.lavlinsky