МЕХАНИЗМЫ БЕЗОПАСНОСТИ И ИЗОЛЯЦИИ ПРИЛОЖЕНИЙ В МОБИЛЬНОЙ ОС АВРОРА

Андрей Шитов, руководитель команды разработки ОС
Открытая мобильная платформа

ОТКРЫТАЯ МОБИЛЬНАЯ ПЛАТФОРМА

Открытая мобильная платформа

- Офисы разработки в Москве, Санкт-Петербурге и Иннополисе
- Дочерняя компания ПАО «Ростелеком»
- Образовательные программы на online площадках и в ведущих ВУЗах страны
- Участники open source проектов и организаций

Открытая мобильная платформа

Продукты и направления:

- Мобильная операционная система Аврора
- EMM-решение Аврора Центр
- Доверенная среда исполнения Аврора TEE
- Средство доверенной загрузки Аврора СДЗ

Все продукты компании в Реестре отечественного ПО.

АВРОРА

АВРОРА ЦЕНТР

АВРОРА ДОВЕРЕННАЯ СРЕДА

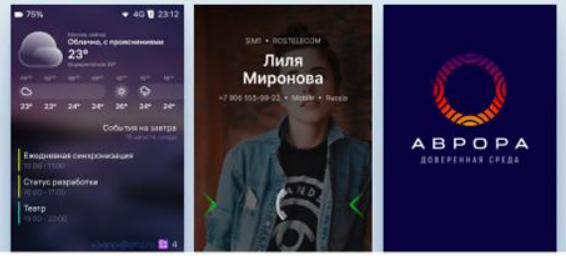
АВРОРА СРЕДСТВО ДОВЕРЕННОЙ ЗАГРУЗКИ

Доверенная мобильная платформа

ОПЕРАЦИОННАЯ СИСТЕМА

Четвертое поколение

Современный мобильный функционал
с фокусом на безопасности



АВРОРА
СВОЯ СИСТЕМА

более 400 000 устройств

в промышленной эксплуатации

ЭКОСИСТЕМА ПРИЛОЖЕНИЙ

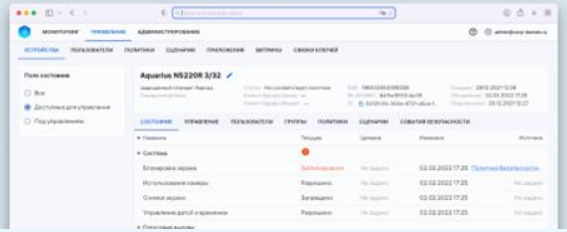
Более 100 партнеров

создают свои приложения
под ОС Аврора

МОБИЛЬНЫЕ СЕРВИСЫ И MDM

Аврора Центр

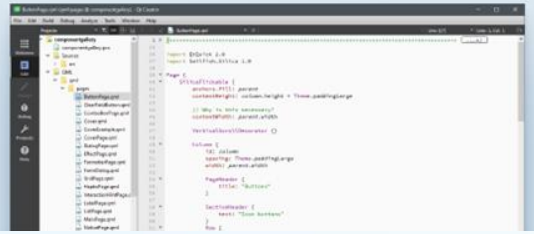
Управление парком устройств
Push-сервис, сервис обновлений,
магазин приложений



СРЕДСТВА РАЗРАБОТКИ

Аврора SDK

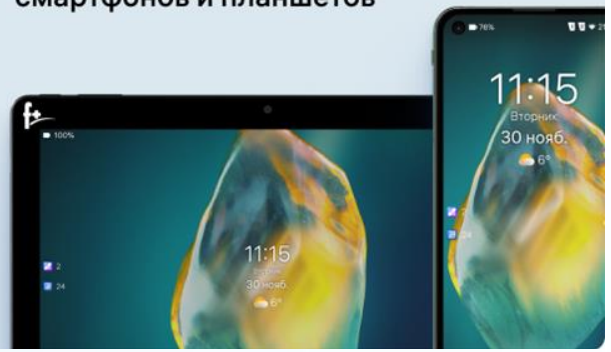
для Windows, MacOS и Linux



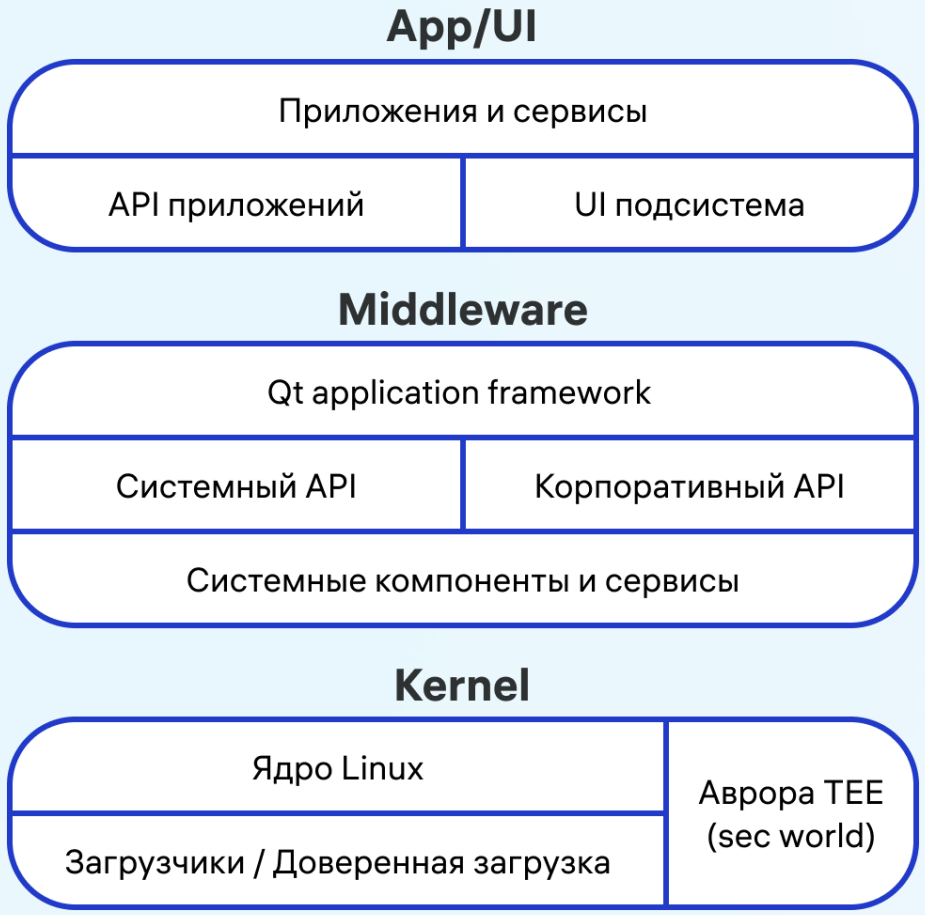
МОДЕЛЬНЫЙ РЯД

10 устройств

смартфонов и планшетов



ОС Аврора



- Linux ядро
- GNU/Linux userland
- Wayland
- Qt Framework
- RPM пакетный менеджер

- 1600+ пакетов open-source и собственных
- 10 новых устройств за 2 года
- SDK для Linux, Windows и MacOS

Механизмы безопасности и изоляции приложений в мобильной ОС Аврора

- Цель данной презентации – обзор угроз и доступных механизмов защиты приложений ОС Аврора
- Общие подходы к безопасности в мобильной ОС Аврора были изложены в докладе OS Day 2021-го года: «Развитие механизмов безопасности мобильной ОС Аврора»
- ОС Аврора обеспечивает прикладную эшелонированную безопасность
  - Прикладная – все механизмы сконфигурированы и работают изначально, не требуют дорогостоящей настройки администратором или интегратором
  - Эшелонированная – используется комплекс различных мер и методов, которые пересекаются и дополняют друг друга

Модель угроз: от кого защищаемся?

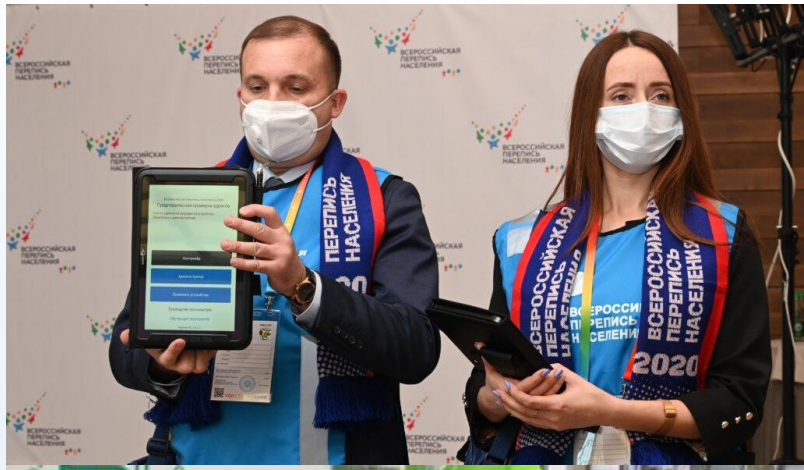
Устройство под управлением ОС Аврора в корпоративных сценариях - это рабочий терминал сотрудника определенной организации.

Источники угроз:

- Кража/потеря устройства
- Jailbreak/перепрошивка устройства
- Доступ посторонних к данным на устройстве
- Недоверенные сторонние приложения
- Перехват данных, передаваемых по сети
- ....

Отличие от десктопа:

- Мобильность, нахождение устройства в различных окружениях
- Частое использование устройств, работа с персональными данными и чувствительной информацией пользователя



Категории угроз и направления атак

Категории угроз:

- Физический доступ
- Сетевое взаимодействие
- Исполнение недоверенного кода и обработка недоверенного контента

Тактики атак в ATT&CK Enterprise:

- Разведка
- Первоначальный доступ
- Выполнение зловредного кода
- Закрепление
- Повышение привилегий
- Обход защиты
- Получение учетных данных
- Обнаружение узлов системы/инфраструктуры
- Распространение внутри системы/инфраструктуры
- Сбор данных
- Управление и контроль агентом
- Эксфильтрация
- Воздействие для отказа в обслуживании

Эшелонированная безопасность ОС Аврора

1 Защита от установки и запуска недоверенного кода на устройстве

- Подпись пакетов
- Динамический контроль целостности исполняемых файлов (IMA)
- Валидация используемого API

2 Защита от последствий выполнения недоверенного кода на устройстве

- Изоляция по namespace
- Разрешения приложений (permissions)
- Запрет небезопасных системных вызовов (seccomp)
- Запрет выполнения JIT кода под root
- YAMA
- Хранилище ключей в Аврора TEE
- Включение доступных механизмов защиты на уровне компилятора

3 Обнаружение и реагирование на последствия исполнения недоверенного кода

- Сервисы securityd/integrityd
- Доверенная загрузка (Secure boot)
- ATIC (Aurora Trusted Integrity Checker)
- Kernel Self Protection

Механизмы безопасности для приложений

Категории угроз:

- Модификация файлов приложения
- Доступ к данным приложения средствами ОС или другими приложениями
- Доступ к закэшированным данным во временной памяти
- Перехват событий ввода, доступ к буферу обмена, снимкам экрана
- Использование API, предоставляемых другими приложениями, в злонамеренных целях
- Влияние приложений на безопасность и стабильность системы через доступные API
- Использование уникальных идентификаторов устройства для нацеленных атак
- Использование средств отладки и трассировки приложений

Средства защиты:

1
- Подпись пакетов
- Валидация используемого API
- Динамический контроль целостности исполняемых файлов (IMA)

2
- Изоляция в «песочнице» и безопасные методы IPC
- Разрешения приложений (permissions)
- Разделяемые каталоги
- Keystore и crypto API, криптоконтейнер
- Опции компилятора, YAMA
- Pseudo-device id

3
- Контроль целостности файлов приложения (Integrityd)

План доклада

1. Изоляция приложений в ОС Аврора
2. Разрешения приложений
3. Механизмы IPC для изолированных приложений
4. Варианты хранения данных для изолированных приложений
5. API для контроля целостности файлов приложений

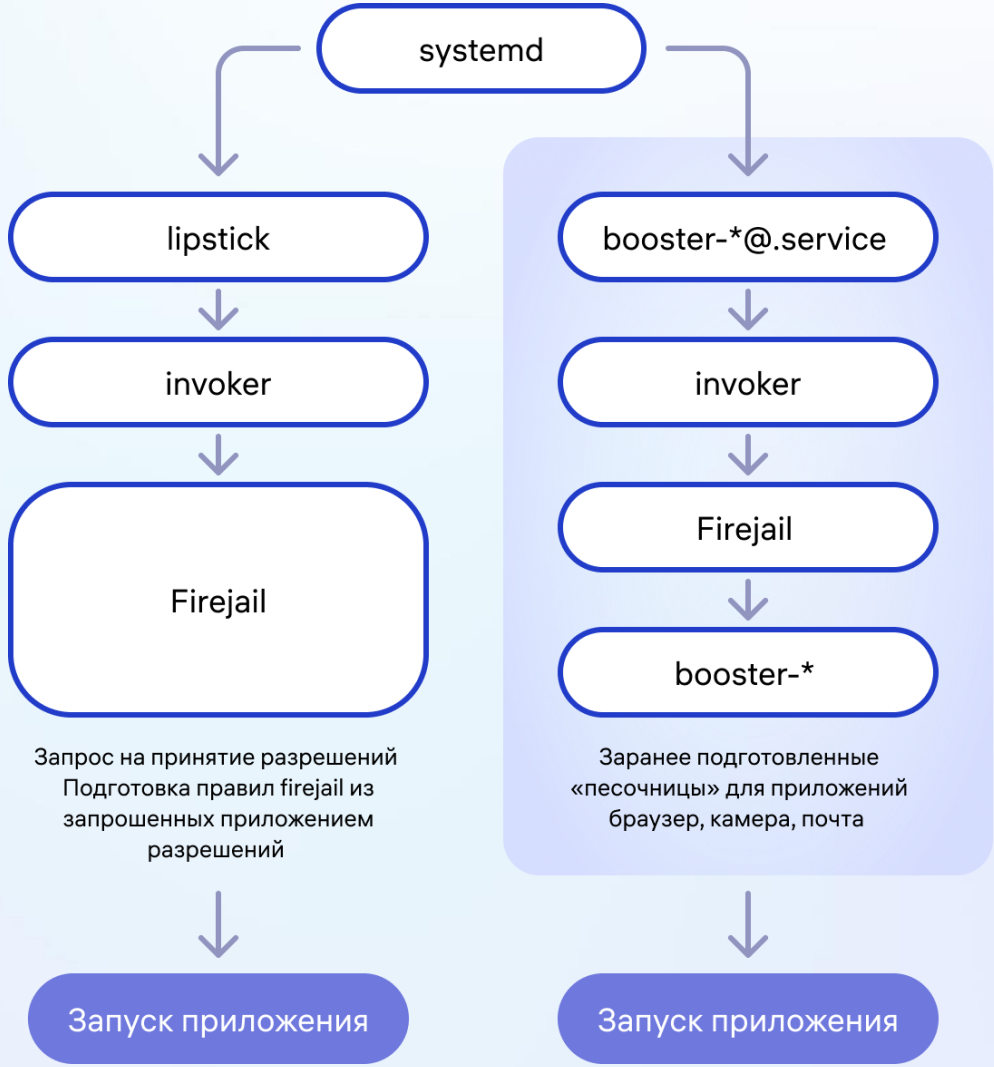
Изоляция приложений в ОС Аврора

Приложения в ОС Аврора 4 запускаются в «песочнице»

- Запуск выполняется через специальный компонент, который является тонкой оберткой над Firejail.

Firejail использует следующие механизмы:

- Linux namespaces – для ограничения доступа к файловой системе
- Seccomp-bpf – для запрета некоторых системных вызовов, например: mount/umount, ptrace, kexec и др.
- xdg-dbus-proxy – фильтрующий прокси для D-Bus

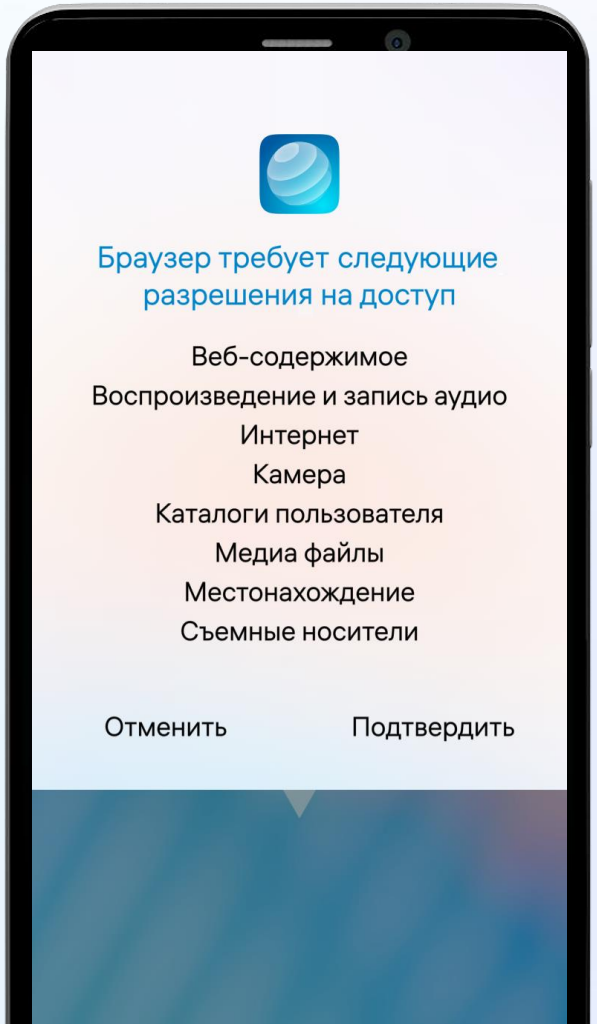


Разрешения приложений (app permissions)

Доступ к системе контролируется с помощью разрешений (permissions)

- Определен набор базовых разрешений (permissions), которые предоставляют доступ к определенным путям, D-Bus интерфейсам, сокетам и бинарным файлам.
- Приложение, написанное для ОС Аврора, должно определить свой профиль: перечислить все необходимые ему разрешения в специальной секции desktop файла.
- Ранее написанные приложения запускаются с профилем по умолчанию – предопределенным набором разрешений, охватывающим основные потребности существующих приложений.

```
Permissions=WebView;Audio;Camera;Location;Internet;UserDirs;Removable
Media;MediaIndexing;ru.cryptopro.nmcades@ru.cryptopro.csp
```



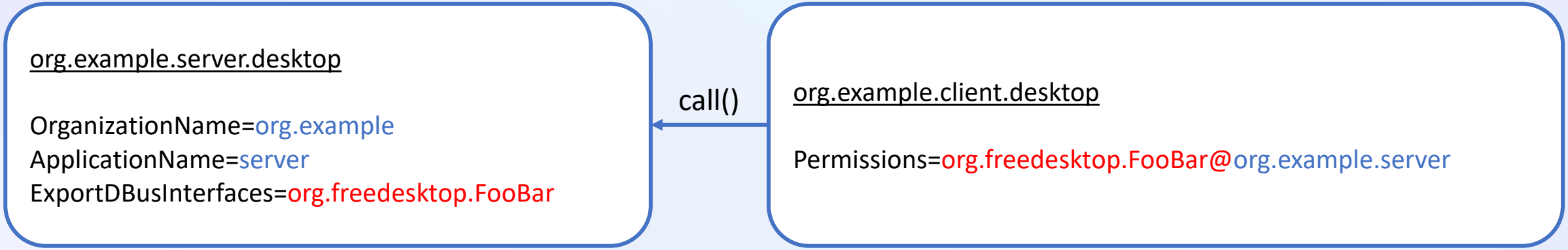
Основные типы IPC между изолированными приложениями

- D-Bus
- Использование дескрипторов, переданных через D-Bus:
  - Pipe
  - Неименованные сокеты
  - ...
- Именованные сокеты

Использование IPC между изолированными приложениями

D-Bus между приложениями:

- Приложения могут экспортировать интерфейсы D-Bus, используя атрибут ExportDBusInterfaces в секции описания разрешений в desktop файле
- Для общения с ним из другого приложения нужно использовать соответствующее разрешение
- Приложение может регистрировать только сервис, соответствующий его имени: например, для приложения org.example.app можно регистрировать только сервис org.example.app
- На имя регистрируемого интерфейса ограничений нет



D-Bus между приложениями: запуск приложения

- Если приложение, к которому обращаются по D-Bus не запущено, для его запуска будет использоваться атрибут ExecDBus
- Оно может обработать опции запуска и, например, не открывать UI при активации по D-Bus

```
org.example.app.desktop:

[Desktop Entry]
# .....
Exec=/usr/bin/org.example.app

# .....
OrganizationName=org.example
ApplicationName=app
ExportDBusInterfaces=org.freedesktop.FooBar
ExecDBus=/usr/bin/org.example.app ---background
```

D-Bus между приложениями: безопасность

Защита от прослушивания через dbus-monitor

- Для анонимного общения между приложениями, можно передать дескриптор на pipe по D-Bus (переданный дескриптор нельзя перехватить dbus-monitor'ом) и общаться дальше через него

```
<interface name="org.example.secret">
  <method name="StartCommunication">
    <arg name="fd" type="h" direction="in"/>
  </method>
</interface>
```

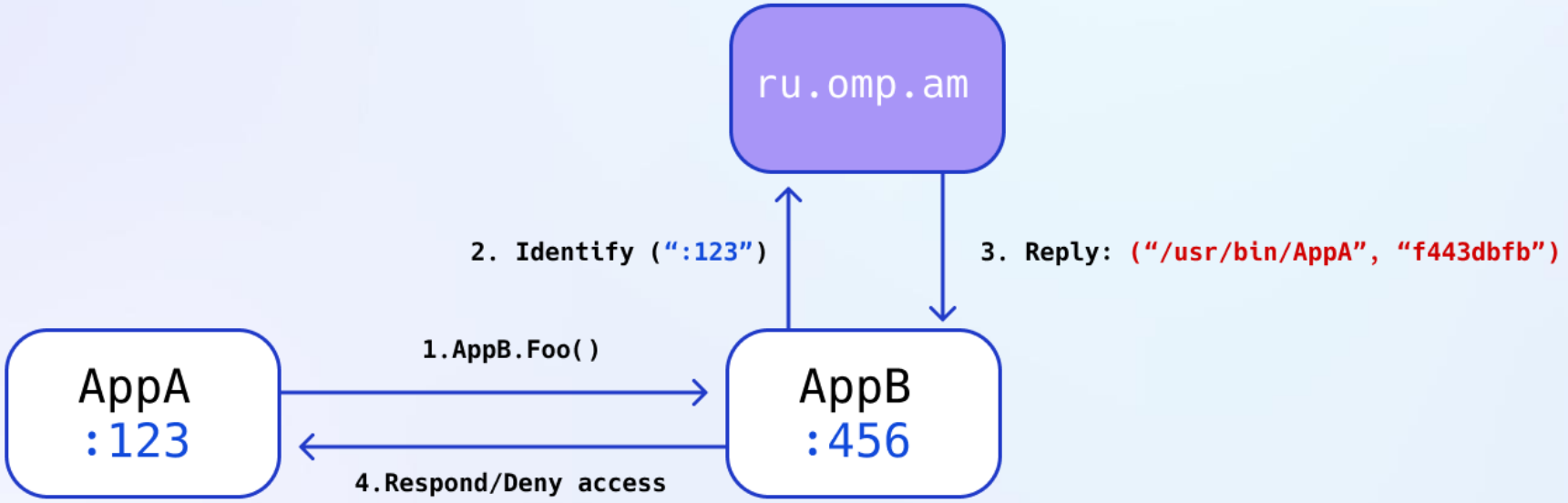
Ограничение списка приложений, которым доступен экспортируемый интерфейс D-Bus

- Можно идентифицировать вызывающее приложение по unique D-Bus имени (начинается с ":") с помощью вызова Identify
  - В ответ будет получен путь до вызывающего приложения и keyid
  - keyid хранится в подписи исполняемого файла (security.ima xattr)

D-Bus между приложениями: безопасность

Ограничение списка приложений, которым доступен экспортируемый интерфейс D-Bus

```
<interface name="ru.omp.am.Identify">
   <method name="IdentifyByBusName">
      <arg name="name" type="s" direction="in"/>      <!-- :123 — unique name -->
      <arg name="exe" type="s" direction="out"/>      <!-- /usr/bin/AppA -->
      <arg name="keyid" type="s" direction="out"/>    <!-- f443dbfb -->
   </method>
</interface>
```



Общие каталоги для динамических данных

- Автоматически создаются при первом запуске приложения
- Доступны только в runtime, в эти директории нельзя помещать файлы при установке приложения
- Варианты ограничения доступа:
  - Доступ только для текущего пользователя либо для всех пользователей системы
  - Доступ только для данного приложения либо для всех приложений (в рамках одного vendor)

Общие каталоги для статических данных

- В эти директории можно помещать файлы только при установке приложения
- В runtime доступны только для чтения

Конфигурационные файлы приложений

- Могут распространяться в виде отдельных rpm-пакетов (путь %{_datadir}/org.example/config в spec файле)
- Для приложения org.example.config путь для установки конфига будет /usr/share/common/org.example/config
- При этом приложение org.example.foobar сможет стучаться во всю директорию /usr/share/common/org.example

Общие каталоги для статических данных

Библиотеки

- Предоставление библиотек для использования другими приложениями
- Установка библиотек доступна только для приложений с **extended** профилем валидации
- /usr/lib/3rdparty/<package_name> - в изолированном окружении доступна только на чтение
- Приложениям **не разрешено** зависеть от несистемных компонентов
- Приложению-провайдеру **рекомендуется** предоставлять статическую библиотеку, скрывающую вызовы dlopen, для упрощения использования предоставляемого API

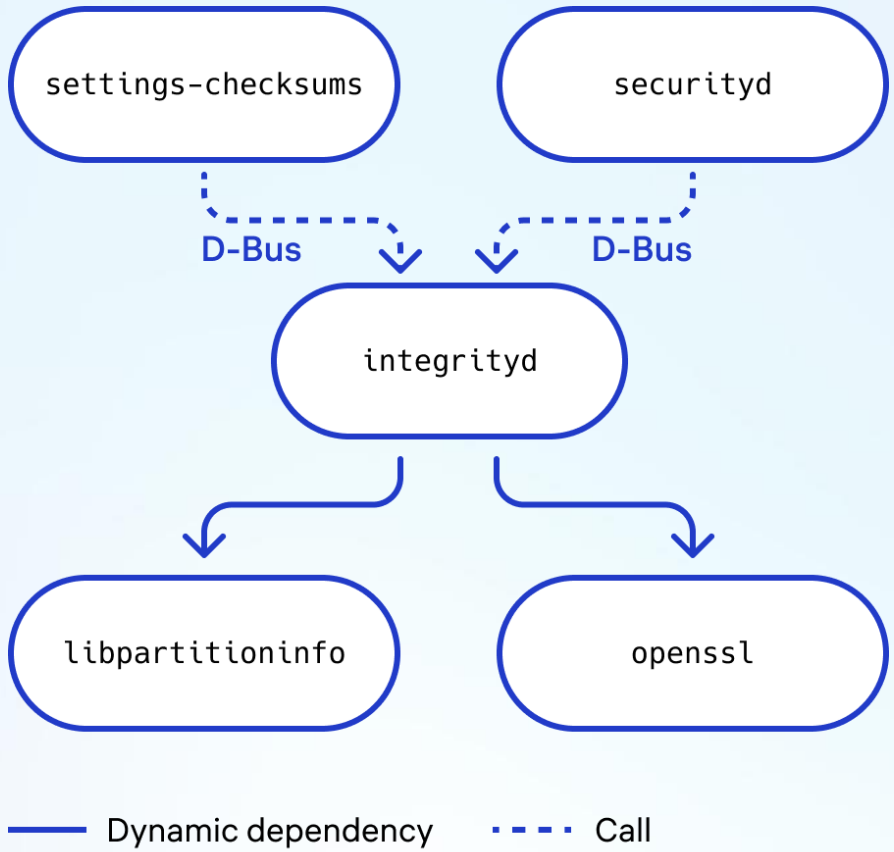
Сервисы контроля целостности

Integrityd – сервис для контроля целостности файлов и разделов

- Умеет проверять подпись файлов и разделов
- Умеет считать контрольные суммы файлов и разделов
- Использует IMA для проверки подписей

Securityd

- Выполняет проверку целостности системы по расписанию, используя Integrityd
- Блокирует устройство при обнаружений несоответствий в системных файлах



Публичное API сервиса integrityd

- Каждое стороннее приложение имеет возможность поставить свои файлы на контроль целостности.
- Для этого при установке пакета необходимо поместить в директорию /etc/integrityd/config.d/ конфигурационный файл (package_name.json) следующего вида:

```
{
    "watch": [
        "file:/path/to/some/regular/package/file",
        "elf:/path/to/package/executable/binary"
    ]
}
```

- Проверка всех указанных файлов происходит автоматически каждые 24 часа и при каждом включении устройства.
- В случае нарушения целостности в журнал попадет событие INTEGRITY_3RDPARTY_FAILED со списком всех файлов, не прошедших проверку.

Выводы

- Защита мобильной операционной системы Аврора является комплексной и адресует различные векторы атаки
- Важнейшим механизмом обеспечения безопасной работы приложений является изоляция и разрешения приложений
- Гибкость и простота настройки изоляции реализуется в ОС Аврора механизмами компонента firejail
- Для возможности безопасного взаимодействия между приложениями в «песочнице» в ОС Аврора добавлены следующие механизмы:
  - Использование D-Bus для вызова интерфейсов других приложений и организации каналов IPC между ними
  - Хранение общих данных для нескольких приложений одного вендора
  - Возможность использования third-party библиотек и конфигурационных пакетов

Вопросы?

omp.ru

auroraos.ru

info@omp.ru