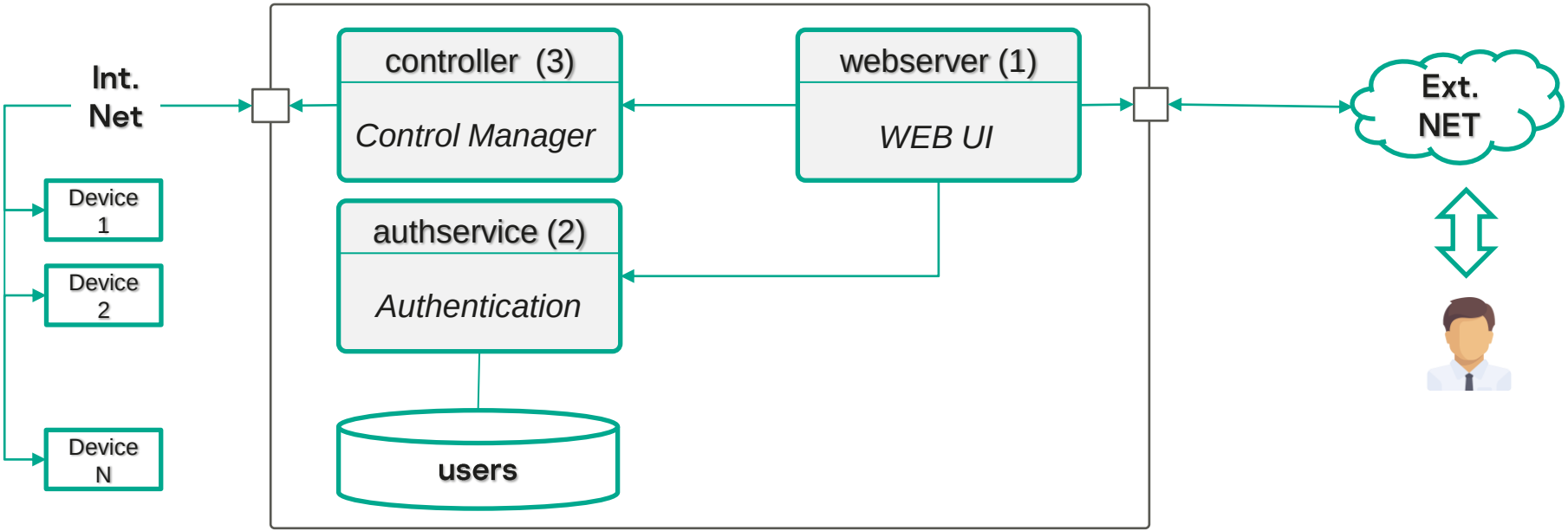
Возможности KasperskyOS и SELinux при создании безопасных решений

Игорь Сорокин
старший архитектор
программного обеспечения

kaspersky

Проектируемое решение



1. Управление через WEB-интерфейс
2. Контроль доступа – аутентификация (user, admin)
3. Обработка команд, предоставление состояния решения

kaspersky

Цели информационной безопасности

1. Предотвращение вредоносной и паразитной активности. Исполнение операций, которые не относятся к реализации функций устройства или обеспечению целей безопасности должно быть предотвращено или остановлено.
2. Авторизованное управление и конфигурирование. Управление и конфигурирование работы решения должно осуществляться владельцем или авторизованным им субъектом. Неавторизованное управление и конфигурирование должны быть исключены.
3. Разграничение доступа к функциям решения должно осуществляться с учетом ролей пользователей (admin, user)

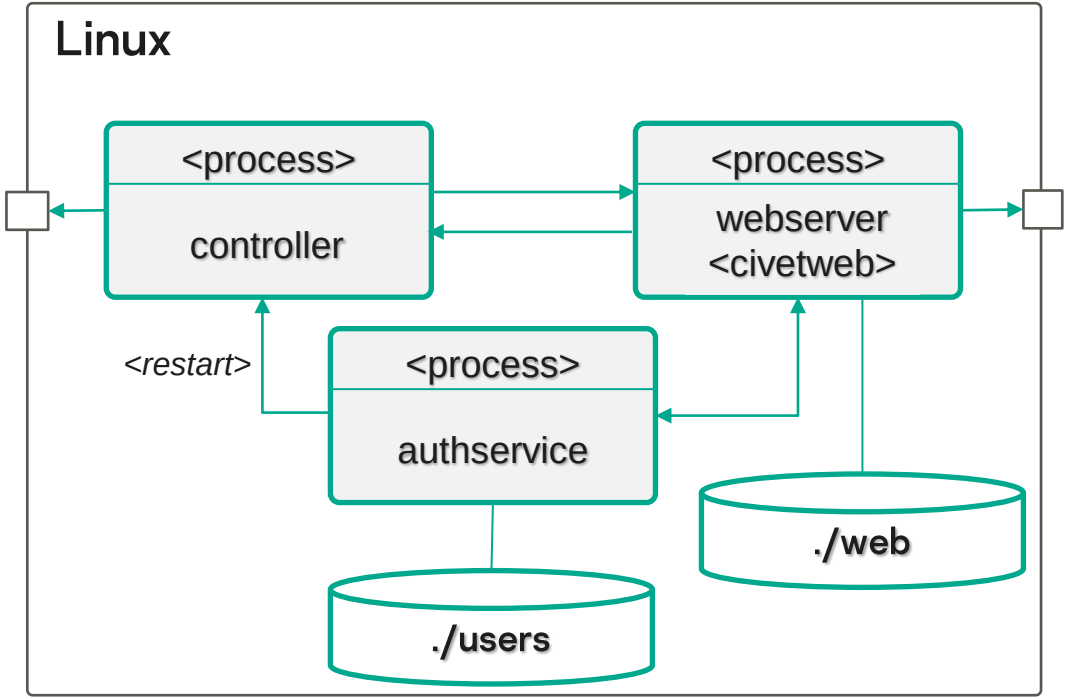
Условия функционирования

1. Ограничение физического доступа к устройству
2. Отсутствие непосредственного взаимодействия внешней и внутренней сети

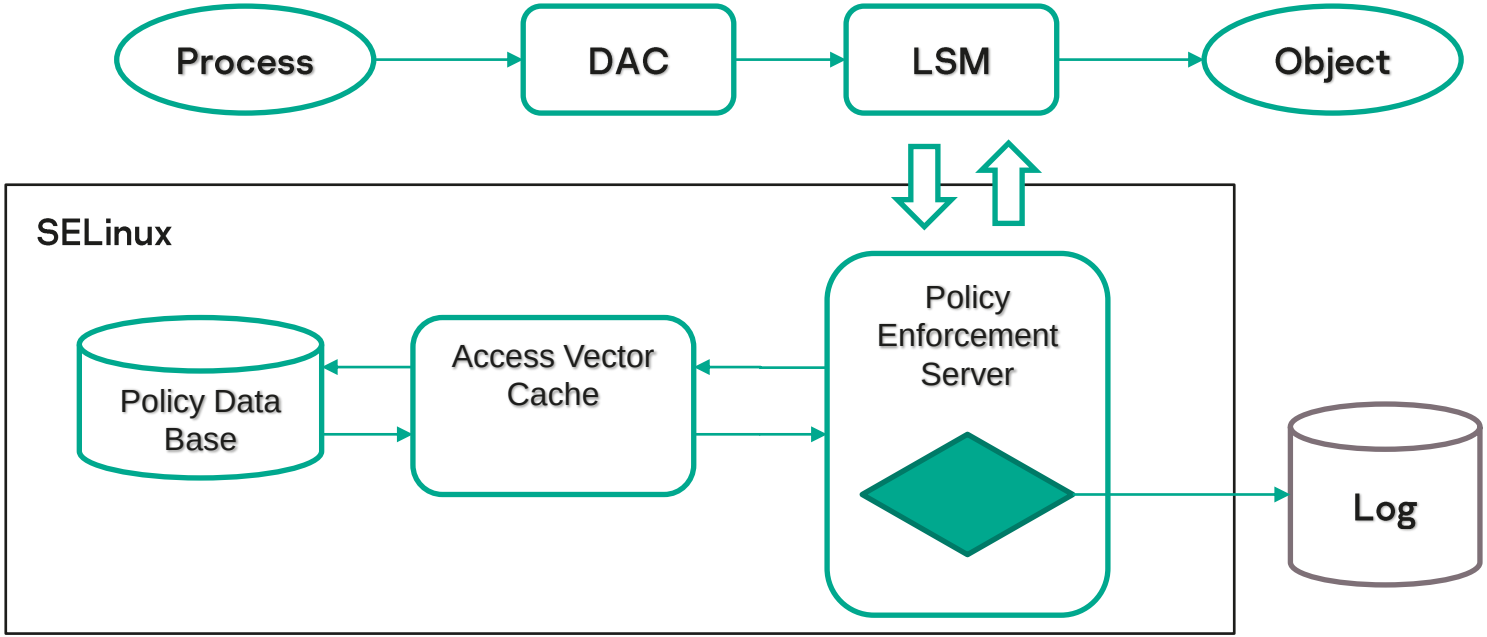
SELinux: Архитектура решения

Linux:
- монолитная ОС
- DAC (discretionary access control).

SELinux:
- расширение ядра для предоставления механизмов контроля доступа;
- Поддерживается MAC (mandatory access control).



SELinux: Принцип работы

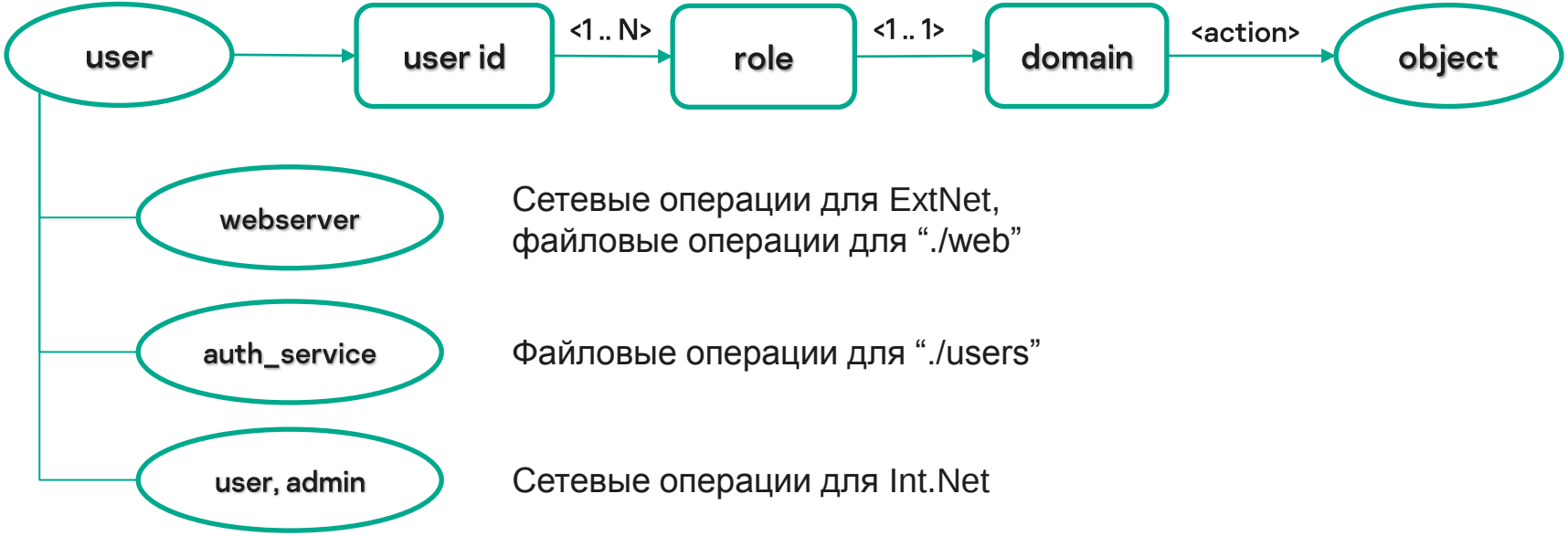


1. Перехват обращения к объектам ОС с помощью встроенного фреймворка LSM (224 hooks of version 5.4)
2. Контроль доступа с помощью политик безопасности

kaspersky

SELinux: Политики безопасности

- Выберем Type Enforcement (TE) политику, для организации ролевой модели управления доступом



SELinux: Разработка политики безопасности

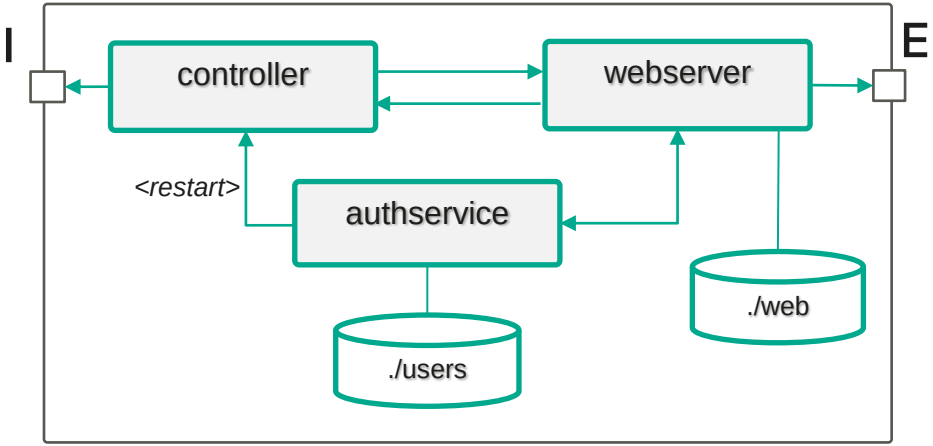
Выполняется разработчиком

1. Определение базовых TE-типов (.te - файлы);
2. Определение правил перехода (.if - файлы):
   - на основе .if правил (*«/usr/share/selinux/devel/include/»* - 436 правил);
   - используя макросы атрибутов и стандартных разрешений;
3. Определение файла контекстов вида (.fc - файлы);
4. Определение ролей пользователей;
5. Компиляция и установка правил;
6. Тестирование:
   - Накопление логов;
   - Анализ логов и уточнение политик (**audit2allow** - generate SELinux policy allow rules from logs of denied operations);
7. Переход на шаг 4;
8. Изменение реализации приводит к необходимости валидации и изменения политики.

SELinux: Что учитывать при разработке решения

Критерии выбора правил политики

1. Работоспособность решения
2. Принцип минимизации назначенных разрешений



authservice
- Логирование
- Взаимодействие authservice-webserver
- Доступ к данным “./users”

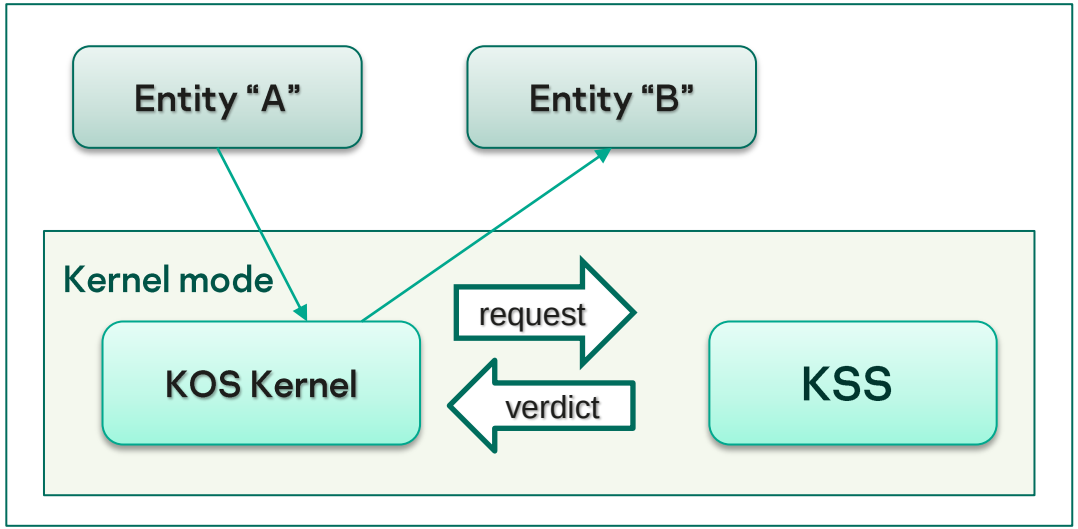
controller
- Логирование
- Взаимодействие controller-webserver
- Сетевое взаимодействие с (I)nternal Network

webserver
- Доступ к данным “./web”
- Логирование
- Сетевое взаимодействие с (E)xternal Network
- Взаимодействие webserver-authservice
- Взаимодействие webserver-controller

kaspersky

KasperskyOS: Основы архитектры

- Микроядерная операционная система, реализующая концепцию монитора обращений.
- Решение на базе KasperskyOS состоит из изолированных *сущностей* (аналог процесса).
- Сущности взаимодействуют друг с другом и с ядром посредством *интерфейсов*.
- Интерфейсы описываются статически с помощью декларативного языка.
- Взаимодействие сущностей основано на IPC-канале.
- Контроль взаимодействия сущностей обеспечивается – Kaspersky Security System (KSS).
- KSS анализирует каждый отправляемый запрос и ответ и на основе заданной политики безопасности выносит решение, разрешить или запретить его доставку.



kaspersky

KasperskyOS: Классы политик безопасности

Набор семантически связанных политик, описывающих определенную модель безопасности.

- Base – базовый класс, реализующий политики grant, assert и deny;
- MIC – Mandatory Integrity Control
- Regex – класс, реализующий валидацию текстовых данных по статическим шаблонам (регулярным выражениям),
- HashSet – класс, реализующий механизмы работы со структурами данных типа "hash-таблица",
- StaticMap – класс, реализующий механизмы работы со структурами данных типа "словарь со статическими ключами",
- Flow – реализация конечного автомата,
- RBAC – реализация модели управления доступом на основе ролей (Role Based Access Control).

Разработка политики безопасности для KasperskyOS

Software Developers

- Архитектура решения
- Выделение доменов безопасности
- Определение IPC-интерфейсов

- Реализация,
  сборка, тестирование, ...

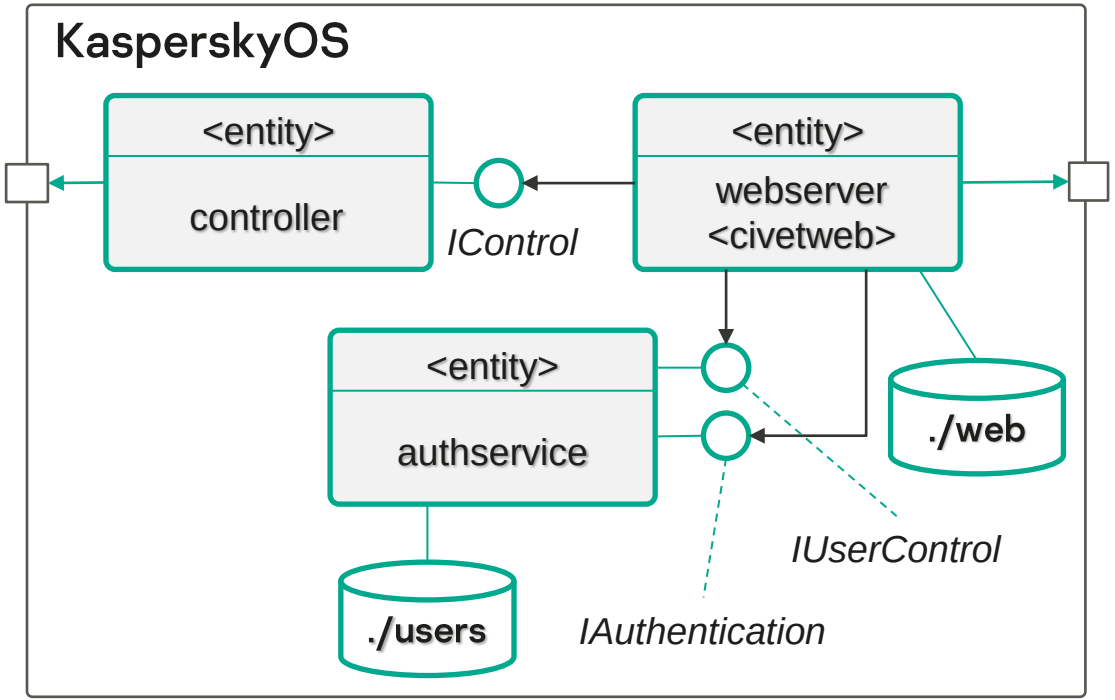
Security Expert

- Анализ архитектуры,
  топологии доменов
- Анализ IPC-интерфейсов

- Разработка политик безопасности
- Описание политик безопасности

Безопасное решение

KasperskyOS: Архитектура решения



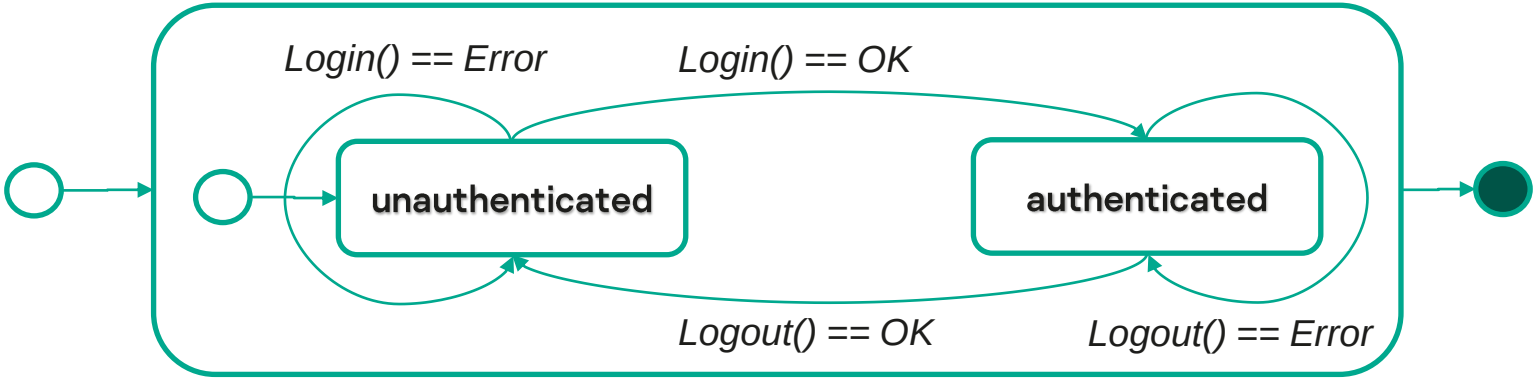
IControl – обработка команд управления
- *Execute(command)*

IUserControl – управление списком пользователей
- *Create(user, password, role)*
- *Read(<users>)*
- *Update(user, password, role)*
- *Delete(user)*

IAuthentication – аутентификация пользователя
Login(user, password)
Logout(user)

KasperskyOS: Политика безопасности решения

- Base – позволяет определить доступные интерфейсы в зависимости от состояния системы
- Flow – позволяет определить состояния системы



IAuthenticated::Login = grant
IAuthenticated::Login = deny
IControl:: = deny*
IUserControl:: = deny*

IAuthenticated::Login = deny
IAuthenticated::Login = grant
IControl:: = grant*
IUserControl:: = <….>*

kaspersky

Выводы

- Возможности SELinux и KasperskyOS позволяют создавать безопасные решения;
- При разработке решения следует учитывать:

Критерий	SELinux	KasperskyOS
Архитектура ОС	Монолитная OS	Микроядерная OS
Политики безопасности	Базируется на низкоуровневых операциях доступных через LSM. Требует участия разработчиков.	Базируется на определении высокоуровневых IPC-интерфейсов, описывающих логику операции. Может быть разработана «офицером безопасности».
Поддержка политик в актуальном состоянии	Требует анализа информации логирования LSM. Необходима тщательная верификация после изменения реализации.	Возможна частичная автоматическая верификация при изменении архитектуры решения компилятором политик.

Спасибо за внимание!

kaspersky