Алекса нам поможет, но так ли просто с ней общаться?

Безопасен ли ваш дом?

Безопасная сервис-ориентированная архитектура в Embedded Linux для IoT

Wire Snark, 2017

Об авторе

- Разработчик демонов и сервисов в GNU/Linux и Android
- Исследователь безопасности Java-приложений (Андроид, серверные приложения)
- Основатель Paranoid Security
- Со-основатель
  DEF CON Нижний Новгород defcon-nn.ru

wsnark@tuta.io @wsnark @wiresnark

GnuPG fingerprint
4497 F125 194A 47AD 0E1B
05A3 E12E D410 7595 4ED2

План доклада

- Что такое Алекса
- ASK: Alexa Skills Kit
- AVS: Alexa Voice Services
- LWA: Login With Amazon
- SDL: Security Development Lifecycle
- Модель угроз
- Принципы безопасного проектирования
- Применение к построению системы голосового управления
- Подводные камни
- Заключение

Что такое Алекса?

- “Умный” голосовой помощник от Amazon
- Языки: английский и немецкий
- Предоставляет информацию о погоде, пробках, новости
- Отвечает на вопросы, предоставляет информацию, в т.ч. из Википедии
- Проигрывает музыку, радио
- Позволяет задавать таймеры, будильники, списки дел и покупок

Что такое Алекса?

- Амазон Эхо
- Приложения для Android и iOS
- Сторонние голосовые команды - Alexa Skills Kit, Alexa Smart Home
- Сторонние клиенты - Alexa Voice Services

Alexa Skills Kit

Голосовой запрос

TTS

AVS

Структурированный запрос

Ответ

Обработчик

Смартфон

Графическое представление

Alexa Skills Kit

Голосовой запрос

AVS

Структурированный запрос

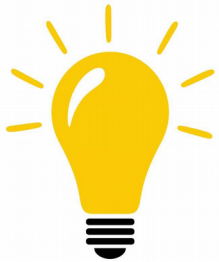
AWS Lambda

Java
Node.js
Python

Адаптер

Управление домом

Контроллер



Alexa Voice Services (AVS)

- Сервис с сохранением состояния клиента
- Транспорт – только HTTP/2 с ALPN (Application-Layer Protocol Negotiation – расширение TLS)
- Взаимодействие через события (event на клиенте) и директивы (directive, от сервиса)
- Авторизация через Login With Amazon OAuth 2.0

Протокол HTTP/2

- Бинарный протокол, развитие Google SPDY
- Определен в RFC 7540, май 2015
- Сложный
- Поддержка в HTTP-клиентах не очень хорошая
- Поддержка нескольких потоков внутри одного TCP-соединения
- Server Push

Транспорт в AVS

- Каждый запрос на распознавание обрабатывается в отдельном HTTP/2-потоке (stream)
- Директивы, инициированные AVS, пересылаются в downchannel stream – потоке, который всегда должен быть открыт
- Начальное состояние клиента и любое изменение состояния необходимо отправлять на сервер
- Коммуникация в составных (multipart) HTTP/2-сообщениях
- Запросы в формате JSON, звук – в бинарном формате, разбитом на части (отправка голосовых данных в реальном времени)

Интерфейсы AVS

- SpeechRecognizer – распознавание запроса
- SpeechSynthesizer – проигрывание ответа
- Alerts – напоминания, таймеры и т.п.
- AudioPlayer – проигрывание музыки, радио
- Speaker – управление громкостью
- PlaybackController
- Settings
- System

Нетривиальные особенности AVS

- Директива StopCapture: поддержка “remote speech endpointing”, т.е. детектирование окончания речи самим AVS
- Директива ExpectSpeech: уточняющий вопрос от AVS пользователю, диалог
- Директивы Play, AdjustVolume, SetMute: от внешнего (по отношению к клиенту) управления – например, из мобильного приложения.
- Директива Play: запрос на проигрывание потока с произвольного URL

Авторизация через Oauth 2.0

- https://oauth.net RFC 6749, 6750, 6819 + куча расширений. Очень большой и сложный.
- Легко реализовать небезопасным образом: https://sakurity.com/oauth

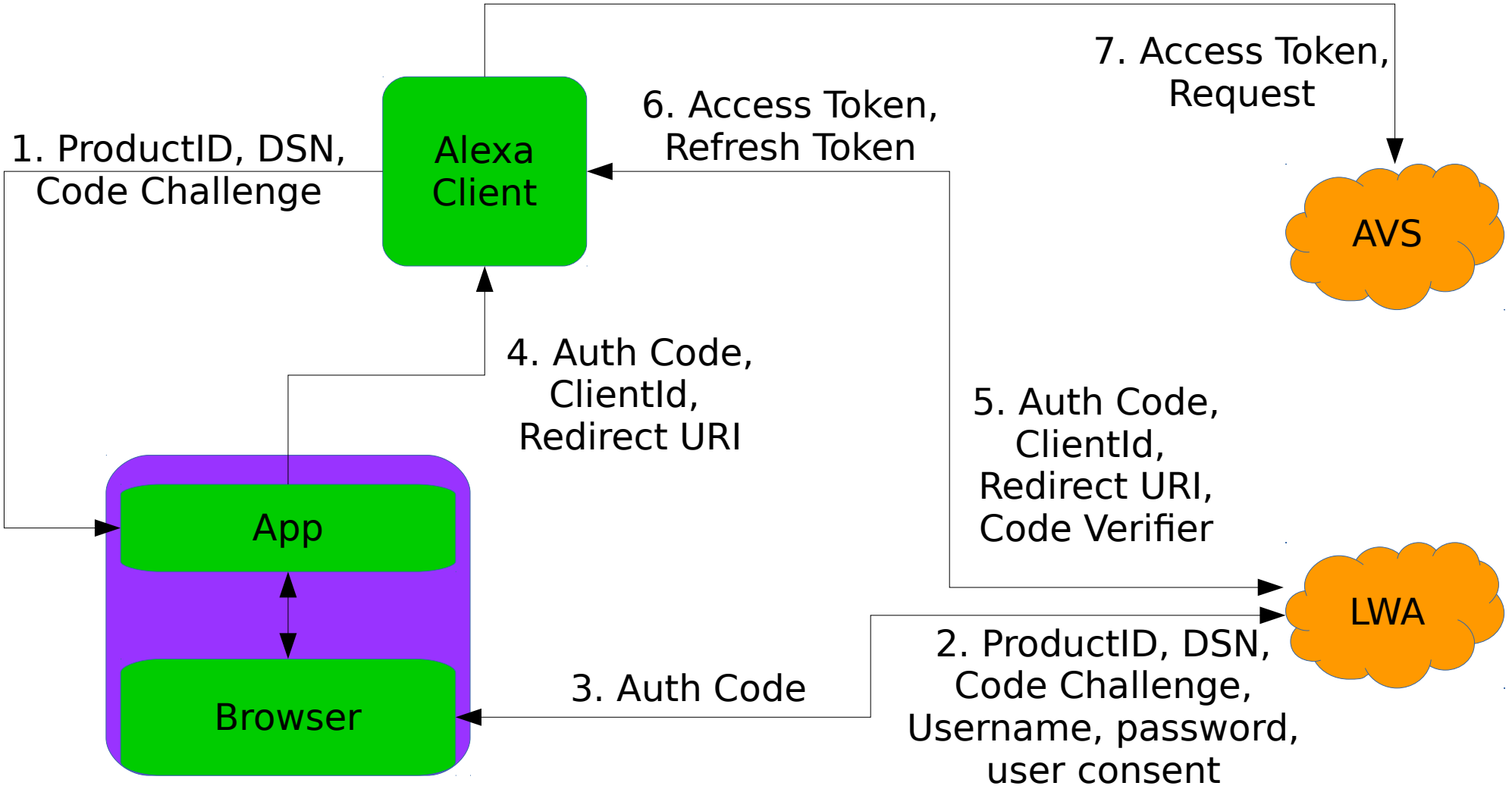
Login With Amazon

- Авторизация внутри мобильного приложения
- Авторизация через веб-сервис
- Авторизация через мобильное приложение

Login With Amazon - понятия

- Access Token, Refresh Token
- ClientId, Redirect URI
- ProductId, DSN (Device Serial Number)
- Auth Code, Code Challenge and Verifier (алгоритм - Proof Key for Code Exchange by OAuth Public Clients)

Login With Amazon - App



Вопросы?

“Интернет вещей”

- Стартапы: игры в “умный дом” без обеспечения важнейших свойств
  - Устройства перестают работать при проблемах с доступом к сети
  - Проблема обновления прошивки
  - Отсутствие безопасности и приватности
- Корпорации: умный дом как способ владения пользователями
  - Все данные в открытом виде идут на корпоративные сервера
- Twitter: @internetofshit

Голосовое управление умным домом: требования

- Работа онлайн - через AVS (с потенциальной возможностью добавить поддержку других подобных сервисов)
- Ограниченная работа оффлайн – локальное распознавание речи
- Безопасность

Security Development Lifecycle (SDL)

1) Обучение
2) Определение требований
   - Требования безопасности и приватности
3) Проектирование
   - Модель угроз, анализ поверхности атаки, требования безопасного проектирования
4) Реализация
5) Верификация
6) Выпуск
7) Реагирование на ициденты

Требования безопасности и приватности

- Конфиденциальность
- Целостность
- Доступность
- Аутентификация
- Авторизация
- Неотказуемость

Модель угроз

- Определяем ценные данные и функции системы (assets)
  - Доступ к микрофону и динамикам
  - Доступ к управлению устройствами умного дома
  - Токены
  - Настройки, системные параметры
  - Логи
  - Мета-данные (факт и длительность передачи данных, использования ресурса или устройства и т.д.)
  - Операционная система

Модель угроз

- Определяем точки входа и выхода, поверхность атаки
  - Интерфейс к AVS (включая сторонние аудио-сервисы)
  - Интерфейс к LWA
  - Интерфейс к мобильному приложению
  - Интерфейс к управлению умным домом
  - Голосовой вход (подсистема записи и распознавания), звуковой выход (подсистема разбора, декодирования и проигрывания)
- Определяем потенциальных атакующих (threat agents)
  - Вредоносное ПО (ненацеленное)
  - Корпораций (конкуренты, инвесторы)
  - Преступники (воры, вымогатели, операторы ботнетов и продавцы персональных данных)
  - Профессионалы (спецслужбы, частные детективы, коллекторы)
  - Индивидуалы (соседи, знакомые, сотрудники, скрипт-кидс)

Модель угроз

- Исходя из предполагаемых возможностей атакующих, выбираем допущения и границы доверия (assumptions, trust boundaries). Клиенту не следует доверять
  - сторонним серверам с контентом
  - серверам AVS и LWA
  - мобильному приложению
  - голосовому вводу
  - устройствам из умного дома (включая управляющие)

Модель угроз

- Классифицируем все вероятные угрозы по модели STRIDE
  - Spoofing: имитация, подделка чего-либо
  - Tampering: вмешательство, изменение
  - Repudiation: отрицание операции
  - Information disclosure: раскрытие информации
  - Denial of service: отказ в обслуживании
  - Elevation of privilege: повышение привилегий

Принципы безопасного проектирования

- Минимизация поверхности атаки
- Безопасность по умолчанию
- Минимальные привилегии
- Разделение обязанностей
- Эшелонированная защита (defence in depth)
- Безопасная обработка ошибок
- KSS – Keep Security Simple

Изоляция недоверенного кода

- Минимизация привилегий + Разделение обязанностей + Минимизация поверхности атаки = изоляция в виде обработчиков с одним входом и выходом

Компонентный дизайн

- auth-manager
- alexa-client
- iot-controller

Платформа

- Yocto Linux / OpenEmbedded
- 500MHz NXP/Freescale i.MX6UL
- 512Gb RAM
- 4Gb eMMC

Локальное распознавание голоса

- Коммерческие проекты, например Sensory THF – работают практически “из коробки”, поддержка, тренировка акустических моделей
- Открытые – CMU Sphinx (poketsphinx)

Выбор средств разработки

- Языки: C, C++, Python, Node.js, Go, Rust, Java
- Межпроцессное взаимодействие: D-Bus, Unix Pipes, Unix Sockets, MQTT
- Системные библиотеки: Qt, GLib2
- Медиа-фреймворк: GStreamer, VLC

Межпроцессное взаимодействие через D-Bus

- Адресация и роутинг
- Активация сервисов
- Граница безопасности (можно задать политики безопасности)
- Возможна передача файловых дескрипторов
- Бинарный протокол
- Требует биндингов, нетривиально в использовании. Для C – GDbus, C++ - QtDbus

Критерии выбора языка

- Обязательные
  - HTTP/2 client with ALPN
  - D-Bus bindings
- Желательные
  - Популярность
  - История использования во встраиваемых системах
  - Производительность
  - Быстрый старт
- Приятные
  - Легкий в использовании и изучении
  - Низкое потребление ресурсов

Результаты анализа

- C: небезопасный; дорогостоящая разработка
- C++: очень сложный, также проблемы с безопасностью и дорогостоящая разработка
- Rust: нет готового HTTP/2-клиента
- Python: HTTP/2-клиент в альфа-версии, низкая производительность
- Node.js: долгий старт и низкая производительность
- JVM-based: нет биндингов D-Bus, долгий старт
- **Go** – лучший кандидат под данные критерии!

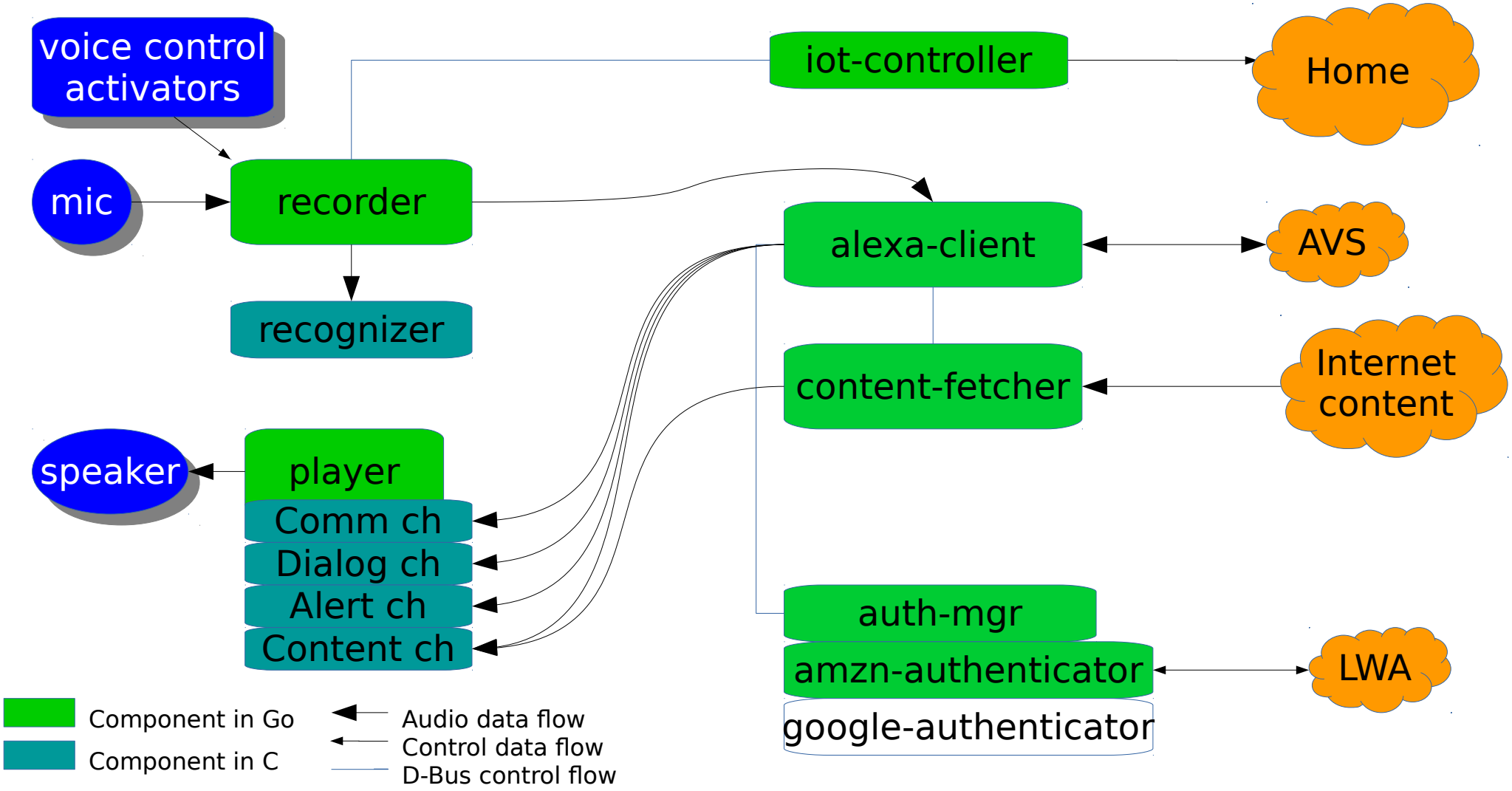
Медиа-фреймворк

- VLC
  - тяжело собрать под Yocto Linux: требует JVM с AWT, OpenJDK для Yocto без него. Пришлось использовать Oracle JDK
  - Непопулярен как фреймворк – скорее плеер
- GStreamer
  - в системе “из коробки”, поддерживается SoC-вендором
  - стандарт де-факто
  - очень гибкий для PoC: gst-launch-1.0 позволяет делать почти всё!
  - сложный API, проблемы с биндингами (Glib2)
  - недоверенный C-код

Подводные камни AVS

- Неспецифицированные области
- Поведение не по спецификации
  - Нужна качественная обработка ошибок на стороне клиента
- Проблемы Golang net/http – слишком высокоуровневый API
- Сложности со снижением задержек

Компонентный дизайн



Компонент alexa-client

```
interface com.lightpad.AlexaClient {
  // Recognize initiates dialog between AVS and the user
  // dialogToken is a unique and secure (not forgeable)
  // token representing this dialog - to reference it later.
  // voicesrc is file descriptor pointing to voice data
  // source (normally a pipe)

func Recognize(dialogToken string, voicesrc fd) void

}
```

Атаки на alexa-client

- Из сети:
  - subverted AVS
  - subverted TLS (security: certificate pinning)
  - malicious audio stream (security: separate content-fetcher process for working with non-AVS audio streams).
- Из локальных процессов:
  - subverted recorder
  - subverted player (highest risk, as it decodes untrusted audio streams)
  - subverted authmgr
  - Неавторизованные процессы вызывают alex-client API (security: D-Bus policy that allows calling alexa-client API only by trusted users)
  - Другие процессы захватывают D-Bus-имя alexa-client (security: D-Bus policy that prevents this)
- Из локальных файлов:
  - Конфигурационные файлы

Поверхность атаки в самом alexa-client

- Соединение с AVS
- Парсер ответов AVS
- Компоненты Go Runtime (TLS, HTTP, D-Bus)
- Нет границы безопасности между AVS и alexa-client, так что компрометация AVS ~ компрометации клиента

Последствия от захвата alexa-client

- Раскрытие пользовательских запросов к AVS, предоставление произвольных ответов, контроль громкости
  - Нет защиты
- Получение доступа к пользовательскому микрофону через директиву RequestSpeech или бесконечная отправка данных (неотсылка директивы StopCapture)
  - DialogToken
  - Таймаут

Заключение

- В IoT нужны инженерные подходы
- Модель угроз помогает построить безопасную архитектуру
- Используйте безопасные языки Go, Rust, Python в Embedded!

Вопросы?

Ссылки

- OWASP https://www.owasp.org
  - Top10, Top10 Mobile, Cheat Sheets, Code Review Guide, Testing Guide, Secure by Design и другие
- FindSecurityBugs: https://find-sec-bugs.github.io/
- Certificate Pinning (GPLv3): https://github.com/moxie0/AndroidPinning https://moxie.org/blog/authenticity-is-broken-in-ssl-but-your-app-ha/
- OAuth & OpenID Connect https://oauth.net/2/ https://openid.net/connect/

Ссылки

- http://cr.yp.to/qmail/qmailsec-20071101.pdf
  DJB o qmail, trusted code base minimization vs minimal privilege