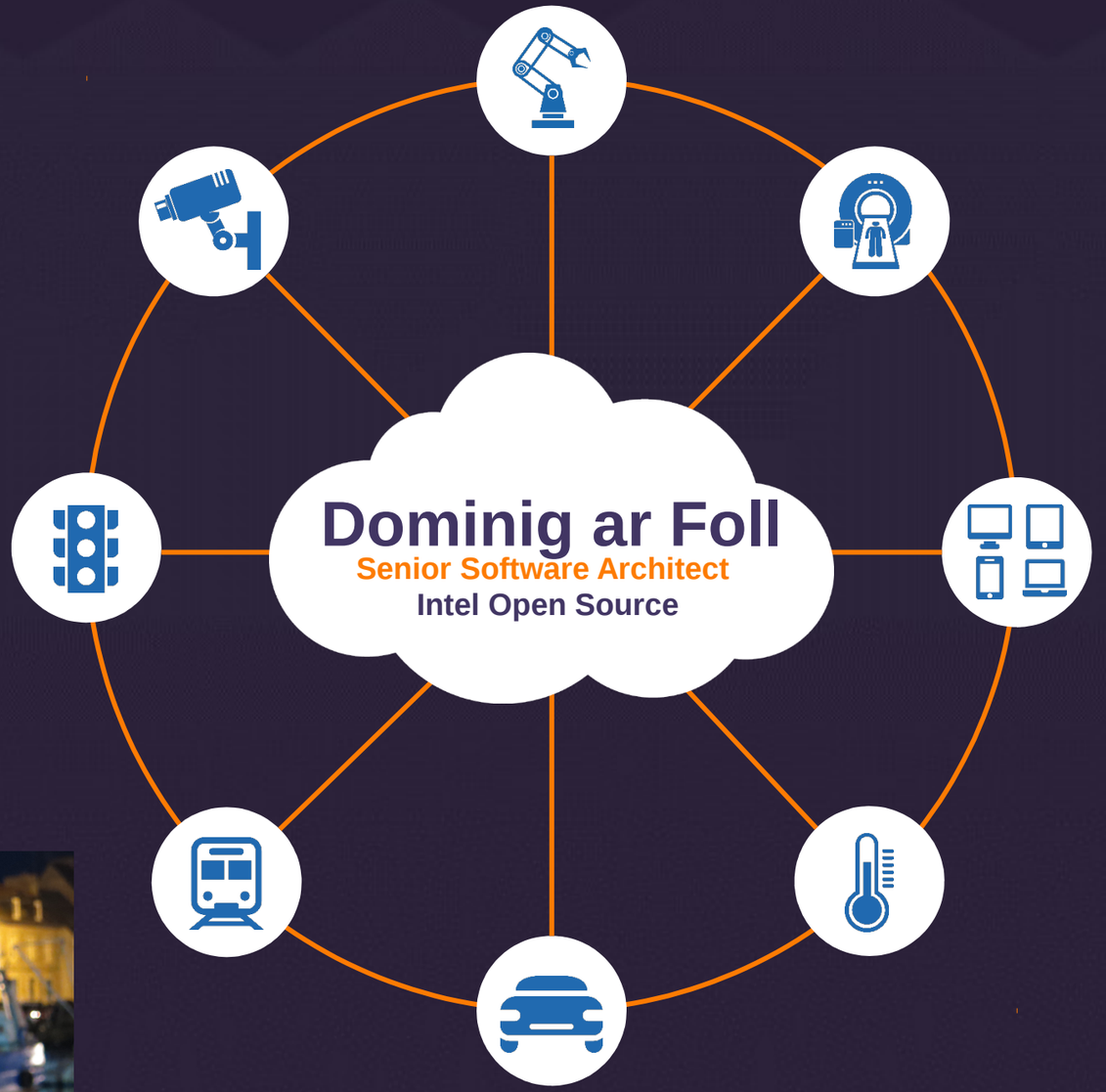
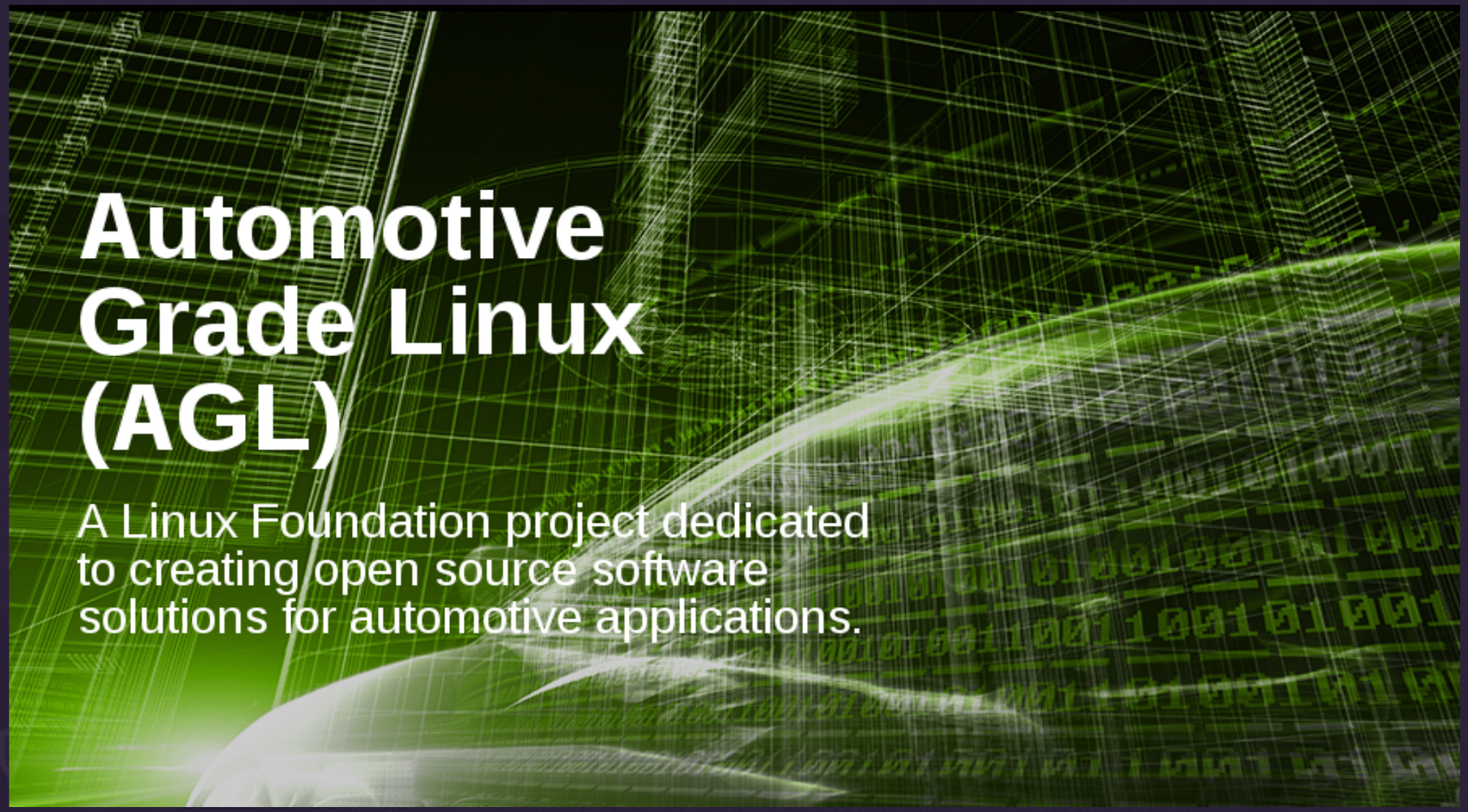


Fosdem 2017, Brussel, Be dominig.arfoll@fridu.net

A harden Embedded Linux



Applicable to any Industrial IoT Linux

Platinum

DENSO

mazda

Panasonic

RENESAS

SUZUKI

TOYOTA

Gold

HONDA
The Power of Dreams

NTT DATA
NTT DATA MSE Corporation

Silver

AISIN AW CO.,LTD.

Continental

Mercedes-Benz

GRIMNES

FUJITSU TEN

irdeto

MITSUBISHI ELECTRIC

NISSAN

Pioneer

QUALCOMM

WIND


Advanced Driver Information Technology
Advanced Telematic SYSTEMS
美しい電子部品を究めます
ALPS
ARM
audiokinetic
AutoIO
bright box
中国移动 China Mobile
cinemo
Codethink
COLLABORA
Elektrobit
ENEA
ETRI
Electronics and Telecommunications Research Institute
Eureka, Inc.
Ford
FORGEROCK
FUJITSU
GlobalLogic
Leaders in Software R&D Services
HARMAN
HI CORP.
HITACHI
Inspire the Next
HYUNDAI MOBIS
igalia
intel
JAGUAR
LAND ROVER
JVCKENWOOD
creates excitement & peace of mind
Konsulko Group

Top 25 Git Committers in 2016

Commits	Name	Company
533	Jose Bollo	IoT.BZH
166	NuoHan Qiao	Fujitsu Ten
146	Jan-Simon Moeller	Linux Foundation
102	Stephane Desneux	IoT.BZH
92	Jens Bocklage	Mentor Graphics
86	Tasuku Suzuki	Qt Company
85	Manuel Bachmann	IoT.BZH
70	Yannick Gicquel	IoT.BZH
64	Ran Cao	Fujitsu Ten
57	Tadao Tanikawa	Panasonic
55	Fulup Ar Foll	IoT.BZH
42	Leon Anavi	Konsulko

Commits	Name	Company
40	Anton Gerasimov	Advanced Telematics
35	Yanhua GU	Fujitsu Ten
22	Christian Gromm	Microchip
21	Ronan	IoT.BZH
20	SriMaldia	Alps
18	Naoto Yamaguchi	AisinAW
15	Karthik Ramanan	TI
13	Scott Murray	Konsulko
11	Kotaro Hashimoto	Mitsubishi Electric
9	Matt Porter	Konsulko
8	Dominig Ar Foll	Intel
8	Yuta Doi	Witz
8	Jian Zhang	Fujitsu Ten

1791 Total Commits
45 Committers
24 Companies

- 01 Jan 2016 – 31 Dec 2016
- Commits to master

LINUX FOUNDATION

AUTOMOTIVE GRADE LINUX

A Linux for Automotive ?

- **Embedded Yocto built**
- Strong interaction with Sensors
- Non Desktop UI
- Dedicated Entry buttons
- MultipleScreens enabled

- **Managed device**
- Any fault will be blamed on system provider
- Applications are gated by system provider
- Long life support
- No admin system to rely on
- ...

From Auto to Industry

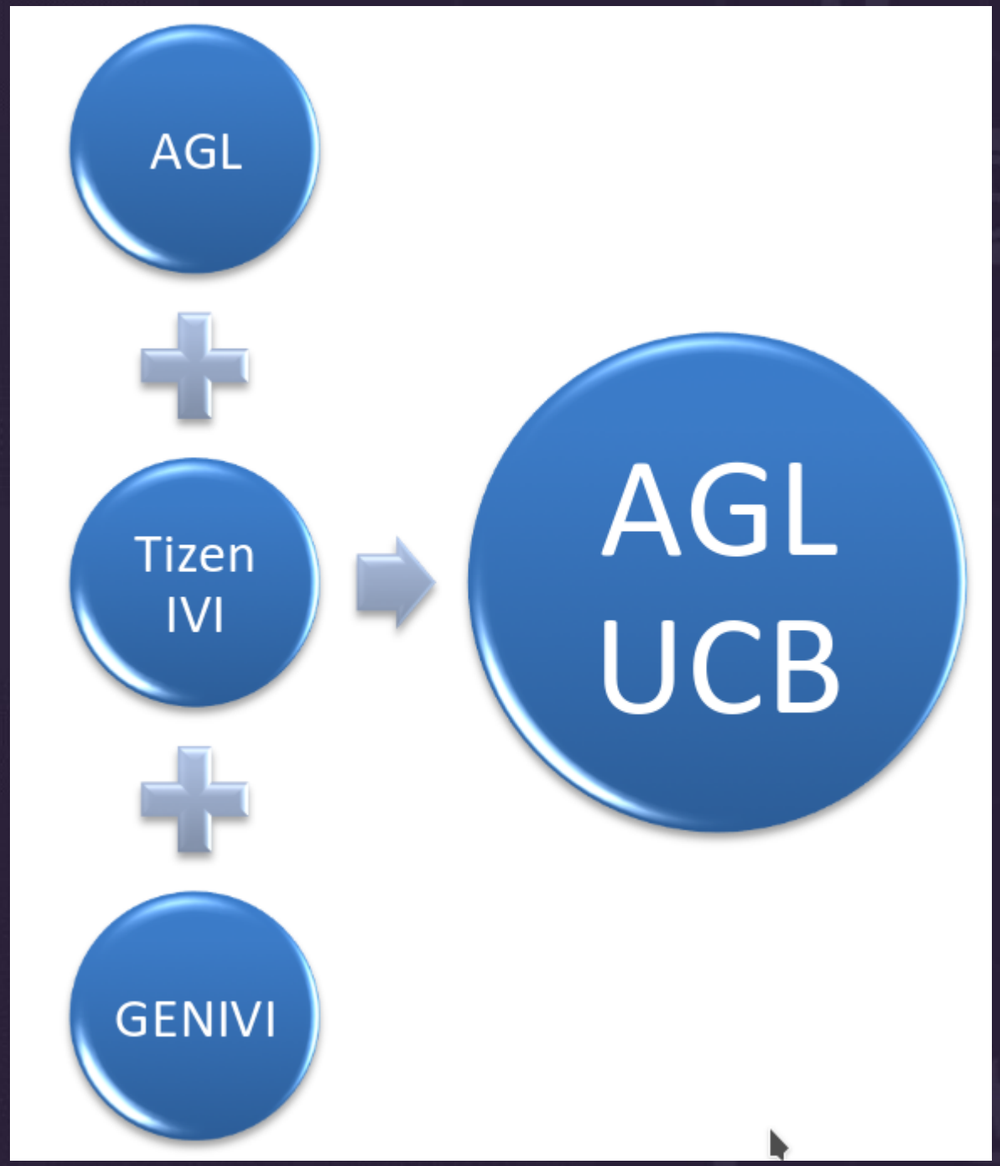
- **Features**
- Speed, position, sensors
- Dedicated UI
- Dedicated Entry buttons
- Multimedia features
- Emergency phone service
- Remote Diagnostic

- **Implementation**
- Embedded Linux with dedicated UI
- Connectivity
- 100% remote support operation
- Very reliable

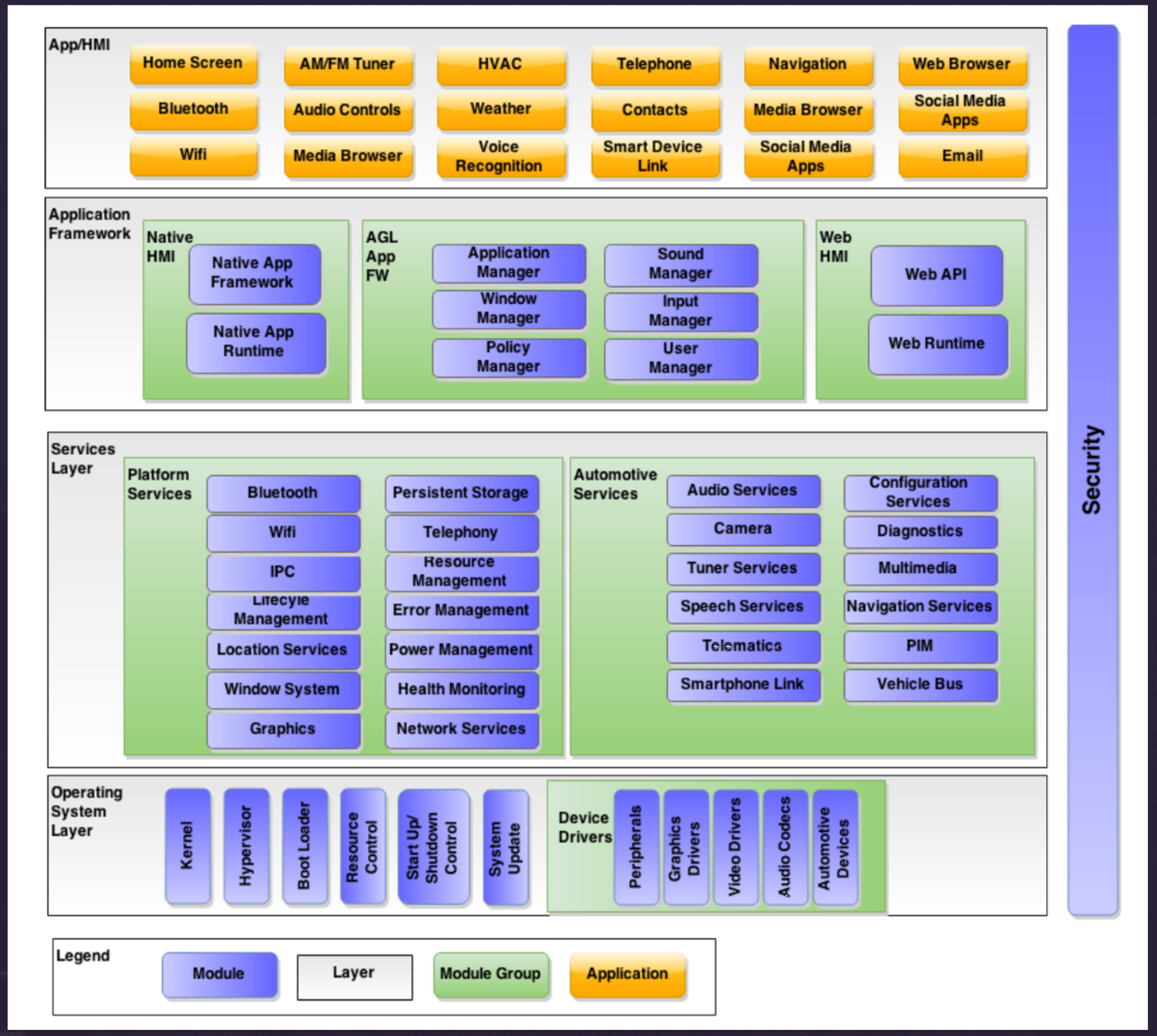
What is AGL (Jan 17)

- **Focus on the core OS**
- Yocto 2.2
- Linux 4.4 or 4.8
- Security model from Tizen
- Standard Layer for BSP
- Source sync via repo tool
- Ready made Docker SDK

- **App and Middleware**
- Isolated from the Core OS
- AppFW enforced security
- No default UI



AGL Architecture



Service isolation

Run services with UID<>0 SystemD is your friend

- Create dedicated UID per service
- Use Linux MAC and Smack DAC to minimise open Access

Drop privileges

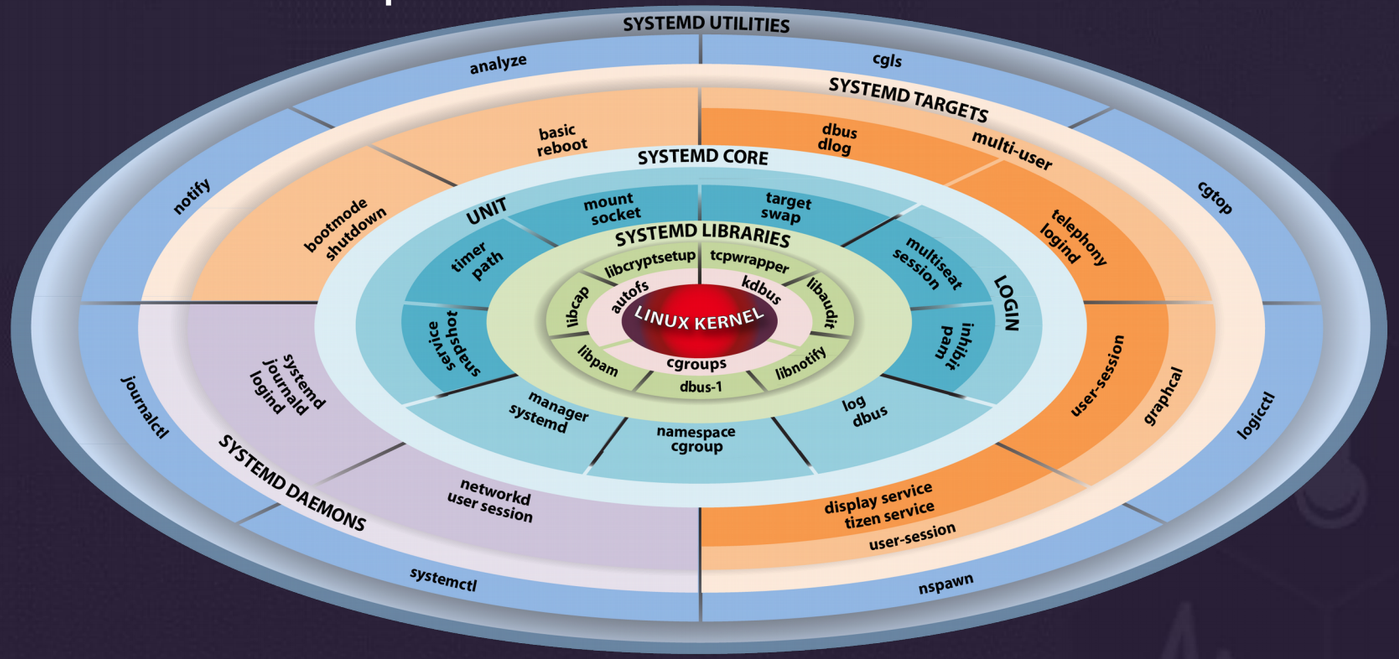
- Posix privileges
- MAC privileges

C-goups

- Reduce offending power
- RAM/CPU/IO

Name Space

- Limit access to private data
- Limit access to connectivity

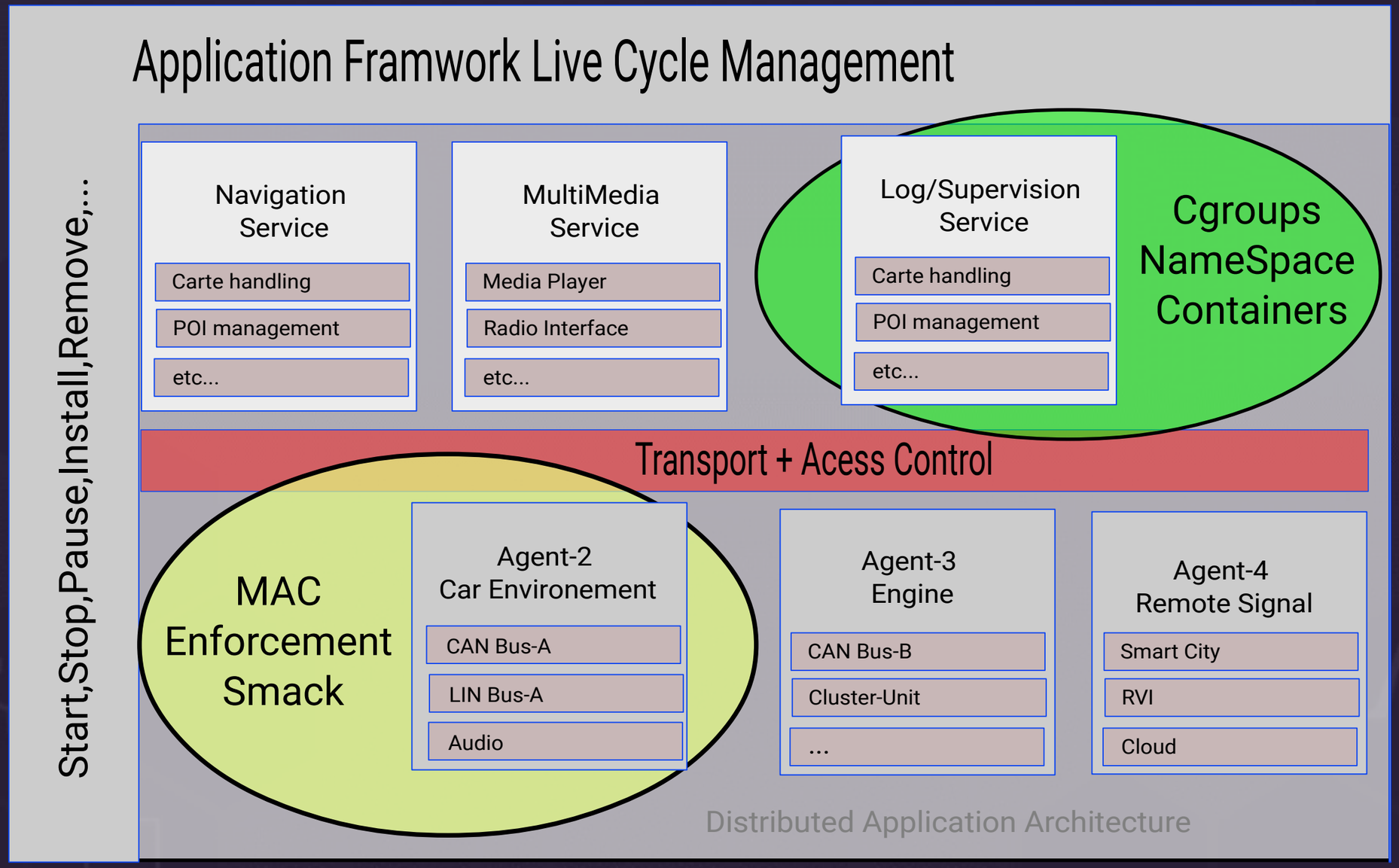


https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
https://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.2/capfaq-0.2.txt
http://man7.org/linux/man-pages/man7/namespaces.7.html
https://en.wikipedia.org/wiki/Mandatory_access_control
https://en.wikipedia.org/wiki/Discretionary_access_control

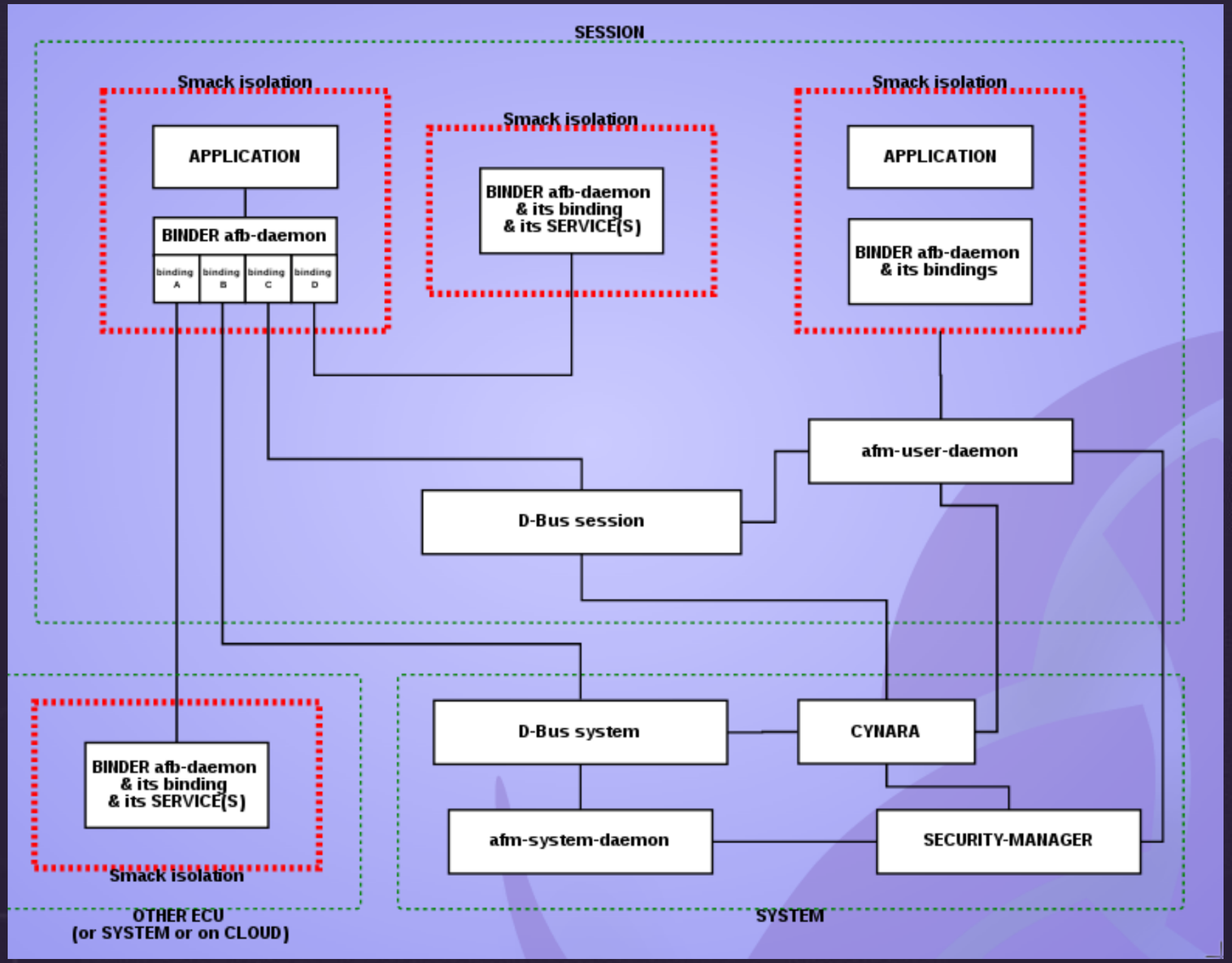
Segregate Apps from OS

- **Application Manager**
  - One system daemon for application live cycle installs, update, delete
  - One user daemon per user for application start, stop, pause, resume
  - Create initial share secret between UI and Binder
  - Spawn and controls application processes: binder, UI, …
- Security Manager
  - Responsible of privilege enforcement
  - Based on Cynara + WebSocket and D-Bus for Legacy)
- **Application & Services Binders**
  - Expose platform APIs to UI, Services, Applications
  - Loads services/application plugins :Audio, Canbus, Media Server…
  - One private binder per application/services [REST, WebSocket, Dbus]
  - Authenticate UI by oAuth token type
  - Secured by SMACK label + UID/GIDs
  - AppBinders runs under user $HOME

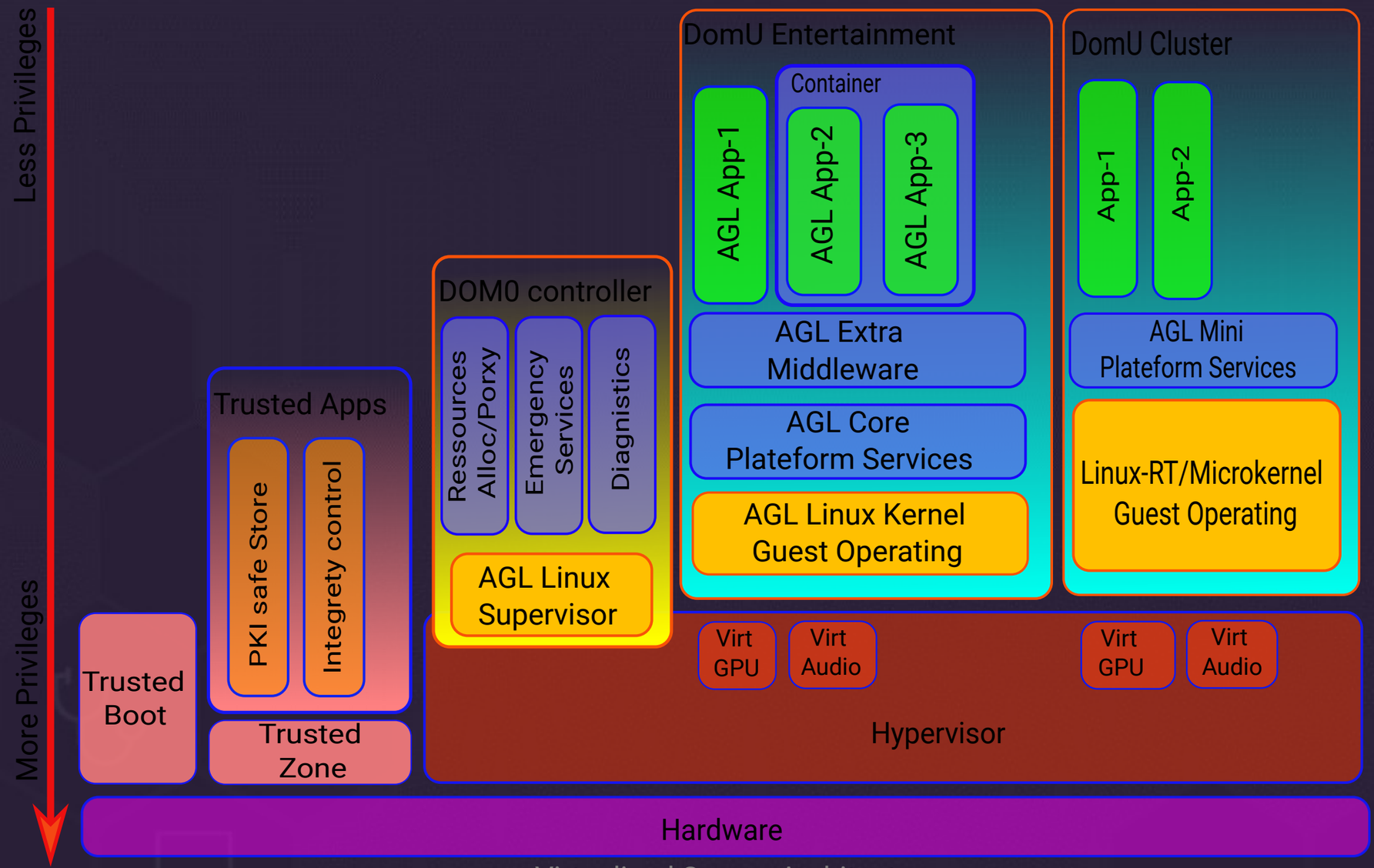
AGL2 Application Security



AGL2 AppFW logic



AGL2++ Virtualised Architecture



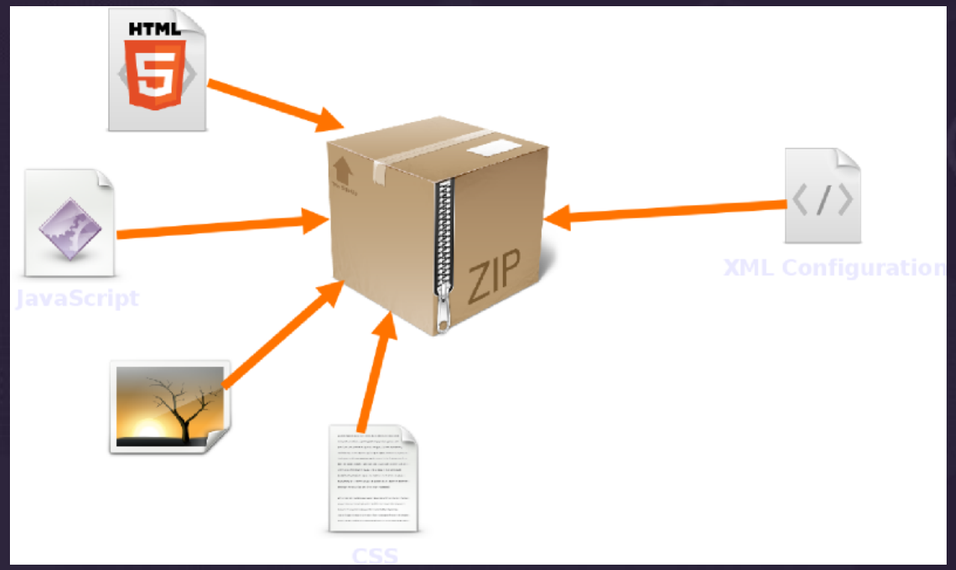
Virtualized Secure Architecture

Building the OS

- **Collection of Yocto Layers**
  - Multi-Architecture (Intel, ARM)
  - Multiple Haker Board support (Minnow, Joule, R3, RasberryPI 3).
  - Hardening by design
  - Critical services provided
  - Design for custom additions
- **No imposed UI**
  - Home Screen as an API
  - Local (Native or HTML5) or remote UI (via REST API)
- **Application and Middleware**
  - Built independently (via yocto SDK)
  - Web Socket based AppFW for easy integration
  - App and Middleware run in isolated security domains

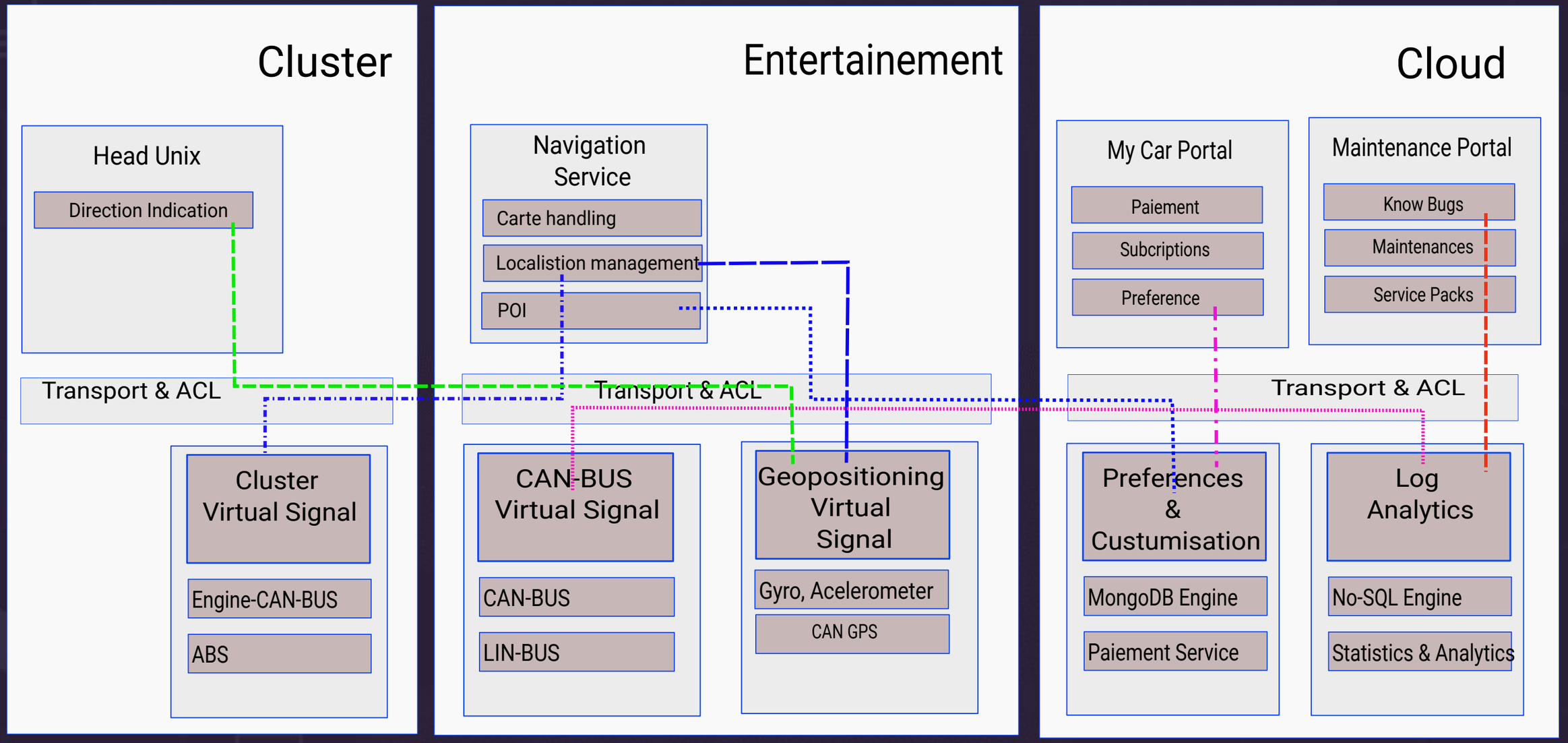
To write an App

- **Write back-end binding**
  - Adds the specialised API to the system
  - Accessible by Web Socket or slow legacy D-Bus
  - Run in its own security domain
  - Can be cascaded



- Write the Front end
  - Typically in HTML5, QML but open to any
  - Connect to back-end binding using REST with secured key (OAuth2)
  -
- **Package**
  - Based on W3C widget
  - Feature allow to handle AGL specificities
  - Install via the AppFW

AGL2+ Distributed Architecture



Multi ECU & Cloud Aware Architecture

Attacking IoT, a viable business

- **Ransom model**
- Stall manufacturing
- Immobilise expensive items (e.g. your car)
- …

- **Competitive advantage**
- Collecting R&D, manufacturing data
- Disturbing production line

- **Indirect**
- Cheap robot for DDoS
- Easy entry point

Security fundamentals

Minimise surface of attack
Control the code which is run
Provide a bullet proof update model
Track security patches
Use HW security helpers when available
Limit lateral movement in the system
Develop and QA with security turned on
Do not rely on human but on platform and tools

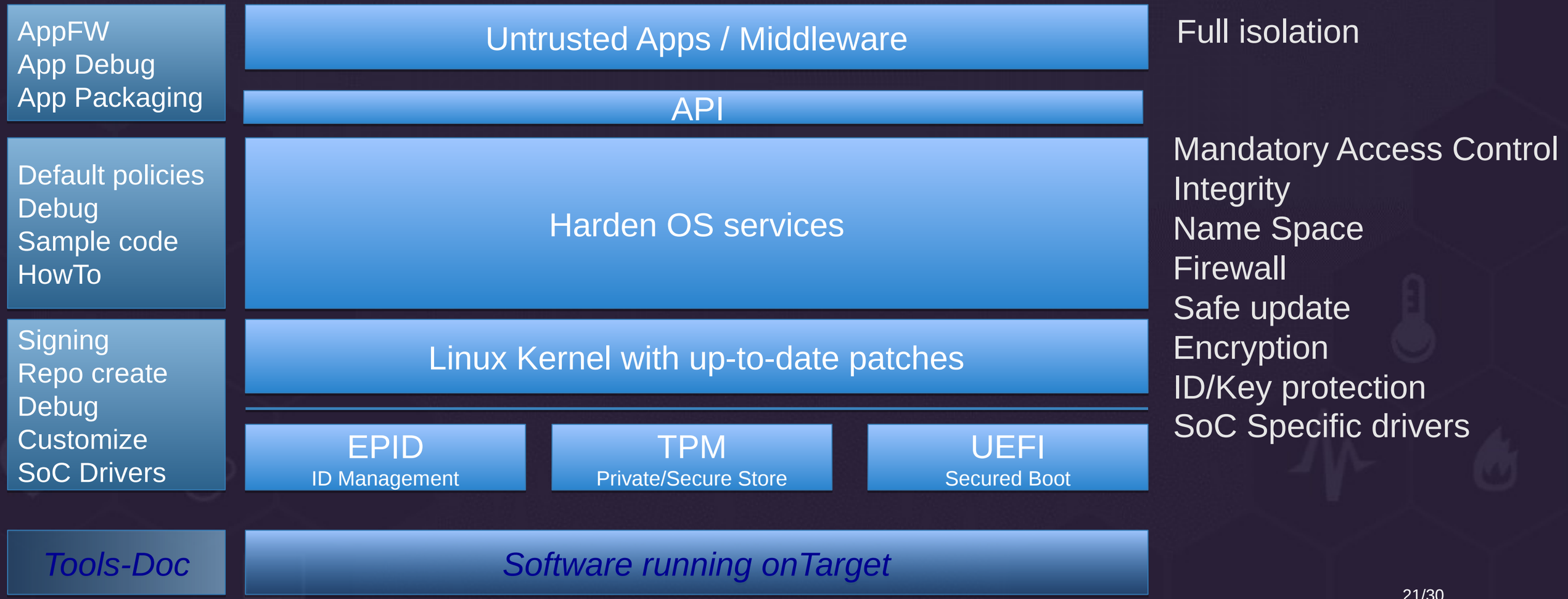
Security cannot be added after the fact

Do not rely on human

- **Security experts are out of reach**
- 9M Mobile Developers
- 8M Web Developers
- 0.5M Embedded Developers
- How many Embedded Security Developers ?

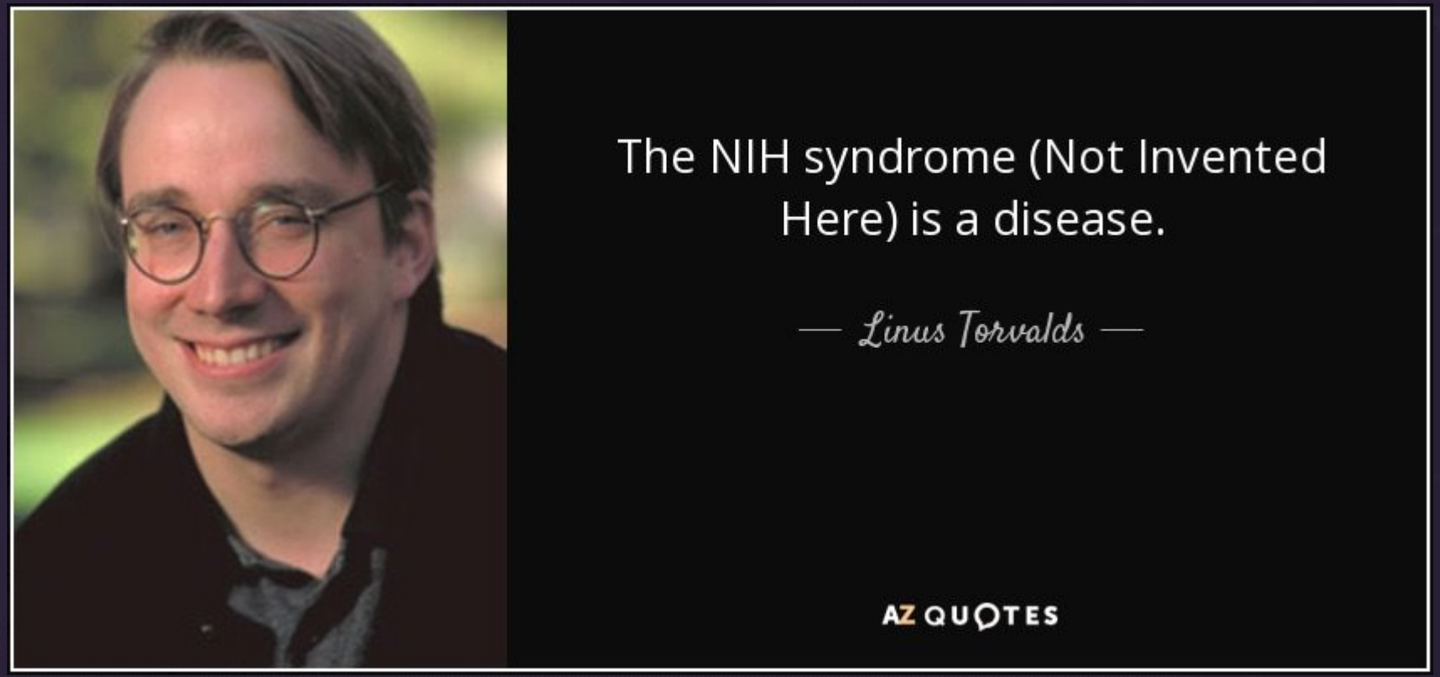
- **Human are unreliable**
- We do not have the time now
- Oups, it’s too late to change it
- No one is interested by our system
- We are too small
- ...

Concepts are Known but what about implementation?



Conclusion

- **AGL is Industry friendly**
  - Automotive have very generic requirements
  - Reuse potential is huge
  - AGL is really open source
  - In AGL code remains king
- **Security ready model**
  - Hardeling comes for free
  - Cybersecurity is a permanent focus
- **Application and Middleware are isolated**
  - AppFW is designed to connect modules via WebSockets
  - Business logic and UI are easy to isolate
  - App and Middleware SW is based on well know Web technologies



Fosdem 2017, Brussel, Be dominig.arfoll@fridu.net

Links

https://www.automotivelinux.org/
https://gerrit.automotivelinux.org/gerrit/#/q/status:open
http://docs.automotivelinux.org/
https://vimeo.com/channels/1196445

Backup slides

Container "A mixed blessing"

Easy to use

- Detach the App from the platform
- Integrated App management
- Well known

Not very secure

- Unreliable introspection
- MAC has no power on the inside of a container
- Updating the platform does not update the middleware
- Beside the Kernel each App provide its own version of the OS
- Each App restart requires a full passing of credential
- RAM and Flash footprint are uncontrollable
- Far more secured with Clear Container but not applicable to low end SoC.

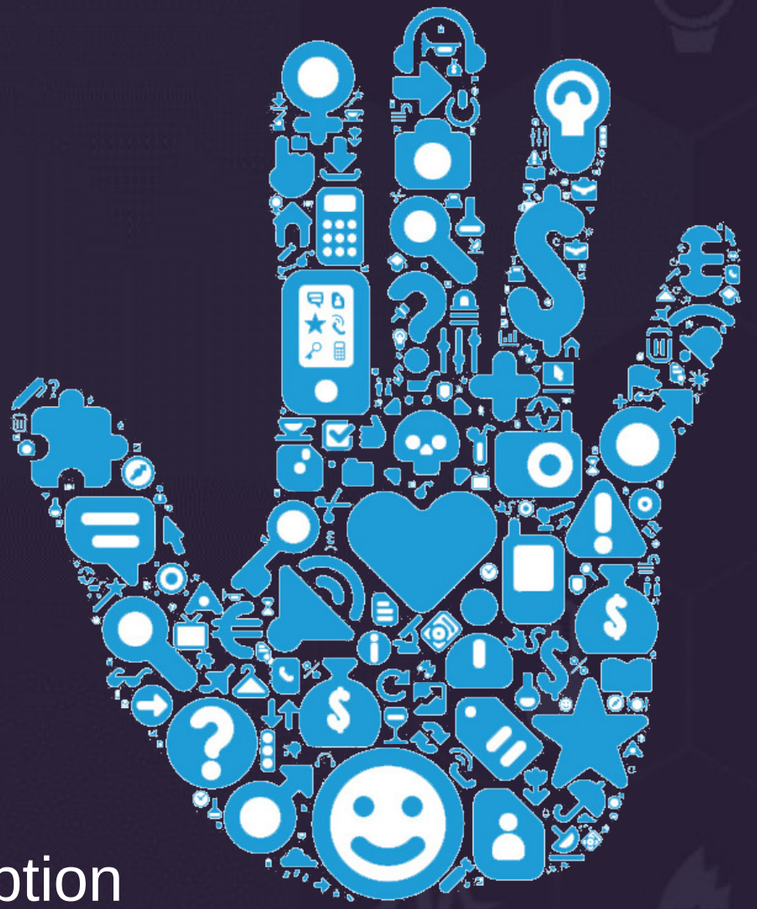
Only I/O via network

- Well equipped for Rest API
- All other I/O requires driver level access or bespoke framework.

https://www.opencontainers.org/
https://lwn.net/Articles/644675/

Know who/what you trust

- **Trusted Boot : a MUST Have Feature**
  - Leverage hardware capabilities
  - Small series & developer key handling
- **Application Installation**
  - Verify integrity
  - Verify origin
  - Request User Consent [privacy & permissions]
- **Update**
  - Only signed updates with a trusted origin
  - Secured updates on compromised devices are a no-go option
  - Factory reset built-in from a trusted zone
  - Do not let back doors opened via containers
  - Strict control of custom drivers [in kernel mode everything is possible]



Layered Architecture

- **Client/UI (untrusted)**
  - Risk of code injection (HTML5/QML)
  - UI on external devices (Mobiles, Tablets)
  - Access to secure service APIs [REST/WS]
- **Applications & Services (semi-trusted)**
  - Unknown developers & Multi-source
  - High-grain protection by Linux DAC & MAC labels.
  - Run under control of Application Framework: need to provide a security manifest
- **Platform & System services (trusted)**
  - Message Services started by systemd
  - Service and API fine grain privilege protection
  - Part of baseline distribution and certified services only

Bullet proof update and ID

Update is the only possible correction

- Must run safely on compromised devices
- Cannot assume a know starting point

Compromised ID / keys has no return

- Per device unique ID
- Per device symmetric keys
- **Use HW ID protection** *(e.g. EPID)*

Non reproducibility

- Breaking in one device cannot be extended
- Development I/O are disabled
- Root password is unique *(or better a key)*
- Password cannot be easily recalculated

Security Check list

Control which code you run

- Secure boot
- Integrity
- Secure update

Isolate services

- Drop root when possible
- Drop privileges

Isolate Apps

- Apps are not the OS
- Enforce – restrict access to standard API

Identity

- Enforce identity unicity
- Use available HW protection

Encryption

- Network traffic
- Local storage

Control image creation

- No debug tool in production
- No default root password
- No unrequired open port

Continuous integration

- Automate static analysis
- QA on secured image

Help developer

- Integrate security in Devel image
- Provide clear guide line
- Isolate Apps from OS
- Focus on standardised Middleware