Application & Security Framework
AGL-2.0
Architecture Proposal

Winter/AGL-AMM Tokyo 2016

Fulup Ar Foll Lead Architect
fulup@iot.bzh

Who Are We ?

Fulup Ar Foll
Lead Architect

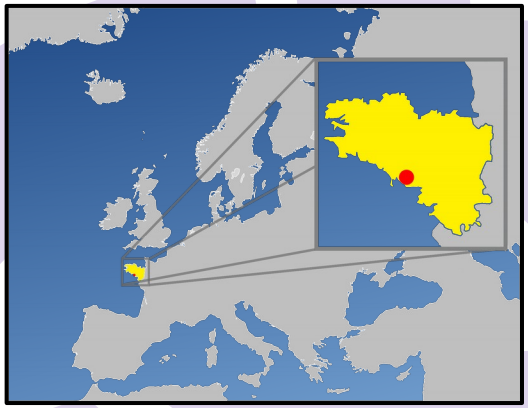
Stéphane Desneux
Release Engineer

Manuel Bachmann
Graphic/Multimedia

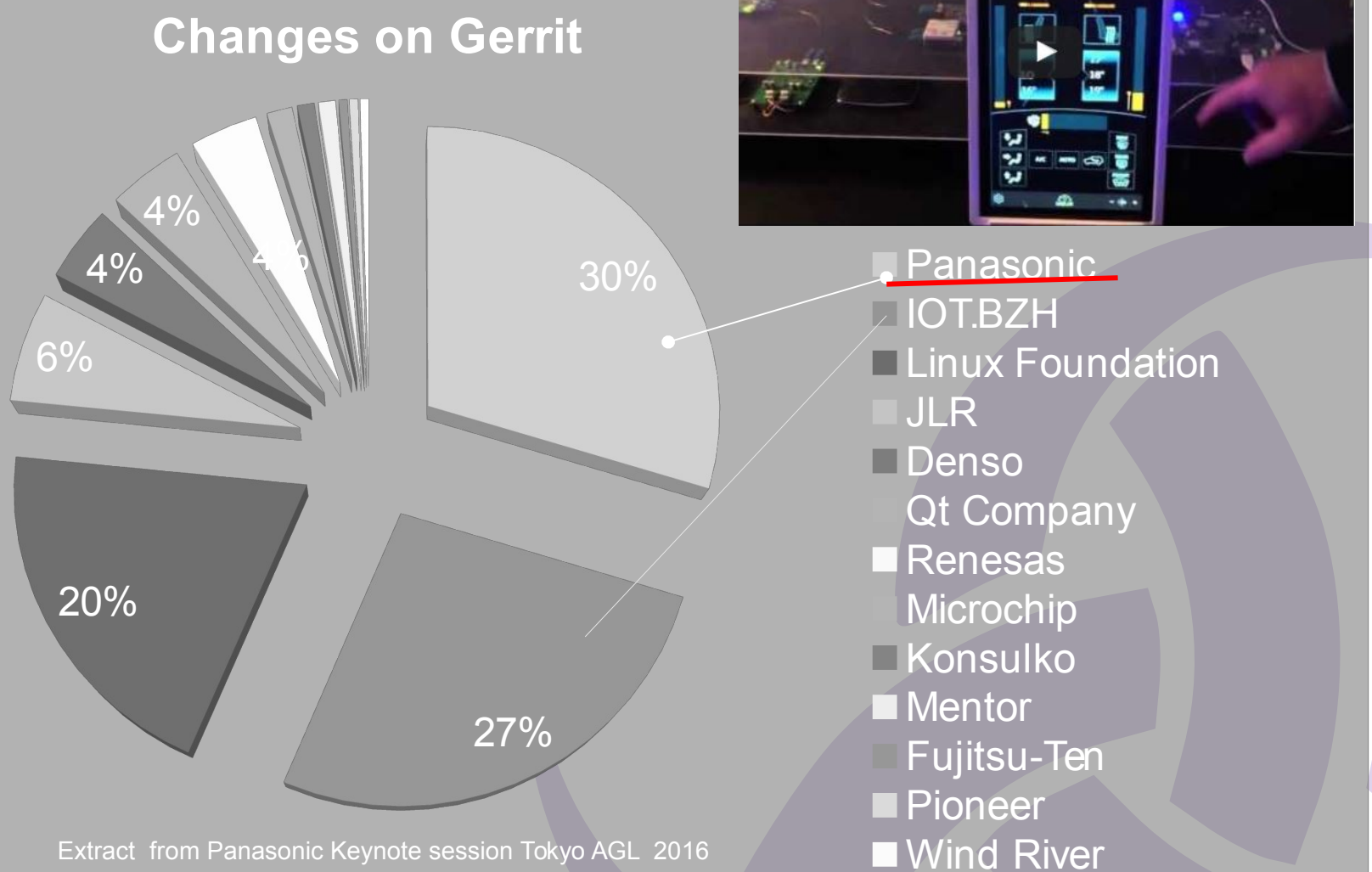
Yannick Gicquel
Kernel & QA

José Bollo
Security

- 100% Dedicated to AGL
- Work in Open http://github.com/iotbzh
- Based in South Brittany



AGL Contributions just below Panasonic



Extract from Panasonic Keynote session Tokyo AGL 2016

Why Securing Connected Cars?

- **Attacking cars is a viable business**
  - Expensive piece of equipment
  - Huge Mass market
  - Enough customers to steal from
- **Attacking cars is complex & expensive**
  - Hackers have time & money
  - Betting on hacker lack of skill is a very risky bet
  - One single small security hole might be enough
  - Automotive industry has limited knowledge and return of experience on being connected.

Car will be Connected & Connected Car will be Attacked

Security Fundamentals

- Minimize attack surface area
- Control the code which is run
- Provide a bullet-proof update model
- Apply security patches within days rather than weeks
- Leverage HW security helpers
- Isolate & compartmentalize wherever possible
- Development and QA with security turned on
- Analysis and report of incidents
- Provide adequate tools to develop with security enabled
- Do not rely on humans but on platform

Security Complexity Mitigation

- **Security Mechanism might be short circuit**
  - Lack of knowledge, Performances
  - Time-to-market, Cost concerns
- **Embedded Security Expert is a rare animal**
  - 9M Mobile Developers
  - 8M Web Developers
  - 0.5M Embedded Developers
  - How many Embedded Security Developers ?
- **Security cannot be added after the fact**
  - Must consist in built-in APIs & be transparent to applications
  - Developers SHOULD not to be in charge of security
  - Baked in from day one: Architecture, Dev, QA, Maintenance,etc.

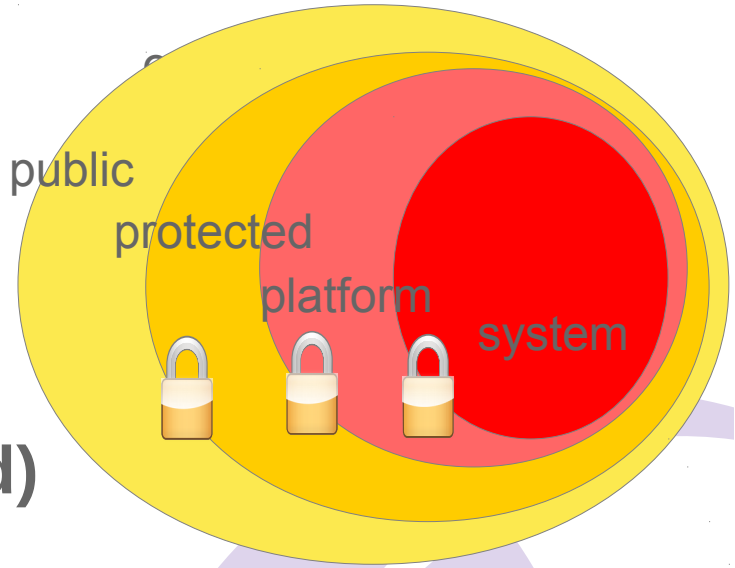
Need for Resilient Architecture

- **Smart Multi Layers Security Architecture**
  - Breaking an application should not break a full layer
  - Breaking a layer should not break the full system
- **Compromised ID / keys are lost for good**
  - Per-device unique ID
  - Per-device symmetric keys
  - Use HW ID protection
- **Non-Reproducibility of breakages**
  - Breaking in one car should not extend to all cars
  - Dev/Debug I/O, Sockets, … should be disabled
  - No Root Password & No shared super-user RSA key
  - Password, when used, should not be easy to compute

Make sure we Run the Right Code

- **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]*

Layers-based Architecture



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

Layer Service Segregation

Run services “not as root”. systemd is your friend

- Create a dedicated UID per service
- Use MAC and DAC to minimize open access

Drop privileges

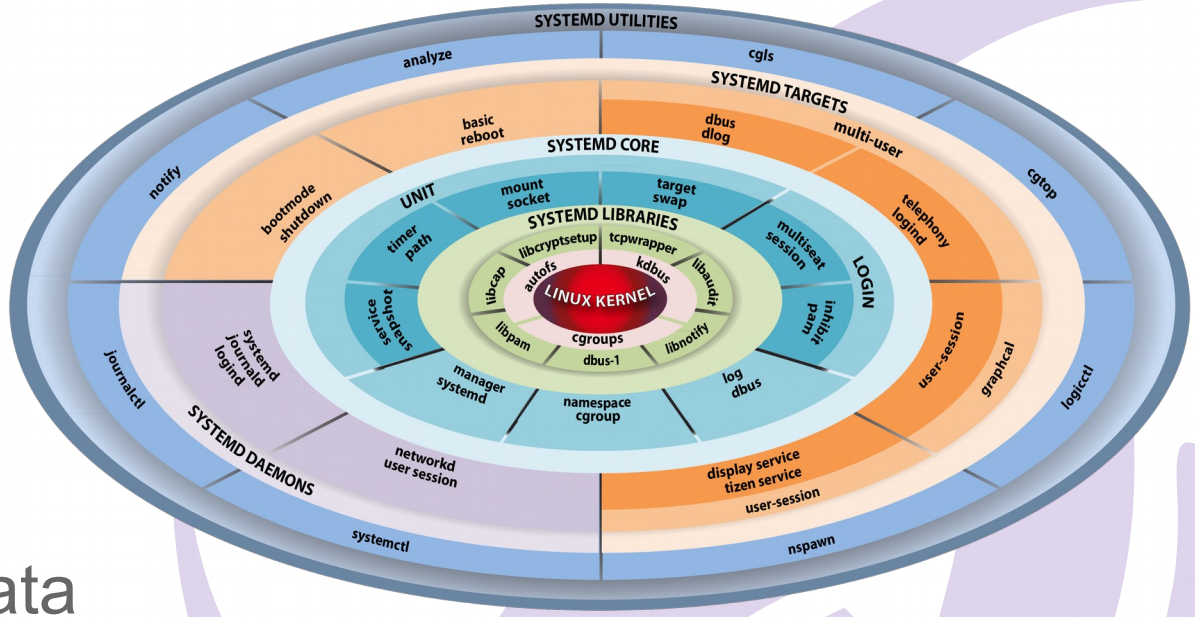
- POSIX privileges
- MAC privileges

Cgroups

- Reduce offending power
- RAM/CPU/IO

Name Space

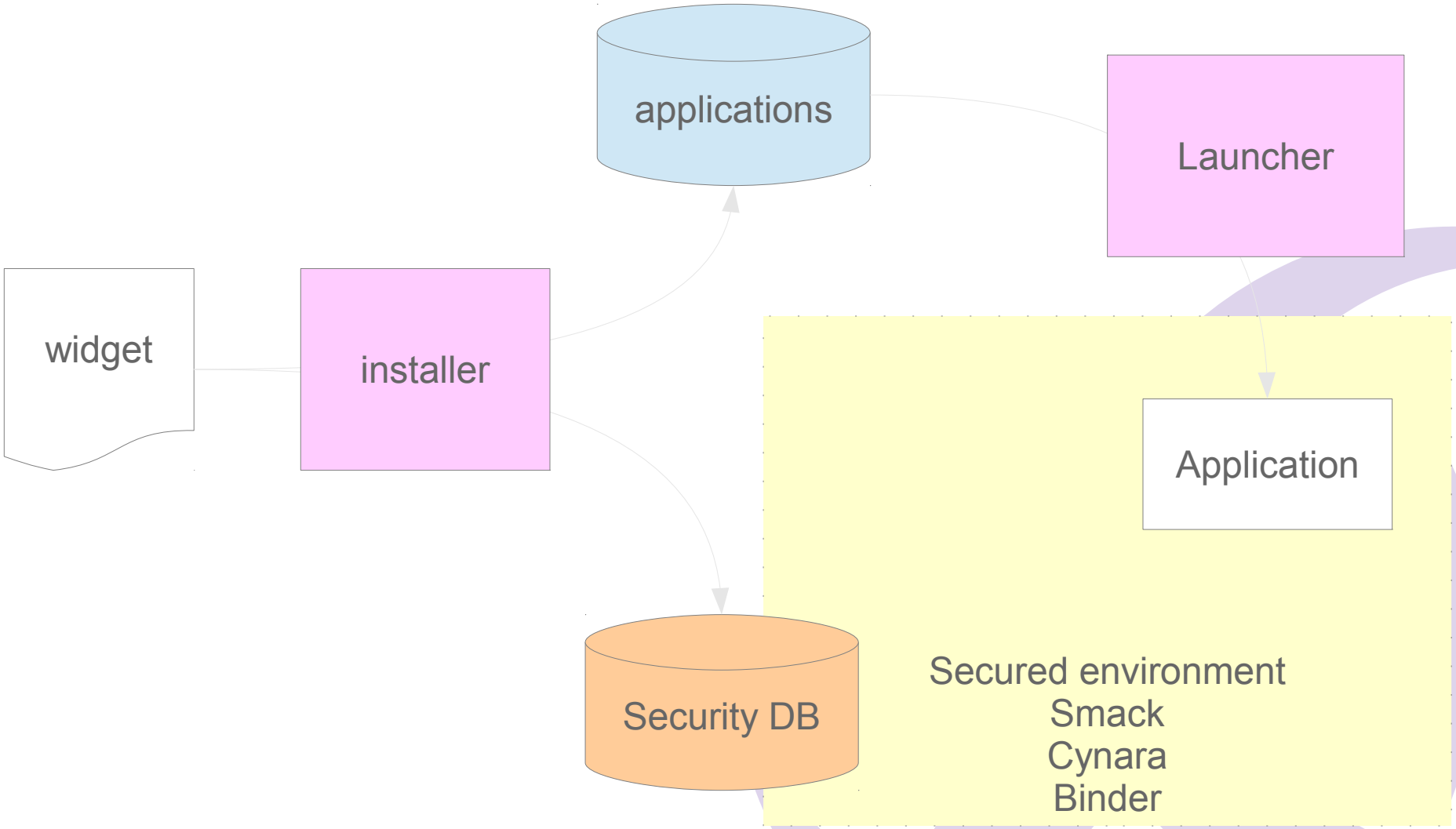
- Limit access to private data
- Limit access to connectivity



Application Security Framework

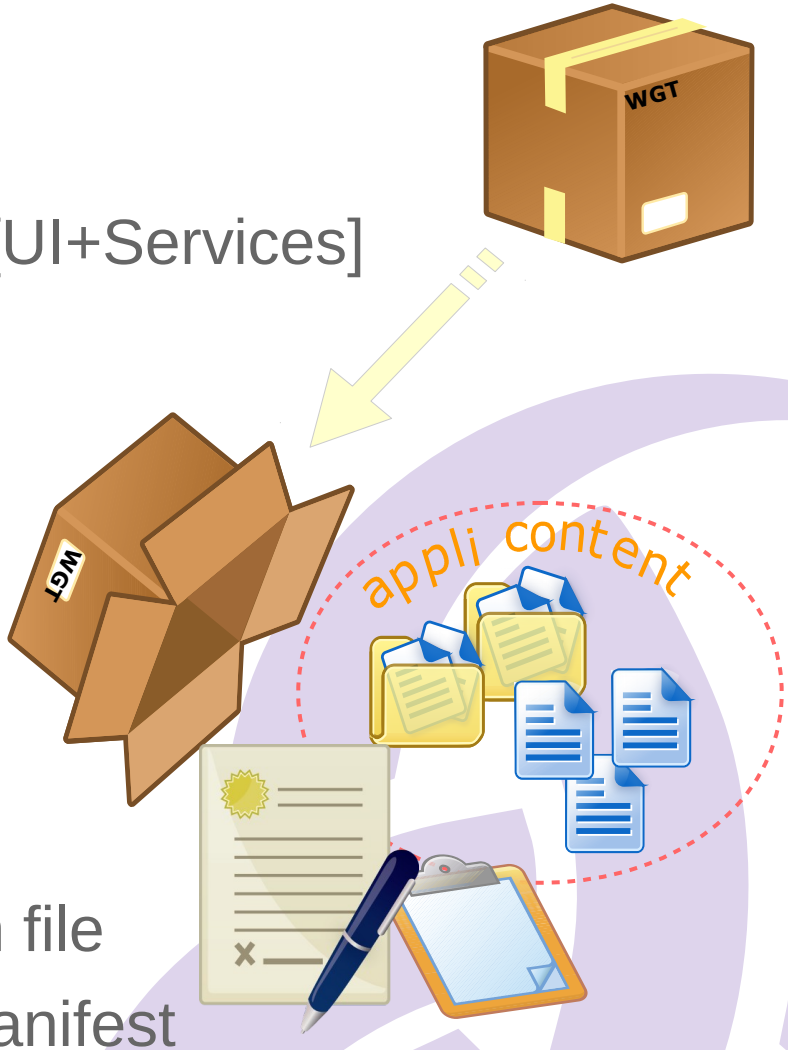
- **Application Manager**
  - One system daemon for application live cycle installs, update, delete
  - One user daemon per user for application start, stop, pause, resume
  - Create initial share secret between UI and Binder
  - Spawn and controls application processes: binder, UI, ...
- **Security Manager**
  - Responsible of privilege enforcement
  - Based on Cynara + Dbus plugin
  - Implement Intel-Meta-IOT-Security Yocto layer.
- **Application Binders**
  - Expose platform APIs as HTTP REST APIs to UI
  - Loads platform/application plugins :Audio, AM/FM Radio, Media Server…
  - One binder per application [REST server, based on libmicrohttpd]
  - Authenticate UI by oAuth token type
  - Secured by SMACK label + GIDs
  - Runs under user UID within $HOME

Application Framework

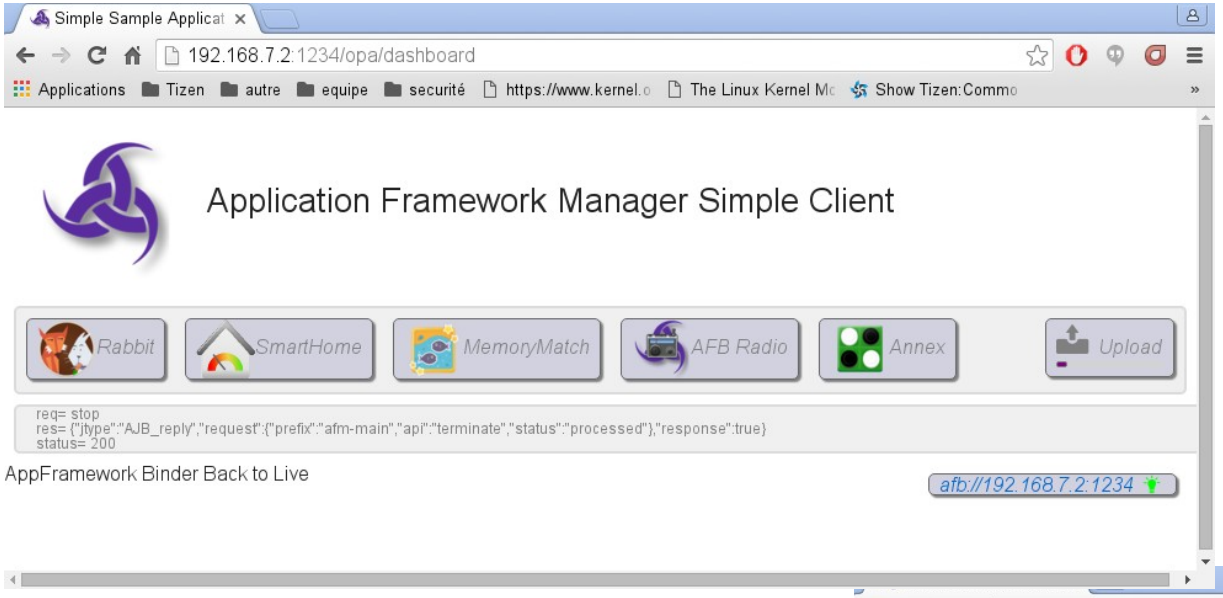


W3C Application Packaging

- **Secured Content**
  - Application files & directories [UI+Services]
  - Security Manifest
  - Signature files
  - Optional post install Scripts
  - Etc.
- **Format (ZIP)**
  - Public Key(s)
  - Manifest with SHA256 of each file
  - Digital Signature of content manifest
  - Cryptographic signature of the digest

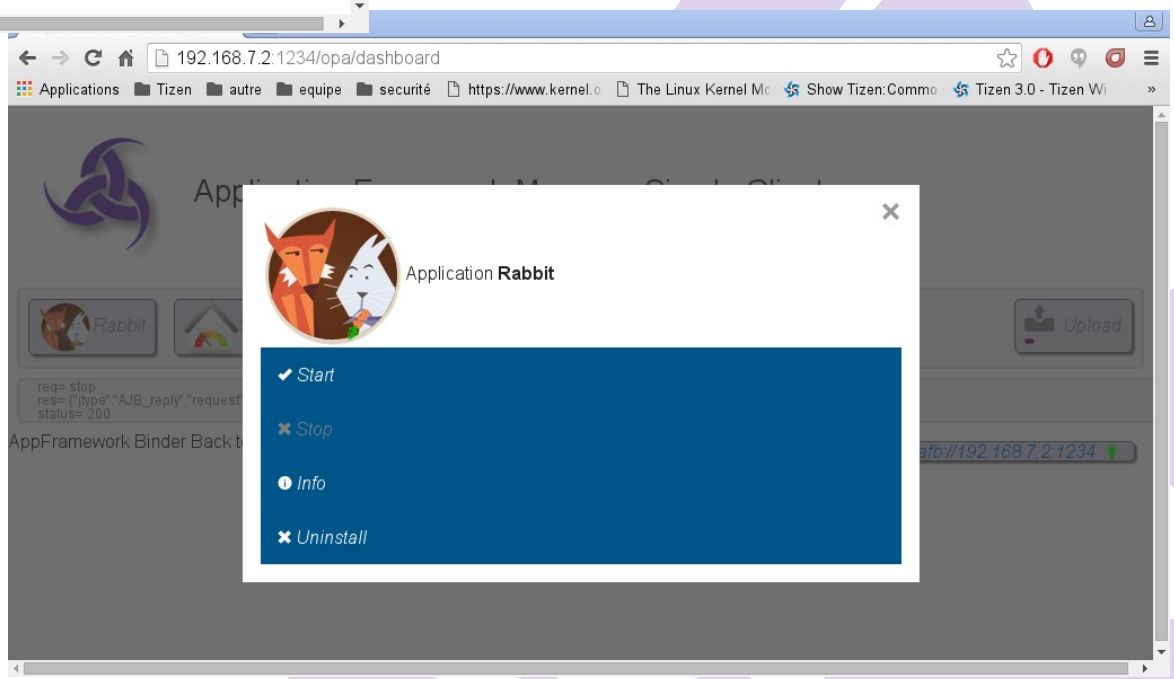


Application Home Screen



Request:
http://localhost/api/afm-main/runnables?token=xxxxx

Request:
http://localhost/api/afm-main/start&token=xxxx?appid=xyz

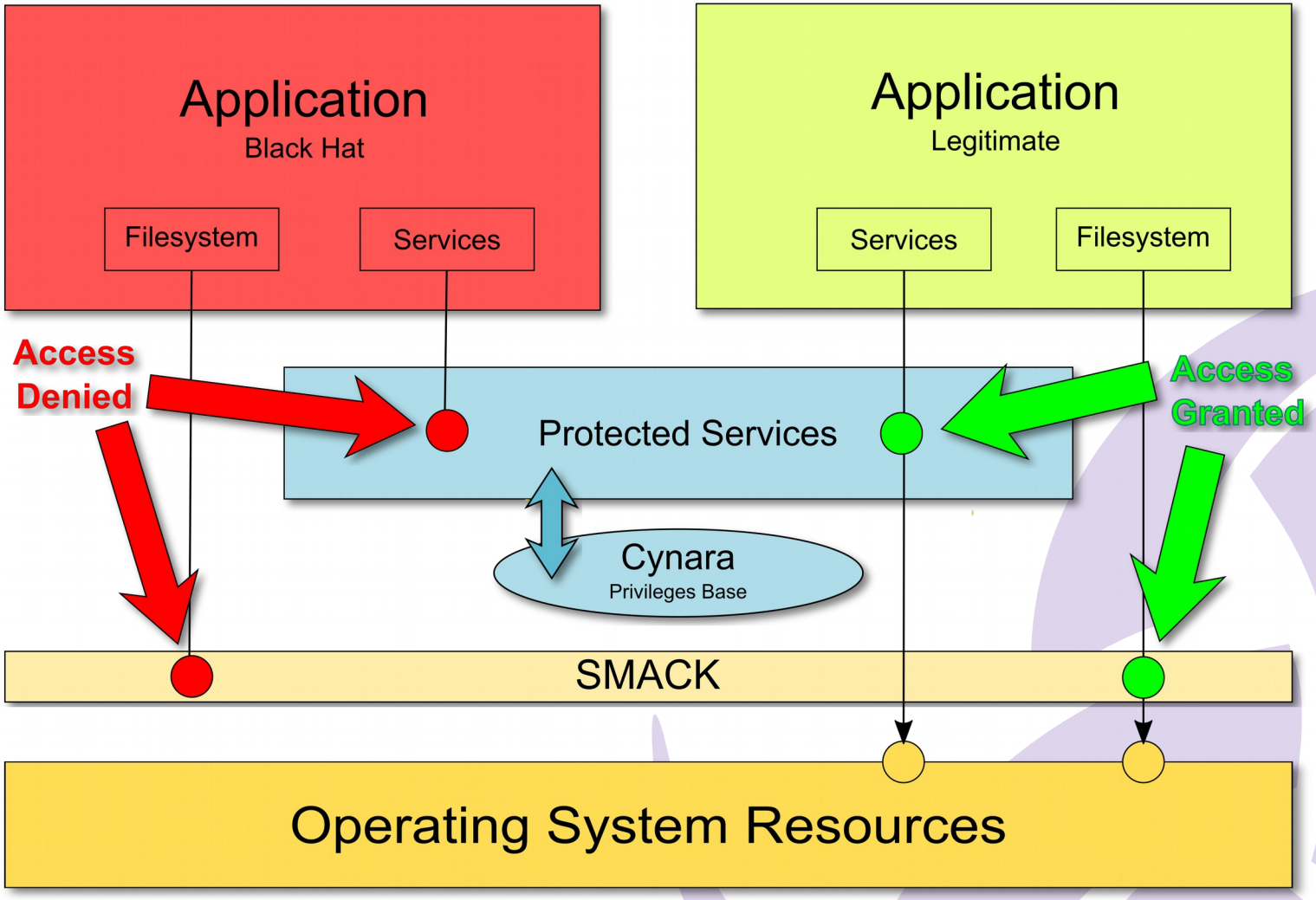


Managing Application Packages

- Easy-build SDK tools for CMake/Gulp/IDE
- Self-signed at least for development phase
- Signed by distributor for application stores
- Privileges based on origin and user consent
- Full life cycle through Application Framework
- Simple and Secured APIs (REST, D-Bus, ...)

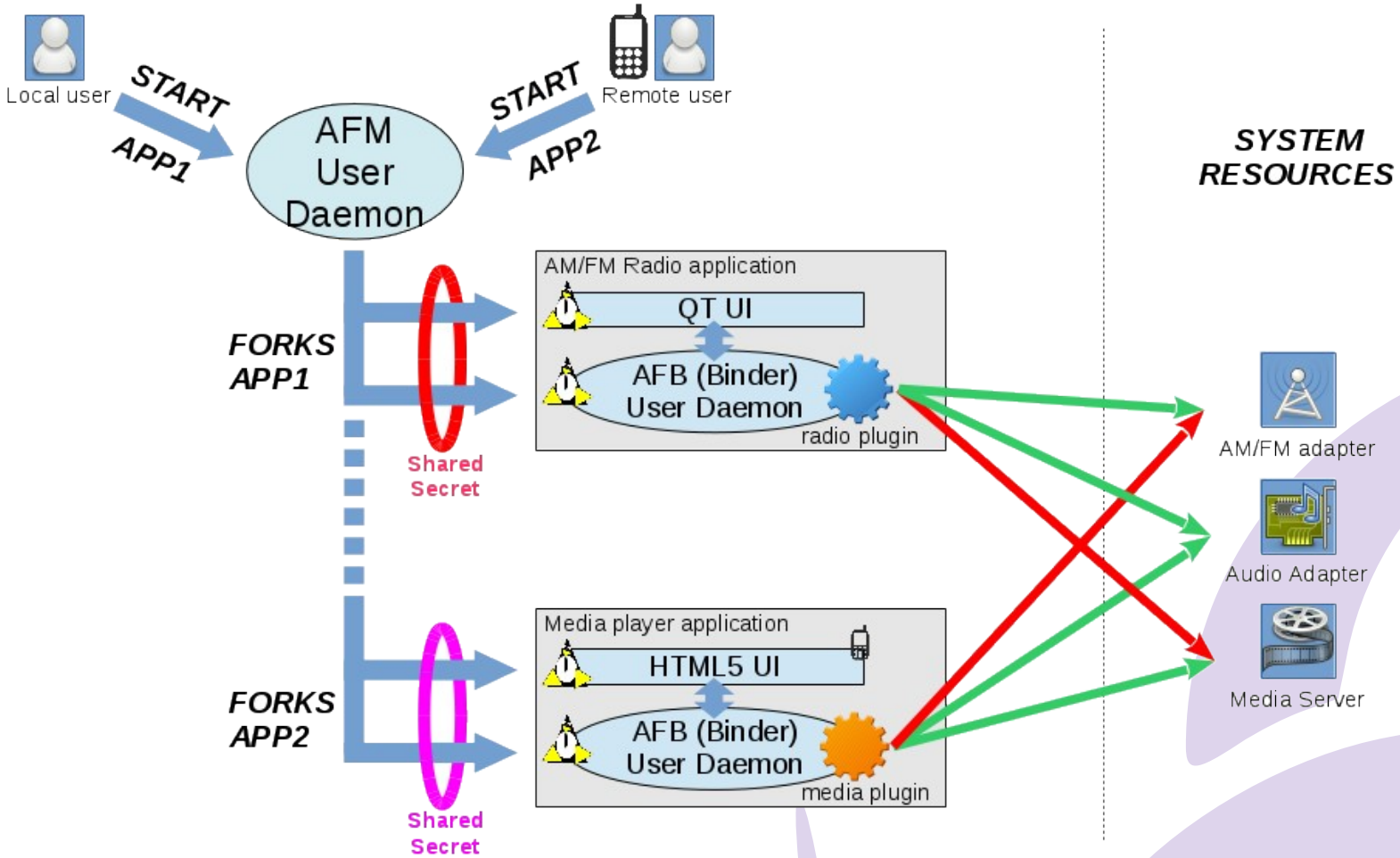
Rely on W3C Packaged Web Apps (Widgets)
https://www.w3.org/TR/widgets

Security Manager Logic



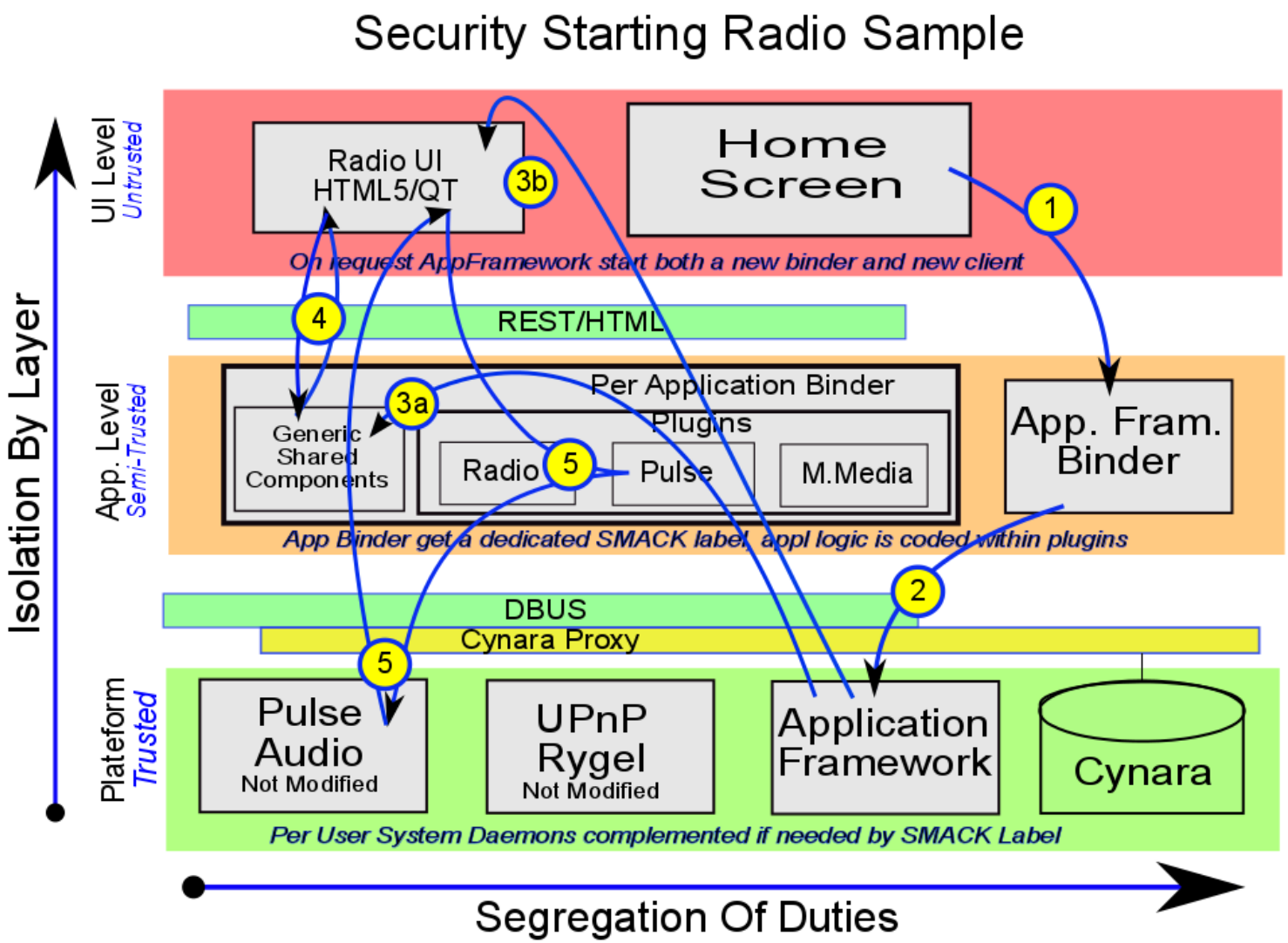
Intel Meta IOT ….

Application Framework Logic



- Binder APIs authenticates by OAuth Tokens
- System Object Access control by SMACK
- Privileges to Services APIs controlled by Cynara

Sample Radio Application Flow



HTML5, QML & Native Apps

Security framework should make standard operations simple, while keeping complex operations possible.

- **Standard Model**
  - UI in HTML5 or QML or external device running in the untrusted zone.
  - Application plugins accessed through REST APIs and controlled by authentication tokens provided by the application framework.
  - Platform services stay unmodified, Cynara control is handled transparently at D-Bus level.
- **Ad Hoc Model** *(when standard approach is not possible)*
  - UI and Application logic run directly at Application level
  - Direct access to platform services bypassing D-Bus
  - Fine grain privileges accessed directly from a modified service daemon.
- **APIs as JSON specifications**
  - REST & D-Bus mapping depending on class of service
  - Independent of application framework & security model

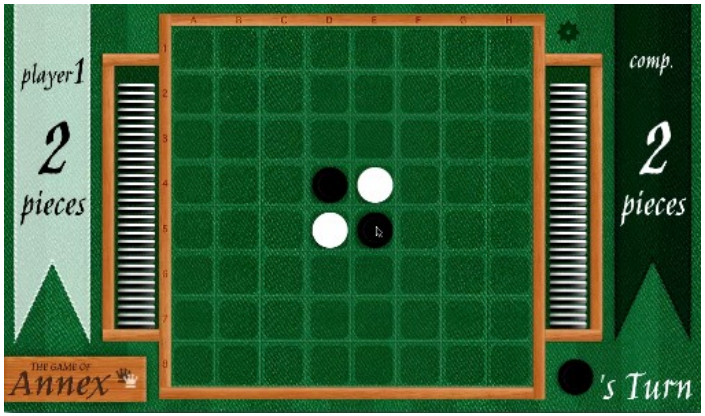
Managing Application Packages

- Easy-build SDK tools for CMake/Gulp/IDE
- Self-signed at least for development phase
- Signed by distributor for application stores
- Privileges based on origin and user consent
- Full life cycle through Application Framework
- Simple and Secured APIs (REST, D-Bus, ...)

Rely on W3C Packaged Web Apps (Widgets)
https://www.w3.org/TR/widgets

Demo

Power
Titre du Morce
Media
Bluetooth
IOT BZH
AGL-V1.0-Linux
1
2
3
4
5
6
FM88.12 Radio
KreizhBreizh
03:25:08
110.2
FM
40
50


player1
2
pieces
comp.
2
pieces
Annex
's Turn

Best Time
Find
Carrots!
Dodge
Foxes!
Run,
RABBIT,
Run!
4
0:01

Kernel LSM Choice

- Standard Kernel LSMs
  - ~~TOMOYO, AppArmor~~
  - SELinux, SMACK
- LSM choice is not structuring
  - Transparent to applications (MUST be)
  - Should only impact Application Framework
  - May have to change in ten years from now
- Why SMACK as 1st choice ?
  - Does the job and much simpler than SELinux
  - Samsung shipped a few millions of Mobile devices,TV, …
  - Intel published meta-intel-iot-security, a security manager with cleaned-up Tizen dependencies.
  - Intel accepted patches for smooth interface with app framework.

Incomplete TBD list

- **Application Framework**
  - A lot of Documentation: Security Blueprint, APIs,…
  - Smart integration with SDK (CMake, IDE, GULP, Debugger,…)
  - Integration with other transversal services ie: IVI shell, Resource Manager,...
  - Add missing functionalities: signalisation back channel, application pause/resume, Monitoring, Statistics,...
  - Define a strategy to attached privileges to a given chain of trust
  - Integration with existing services [AMB, SDL,…]
  - Application Store [dependencies handling, containerization, DRM, …]
  - Integration with existing hardware capabilities [crypto, trusted zone, ...]
- **User Management**
  - Multi Seat today & keep multi user possible tomorrow if needed
  - Authentication of external devices
  - Interface with cloud services
- **MUST HAVE features independent of Application Framework**
  - Secure boot
  - System and Application update strategy
  - Rootfs in read only for production mode
  - Etc…

Conclusion

- **Strong isolation & compartmentalisation**
  - Untrusted client can only access services through a serialized API and never have access to direct library mapping.
  - Application Binders in charge of presenting APIs to clients are constrained with a private SMACK label and run with userID rights.
  - Platform Services are protected by Cynara D-Bus proxy and only receive permitted requests.
- **Native applications and shortcuts remain possible**
  - Services not compatible with a full isolation model, can bypass part of the security framework while still benefiting partially of it.
- **Reduce costs of development**
  - Compliant with both internal display and external devices
  - Plugins are independent of Web Engine (browser) or Graphical Toolkits (Qt and others)
  - D-Bus platform services don't need to be changed.
  - Compliant with standard Web/Mobile UI toolkit such as AngularJS/Foundation

Application Framework is a MUST HAVE feature.

It is a structuring component that need to be approved before moving further to build a useful AGL distribution.

Further Information

- Some References
  - https://www.automotivelinux.org/automotive-grade-linux-security-white-paper
  - http://iot.bzh/download/public/2015/tizen-security-lessons-learnt-initial.pdf
  - http://bgr.com/2015/10/13/why-is-android-security-so-bad/
  - https://www.blackhat.com/presentations/bh-usa-09/BURNS/BHUSA09-Burns-AndroidSurgery-PAPER.pdf
- Download links
  - https://github.com/iotbzh/afm-main
  - https://github.com/iotbzh/afb-daemon
  - https://github.com/iotbzh/afm-client
  - https://github.com/iotbzh/afb-radio