History
& high level goals
History of the security model

- **Meego**
  - June 2014
- **Tizen 2**
  - Dec. 2014
- **Tizen IVI**
  - March 2015
- **Tizen 3**
  - June 2015
- **Ostro**
  - Nov. 2015
- **AGL**

- Security model changes from Tizen 2 to Tizen 3
- First meta-intel-iot-security release
- AGL spec 1.0 may 28th
- First Tizen 3 security release
- Start of refining framework
- Integration to AGL
Goals of the security framework

• **Security by design from day one**
  - Adding security after the fact is just impossible

• **Make security acceptable to developers**
  - Keep budget and time to market objectives under control

• **Open the system to partners**
  - Good for business

• **Provide abstraction above implementation**
  - Security Implementation detail should be transparent to user/developers
Design philosophy

KEEP IT SIMPLE
but flying
Why a refit version of Tizen framework?

- Same features
- Less code

Photography © Sarah Klockars-Clauser for openphoto.net
Why a refit version of Tizen framework?

- Same features
- Less code

- app-framework-main

- platform/appfw/app-installers
- platform/core/security/cert-svc
- platform/core/appfw/ail
- platform/core/appfw/aul-1
- platform/core/appfw/libslp-db-util
- platform/core/appfw/pkgmgr-info
- platform/core/appfw/slp-pkgmgr
- tizen-platform-config
- and probably some more...
Binder added to AGL framework?

Current AGL application and security framework

Binder adds methods to connect applications to secured API

app-framework-main

app-framework-binder
Security model, current design

• The same fundamental as Tizen
  – Smack + Cynara + Audit
  – Signed application
  – Permission management

• A new model to use and secure API:
  – Flexible connection to API
  – Token exchange
  – Automatic and hidden transport
  – Flexible message format
  – Debugging and logging features
  – Micro service architecture capability
Key components

• **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 + WebSocket

• **Application & Services Binders**
  - Expose platform APIs to UI, Services, Applications
  - Loads services/application plugins: Audio, Canbus, Media Server…
  - Private binders per application/services [REST, WebSocket, Dbus]
  - Authenticate UI by oAuth token type
  - Secured by SMACK label + UID/GIDs
Inside the kernel, **Smack checks all accesses** to files, memories, devices or sockets.

Any access not authorized is forbidden.
Cynara overview

1. installer provision permissions
2. cynara records permissions
3. application accesses protected API
4. provider of service checks the access
5. cynara grants or not the access (can ask user consent)
Smack + Cynara
Packaging applications: widgets

1. Developer team produces the application or service

2. Developer team describes its product and its requirements

Configuration file: config.xml
Security manifest: config.xml

<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns="http://www.w3.org/ns/widgets"
   id="org.openstreetmap.agl.pushviews" version="1.0.1">
  <name>Push street views</name>
  <icon src="pushviews.png"/>
  <content src="index.html" type="text/html"/>
  <description>Pushes camera views to openstreetmap</description>
  <author>Open Street Map</author>
  <license>GPL</license>
  <feature name="urn:AGL:widget:required-api">
    <param name="gps" value="auto"/>
    <param name="sensors" value="auto"/>
    <param name="camera" value="auto"/>
    <param name="pushviews:https://agl.openstreetmap.org/cloud/pushviews"
       value="cloud"/>
  </feature>
...
Security manifest: config.xml

... 

<feature name="urn:AGL:widget:required-permission">
  <param name="urn:AGL:permission:gps:public:read" value="required" />
  <param name="urn:AGL:permission:sensors:public:read" value="required" />
  <param name="urn:AGL:permission:camera:public:take:front" value="required" />
  <param name="urn:AGL:permission:camera:public:take:rear" value="optional" />
  <param name="urn:AGL:permission::public:HTTP:POST" value="required" />
  <param name="urn:AGL:permission::public:HTTP:GET" value="required" />
</feature>

</widget>
Packaging applications: widgets

3. Developer team signs its product
Packaging applications: widgets

4. Authorities sign the product
Packaging applications: widgets

5. zip the product, its config and its signatures
Global architecture
What is already in CC?
# The security Framework
(implementation of the security model)

## Manager part
(app-framework-main)

**Application management**
- Install
- Uninstall
- List
- Start
- Stop

## Binder part
(app-framework-binder)

**API connection**
- Unified client protocol (http, websocket)
- Loading and/or connecting to API
- Transparent connection to remote
Figure of the security framework

1. A service signed by Mitsuko
2. Installed in the system
3. Delivered to users
4. Loved by users
Figure of the security framework

cryptographic layered permissions

DB of apps

Security Context

application

Bindings

afb-daemon

afm-system-daemon

afm-user-daemon
Binder/Binding a mechanism to securely expose API

Security Context A
- User Interface
- Binder
  - Shadow API
  - local API

Transport Layer

Security Context Z
- Binder
  - Effective API

Same ECU
- Security Context S
  - Binder
    - Effective API

NETWORK

Other ECU or Cloud
- Binder
  - Effective API
Ongoing work
Main goals for ALS 2017

- ✔ Integration with Systemd
- ✔ Automatic connection to service and API
- ✔ Identity agent
- ✔ OpenId connect service
- ✔ Bugs!
WIP: integration & permission

Integration

- **Systemd** integration
- Automatic connection of APIs

Permissions

- Definition of **permissions**
- Implementation of cryptographic check of **permission** (PKI) at install
- Declaring **permissions at API level** in bindings
WIP: QA & developer & identity

QA & developer

• Test framework for QA
• More documentation
• Hooking events
• API for debugging
• Integration of packaging within SDK

Identity

• User identity handling
• OpenId connect service
Integration with systemd

- Replace **afm-user-daemon** with systemd
- Leverage the power of **systemd**
  - Definition of slice hierarchy for predefined Cgroup
  - Security-manager launch (user, groups, Smack label, ...)
  - Manages API dependencies
  - Leverage namespaces (containers)
- Link systemd features with AGL permissions
- Finalize specifications of features of config.xml
Integration with systemd

- **config.xml**
  - Integration with systemd
  - `/etc/afm/afm-unit.conf`

- **Format**
  - JSON
  - Mustache engine

- **Units**
  - `.service`
  - `.socket`

  - If permission validated

- **Temporary data**

- **Unit installer**
  - Units description
Definition of permissions

Proposal:

urn:AGL:permission:<api>:<level>:<hierarchical-name>

- The API providing (requiring) the permission. If starts with @ then it is a transversal permission (not linked to an API).
- One of the values:
  - system
  - platform
  - partner
  - publisher
  - owner
  - public
- Colon separated names possibly defining hierarchies of permissions

Advantage: self declaration of api, level and scoping.

Note that other schemes can be defined for being used together with this one.
Cryptographic check of permission

- PKI model for signing authority
- Public keys distribution
- Use cryptography to compute the maximal set of permission for an installed app
- Records and check publisher and owner certificates
- Link file digests with IMA
Permissions at API level

• Improves binding API definition
  – Define permissions by API and by verb
  – Enforce security
  – Improve compatibility management

• Automatic DBUS security rules

• Cynara check for websockets
  – Efficient automatic check of permissions
  – Enforce security for untrusted softwares
User identity handling

- Default users exist: agl-driver and agl-passenger
- Study the several ways of authenticating users
  - Key RF
  - Phone
  - Gesture
  - ...
- Provide dynamic allocation of uids for authenticated users
- Use uids to ensure privacy of users and Smack for privacy of applications
- Link identity with the Oauth2 service to allow federation
Identity agent

• Provide a service to manage data for the current user in the context

• Provide a service to request an openid connect token for the current context and to refresh associated tokens

• Provide a service that allow the car to be a connect id token provider
Hooking event and debugging

• Add hooks on events
• Provide an API to debug applications
  – Listen and record events
  – Generate events or requests from a debugger
  – Provide mock apis
  – Introspection
  – Compute statistics
• Provide an API for LOG control
  – Compatible with DLT spirit
• Link with Open API
Packaging integration in SDK

- Integration of PKI in SDK
- CMake & Autoconf features to create widgets
- Study how debug & dev can be provided
  - On a real target
  - On an emulator
  - Within common IDE (eclipse)
Future Work

- Update to new version of libmicrohttpd
- Optimisation of json performance
- Add new transport as Bus1
- Bindings in javascript
- Secured websockets
- Integration of Open API for docs and tests
- Management of services
- Helper library to speed up bindings development
- Container instanciation within systemd
- Study new WS protocol (optimisation of the current one, use of binary data like CBOR or BJSON)
Agenda

- For ALS 2017 (31 may)
  - Integration with Systemd
  - Automatic connection to service and API
  - Identity agent
  - OpenId connect service
- For AMM winter 2017 (oct)
  - QA and tests
  - Logging + hooking + debugging + statistics
- For CES 2018 (January 2018)
  - Cryptographic check of levels of permissions
Questions

Too late for prevention