systemd integration and user management

José Bollo
jose.bollo@iot.bzh
Overview of applications

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

Security Context Z
- Binder
  - Effective API

Effective API

Transport Layer

Same ECU
- Security Context S
  - Binder
    - Effective API

Other ECU or Cloud
- Security Context Z
  - Binder
    - Effective API

NETWORK
Setting the security context

1. fork

Launcher

Launcher clone

2. prepare security context

Launcher clone with targeted security context

3. drop privileges

Launcher clone with targeted security context

application in its security context

4. exec
Launcher and launching

• The launcher set the setting security context of applications that it launches

• Before, integration with systemd, the daemon `afm-user-daemon` (privileged) prepared the security environment of the forked process before to launch it for real. It performed the following actions:
  
  – Set supplementary groups
  – Set smack label
  – Drop capabilities

  Done in fact using the client library of the security-manager of Tizen 3
Issues with afm-user-daemon launching

- afm-user-daemon had to be privileged
- afm-user-daemon was NOT managing:
  - Autostart of services
  - Dependency of applications to services
  - Resource management using cgroups
  - Namespace containerization
  - Setting environment of a different user

But systemd does!
Benefits of using systemd

- Removal of one privileged process: afm-user-daemon
- Access and use of high level features:
  - Socket activation
  - Management of users and integration of PAM
  - Dependency resolution to services
  - Cgroups and resource control
  - Namespaces containerization
  - Autostart of required API
  - Permissions and security settings
  - Network management
  - ...
Design philosophy

KEEP IT SIMPLE and Flying
History of the framework

Tizen 3

Ostro

AGL

- AGL spec 1.0 may 28th
- First tizen 3 security release
- Start of refitting the framework
- Introduction of the binder
- Integration to AGL
- Integration to systemD
- Integration to systemD with users

- March 2015
- June 2015
- Nov. 2015
- March 2017
- Oct. 2017
Integration with systemd in AGL

• Charming chinook:
  – Privileged afm-user-daemon+security-manager as launcher

• Daring dab:
  – Unprivileged afm-user-daemon leverages systemd --user to launch applications
  – Works only for user root 😞

• Electric Eel (featuring EE RC3):
  – No need of afm-user-daemon (emulated for compatibility during transition)
  – Half privileged afm-system-daemon leverages systemd – system (pid1) to launch applications (and services) for system and for any user
Workflow
Packaged applications (widgets)

- **appli content**
- **signed content**
- **config.xml**
- **packaged content**
Figure of the framework

Security Context

application

Bindings

afb-daemon

afm-system-daemon

systemd

cryptographic layered permissions

DB of apps

= Widget’s files + systemd units
Generation of systemd units

- config.xml
- /etc/afm/afm-unit.conf
- json
  - temporary data
  - Units description
- Mustache engine
- *.service
- *.socket
- Unit installer

If permission validated

18 October 2017
AGL AMM, Dresden, systemd integration
Internal widget config

EXTRACT OF CONFIG.XML

```xml
<feature name="urn:AGL:widget:required-permission">
  <param name="urn:AGL:permission:real-time" value="required" />
  <param name="urn:AGL:permission:syscall:*" value="required" />
</feature>
```

INTERNAL JSON

```
"required-permission":{
  "urn:AGL:permission:real-time":{
    "name":"urn:AGL:permission:real-time",
    "value":"required"
  },
  "urn:AGL:permission:syscall:*":{
    "name":"urn:AGL:permission:syscall:*",
    "value":"required"
  }
}
```

Internal transform during installation
Mustache templating

Hello {{name}}
You have just won {{value}} dollars!
{{#in_ca}}
Well, {{taxed_value}} dollars, after taxes.
{{/in_ca}}

Hello Chris
You have just won 10000 dollars!
Well, 6000.0 dollars, after taxes.

Details: https://mustache.github.io/
https://gitlab.com/jobol/mustach
Example of afm-unit.conf

[Service]
SmackProcessLabel=User::App::*{{:id}}
User=%i

CapabilityBoundingSet= AmbientCapabilities=
Slice=user-unlimited-%i.slice

{{#required-permission}}
  {{#urn:AGL:permission::public:display}}
    SupplementaryGroups=display
  {{/urn:AGL:permission::public:display}}
{{/required-permission}}

Setting of SMACK

UID of the user is the parameter

Conditionnal group
Demo
Transition mode

USER LAND

afm-user-daemon (legacy)

afm-util

dbus

afb-daemon

Bindings

Security Context

application

SYSTEM LAND

afm-system-daemon

ws-client

systemd
Managing resources
Possible resource control using slice

{{#required-permission}}
  {{#urn:AGL:permission::platform:unlimited}}
    Slice=user-unlimited-%i.slice
  {{/urn:AGL:permission::platform:unlimited}}
  {{^urn:AGL:permission::platform:unlimited}}
    Slice=user-limited-%i.slice
  {{/urn:AGL:permission::platform:unlimited}}
{{/required-permission}}
Possible container isolation

```{#required-permission}
{^{urn:AGL:permission::platform:not-sandboxed}}
  ReadOnlyPaths=/
  InaccessiblePaths=/home
 ReadWritePaths=/run/user/%i /home/%i /dev
  PrivateTmp=yes
{^{urn:AGL:permission::platform:not-sandboxed}}
{#/required-permission}
```

(we can also consider use of `systemd-nspawn`)
Dealing with users
User identity handling

- Default users exist: agl-driver and agl-passenger
- Existing 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 workflow

1. initiate authentication
   → PAM activation

2. start user session

Cloud identity

3. sync

USER SESSION

- User identity service
- User services
- User applications

- Plugable Authentication Module
- Standard
- Highly configurable
- Modular design
- Face recognition
- Voice identification
- Password

1. initiate authentication
2. start user session
3. sync
Identity service

- Manage user data
- By application
- Globally
- Manage synchronisation and caching
- Provide OpenId negotiation
Launching application

USER SESSION

Home screen
User applications

afm-system-daemon
systemd

UID of the client is the parameter
Questions

Available in meta-agl branch sandbox/jobol/sds