AGL Application Framework - Privileges Management

Version 2.0

September 2016

Abstract

In AGL, as in many other embedded systems, different security mechanisms are settled in the core layers to ensure isolation and data privacy. While the Mandatory Access Control layer (SMACK) provides global security and isolation, other mechanisms like Cynara are required to check applications permissions at run time.

Applicative permissions (also called "privileges") may vary depending on the user and the application being run: an application should have access to a given service only if it is run by the proper user and if the appropriate permissions are granted.

This document explains how applicative permissions are handled in the AGL Application Framework proposed by IoT.bzh.

Document revisions

Date	Version	Designation	Author
12 Sep. 2016	0.5	Initial revision	S. Desneux [IoT.bzh]
20 Sep. 2016	2.0	Review	J. Bollo [IoT.bzh]

Table of contents

1.Overview....4
2.Privilege: definition....7
2.1.Format....7
2.2.Levels....8
2.3.Privileges Hierarchies/Dependencies....9
2.4.Relationship with APIs....10
3.Cynara....11
4.Developing applications with privileges....13
5.Similar mechanisms on other systems....14
5.1.Tizen....14
5.2.Android....22

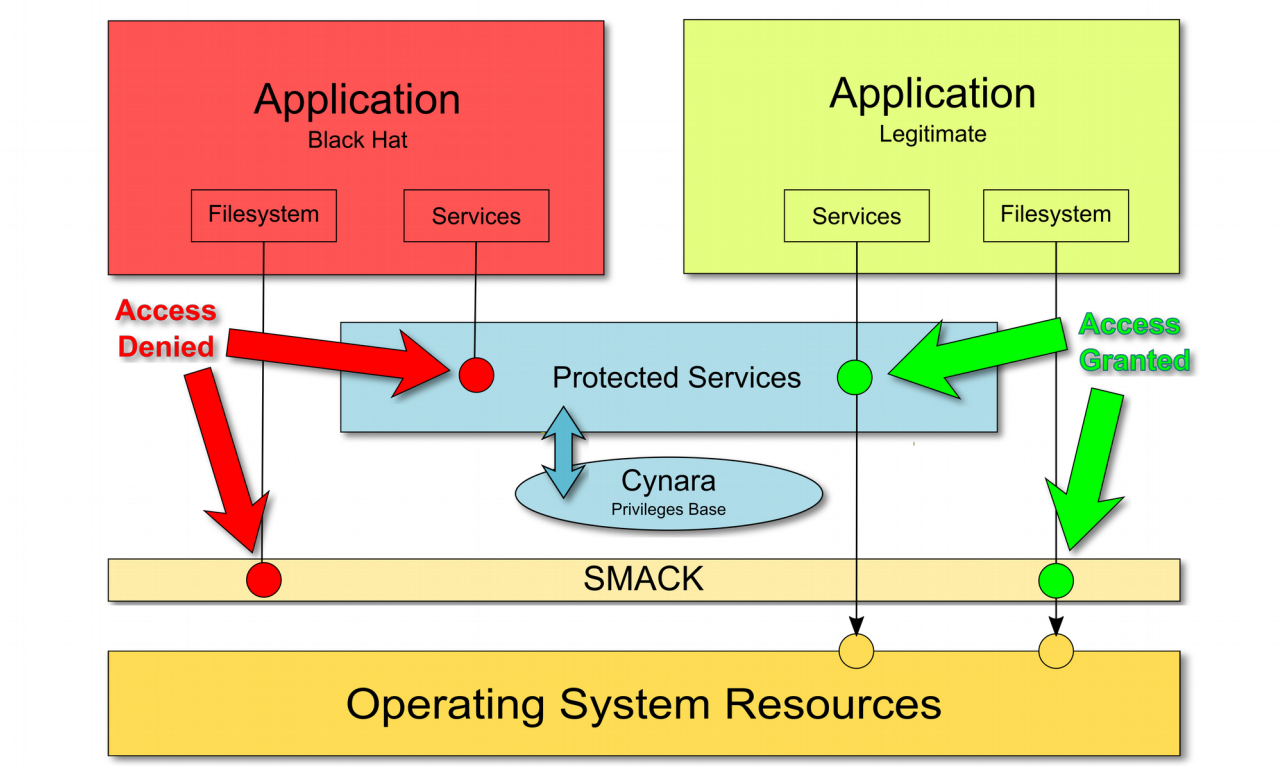
1. Overview

For AGL to fulfill minimal security requirements, there are two basic mechanisms to implement regarding applications execution. The system should be able to:

- Restrict permissions by user: Multiple users can be associated to a given car, such as the owner, the mechanic, the driver, and the passengers. The privileges of these users are not equivalent and some should have more rights than others. This is managed by permissions associated with users or profiles.
- Restrict permissions by application: An application has to request the needed permissions for proper operation. When requested permissions are granted, the application can then access protected services or resources.

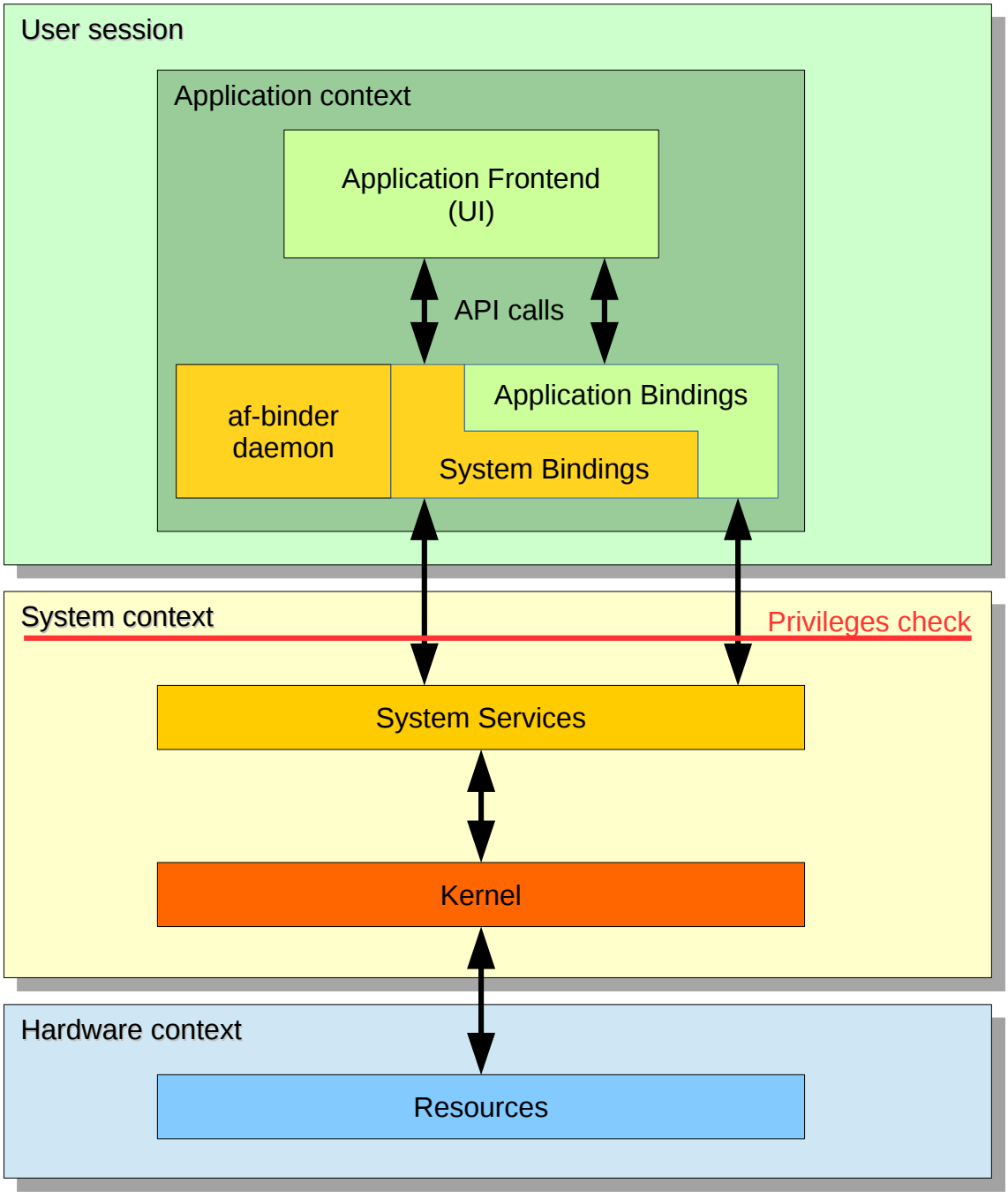
This document focuses on the later: when an "application" running for a "user" needs to access to resources restricted by "permissions", a check on the triplet (permission, user, application) must be done.

The following diagram summarizes how restrictions should apply:



In the current implementation of AGL Application Framework, access control mechanisms (DAC[1]/MAC[2]) are used for global security and isolation of a full system. Both mechanisms are used to force applications to use authorized communication channels and APIs to access lower level services and hardware. But there's a need for another mechanism responsible for checking applicative permissions: currently in AGL, this task depends on a policy-checker service (Cynara).

Here's the typical architecture used to run applications through AGL Application Framework:



1 DAC: Discretionary Access Control (POSIX user/group permissions)
2 MAC: Mandatory Access Control (currently implemented with SMACK)

The architecture is layered to provide different security contexts and force applications to make known APIs calls to access lower services and resources. The MAC mechanism (SMACK) is responsible for blocking direct accesses from an application to system services.

Any user installing an application expects that application to do what it claims and nothing more: for example, a cautious user would not install a calculator application that either needs to send SMS, to read his address book, or to use internet access.

Applicative permissions (or "Privileges") are used to verify if an application run by a specific user has the rights to access a service and use some parts of the service API. This operation is labeled "Privileges Check" in the previous diagram.

Applicative permissions make sense only if the principle of "Privilege Separation" is respected: it dictates that every application has to possess only the privileges and the resources necessary for its execution, and nothing more. So in case of failure of the system, the damage cannot exceed what is authorized by the privileges and the resources used (which may also be limited by the separation of privileges). Said differently: in theory, an application should always request a minimal set of privileges and never more than needed.

Therefore, applications that use sensitive/protected APIs must declare the required privileges in their configuration file stored in WGT packages: the required privileges are inspected and stored at installation time by the application framework and Cynara, after user approval.

At this step, the user can cancel the installation or tune the required privileges. Tuning privileges means setting the privilege in one of the following category:

- Permit: Use of the capability is always permitted, without asking the user.
- Deny: Use of the device capability is always denied

It's also possible to envision dynamic privileges granted/revoked by the user at run time by using some extra states:

- Blanket Prompt: User is prompted for confirmation the first time the API function is called by the widget, but once confirmed, prompting is never required again.
- Session Prompt: User is prompted once per session.
- One-Shot Prompt: User is prompted each time the restricted API is invoked.

Another way to manage privileges smartly could be to define optional privileges: if at run time, the privilege is not granted, the application should behave gently and re-asking for the privilege later or use another mean to achieve its goals. Finally, at run time, the system will ensure that a given application only uses services for which access privileges were claimed and granted. Depending on the user settings at installation time, some dynamic privileges grants or revocations may occur at run time.

2. Privilege: definition

2.1. Format

A privilege can be described and identified by any unique characters string.

However, it's probably a good habit to use a common notation that would help to easily recognize the privilege, its origin and the context ("profile") in which it applies.

In this perspective, we could start by adopting one of the following formats:

- URN style (preferred)

  urn:<vendor>:**privilege**:<profile>:<name>[:<operation>]

- URL style (close to Tizen notation):

  http://<vendor>**/privilege/**<profile>**/**<name>[**.**<operation>]

With these notations, we could define some privileges such as:

- `urn:automotivelinux.org:privilege:common:alarm:set`

  used in any AGL ECU to control access to alarm settings

- `urn:automotivelinux.org:privilege:telematics:diagnostics:send`

  used in an ECU with "Telematics" profile

- `urn:renesas.com:privilege:porter_ext:imu`

  would apply to Porter board with Extension board to control access to the IMU chip

- `urn:toyota.com:privilege:hybrid:energy:manage`

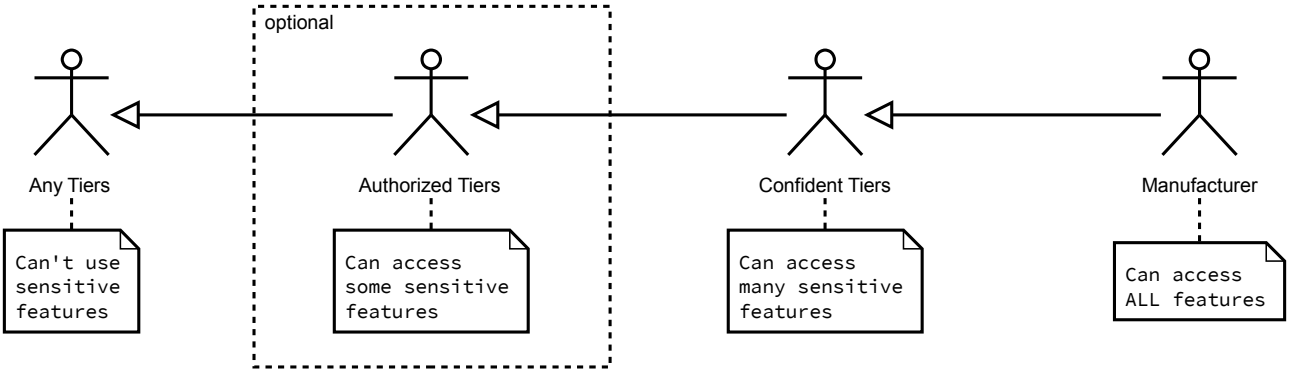
  is an example of privilege defined by an OEM for some product range

The list of privileges on AGL is not determined at the time of writing, but the privileges defined on different embedded systems may be used as a starting point: AGL could probably reuse or share the same privileges separation to protect base resources (see §5 for an exhaustive list).

2.2. Levels

Every installed application package should contain valid signatures[3]. The signatures mainly identify the author of the package and may also contain signatures of distributors. These signatures can then be used to set some privileges only available to some identified authors who have a required "level".

Seen differently, privileges levels are a way to introduce a hierarchy: a privilege with a given level can only be granted to an application at installation time if the origin of the package/application is known to have the required level.



For example, Tizen defines three levels of privileges: platform, partner, public:

- "platform" is the highest one. An application signed with platform's signature can use any defined privilege. For example, the privilege "KEYGRAB" is a platform privilege.
- "partner" is an intermediary that may have some form of extended privileges, but with a reduced scope compared to "platform".
- "public" is the lowest one. An application of this level is never authorized to use the privileges that are of higher levels.

On AGL, we could have a similar hierarchy:

- "vendor": highest level, reserved for OEM privileged applications
- "tier1": medium level, reserved for Tier1 privileges
- "partner": low level, reserved for other partners
- "public": base level

3 http://www.w3.org/TR/widgets-digsig/

2.3. Privileges Hierarchies/Dependencies

On one side, the number of static privileges defined for a system may be important due to the number of different domains and the granularity of protected operations (see in §5: ~100 for Tizen 2.x, ~130 for Android).

On the other side, having a growing number of security rules becomes a security hole due to complexity and to the fact that end-users won't take the time to review long privileges lists and will grant them more or less blindly to applications. This is also a hassle for developers as maintaining a list of privileges for their applications may become a nightmare.

Given these facts, it may be interesting to introduce a hierarchy in the privileges, instead of running with a flat-organized model. For example:

- Grouping privileges would allow to enable/disable Bluetooth globally as well as authorize/deny a single feature or profile.
- Introducing a hierarchy within privileges would enable an application with a "write access" to phone directory to automatically inherit the "read access" without having to request it.
- Introducing abstractions: using geolocation API would mean getting access to GPS data or UMTS Network geolocation

This brings significant advantages:

- from the developer point-of-view, it may be easier to handle when writing the application configuration file
- it can create an abstraction which is very interesting to handle heterogeneous systems on the long run: the privileges hierarchy could evolve over time, without changing the meta-privileges declared by applications.

Note: this topic is an on-going work and still deserves more studies on ins and outs.

2.4. Relationship with APIs

As privileges are defined to protect APIs, it's obvious that we should define for each privilege the APIs/verbs which are protected.

So the general relationship could be expressed by:

1 privilege can protect *n* verbs in *m* APIs (*n*>=1 , *m*>=1)

This is very versatile, but could lead to great complexity:

- some verbs may be protected by multiple privileges
- the number of privileges could be huge

So there's a need for keeping things simple and keep the complexity as low as possible. For this, we propose to restrict the relationship as much as possible to:

1 privilege can protect *n* verbs in 1 API (*n*>=1)

There are many services where API methods/verbs are naturally partitioned in three categories:

- methods for reading/enumerating
- methods for writing/storing
- methods for management/settings

As a consequence, an easy way to protect a service API is to declare a reduced set of privileges where each privilege protects a part of the API without overlapping on others. We could estimate that the number of privileges would be roughly:

(number of services) x 3

Defining a small number of privileges per API, following a known scheme, allows to:

- keep the list of privileges as small as possible
- lower the complexity of privileges management for developers
- present things clearly when it comes to asking permissions to end users

3. Cynara

Cynara is a fast, simple and safe policy-checker service.

It was initially developed for Tizen 3.x where the requirements are close to those of AGL: in both cases, services need to control if callers have sufficient privileges to make some API function calls. Cynara records 4-uples [user, application, privilege, authorisation] and also manages some aspects related to the user session.

The main goal for Cynara is to provide a fast implementation of the following functionalities:

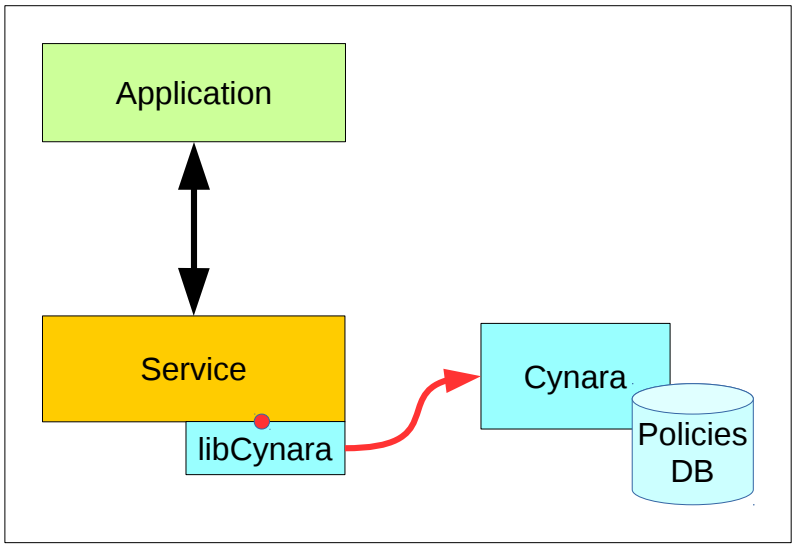
- checking access for some privileges
- holding policies
- simple, single function API when checking policies
- thin client library (to make access control even more simple)
- ability to use external agents (in case of policies that can't be fully processed in Cynara and should be delegated to plugins)

The service that implements the set of actions related to a privilege must check Cynara to know if it can serve its client (if the client has the required privilege). The call to Cynara for checking if a privilege is granted can’t be made by the client. It must be made by the service because believing the client is unsafe. Thus the checking can not be made in client library. The service that receives a request from a client via UDS (Unix Domain Socket) has access in a secure and reliable way to:

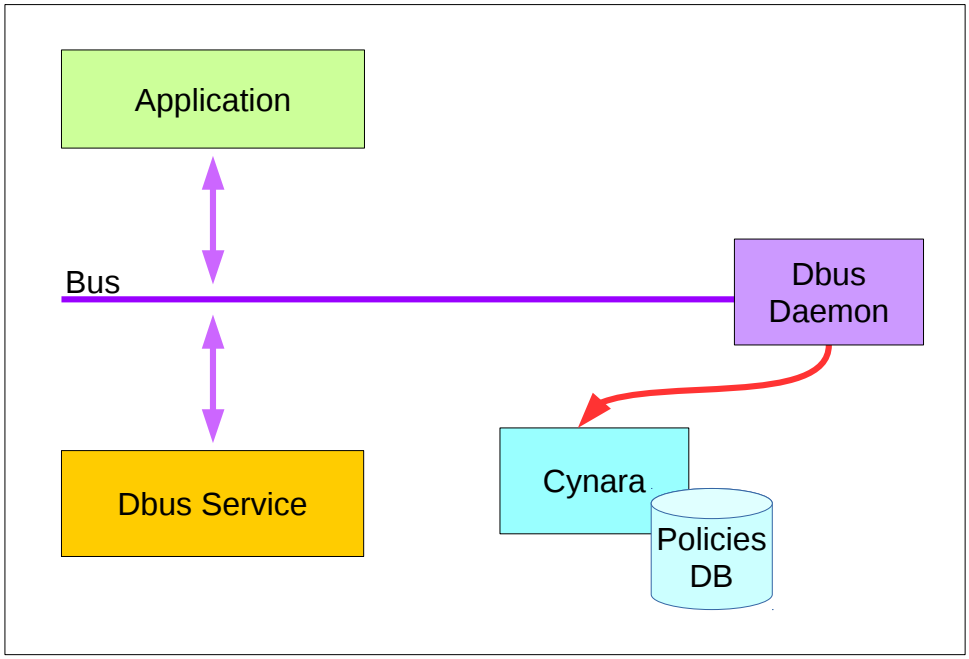
- the user id and group id (UID/GID) running the application
- the process id (PID) of the client
- the SMACK label of the client – as the application framework daemon starts applications with a dedicated SMACK label, this label also identifies the application

These attributes can then be used for queries to Cynara (and also by the service itself for other checks as well).

The following diagram gives an overview of how Cynara can be used by services to check privileges:



DBus was also patched to seamlessly call Cynara and transparently check privileges in the case of a Dbus service. This model is interesting because it allows the integration of services without any modification:



In reality, it isn't as trivial because Cynara may interact with the end user to collect his consent. In such cases, DBus message must be deferred in a queue while waiting for end user answer: user consent may conflict with default DBus time-out.

More information on Cynara can be found on the Tizen Wiki:

https://wiki.tizen.org/wiki/Category:Cynara

4. Developing applications with privileges

When developers create an application, they describe the requirements of this application with a file named "config.xml". The format and usage of this description file is part of the documentation about how to write a widget package for AGL. This documentation can be found in the public internet folder http://iot.bzh/download/public/2016/appfw/

This configuration file, in its current version, allows:

- to request the permissions needed using the feature of name **urn:AGL:required-permission**
- to declare the permissions that will be checked by the service (if a service is installed) using the feature of name **urn:AGL:defined-permission**

The following extract is an example of what could be a "config.xml" file:

```
<widget id="myplayer" version="1.1">

     ...

     <feature name="urn:AGL:required-permission">
          <param name="urn:AGL:permission:multimedia" value="required"/>
          <param name="urn:AGL:permission:bluetooth:read" value="optional"/>
          <param name="urn:AGL:permission:internet" value="optional"/>
          <param name="urn:AGL:permission:publicapi" value="optional"/>
     </feature>

     <feature name="urn:AGL:defined-permission">
          <param name="urn:myplayer:permission:play" value="public"/>
     </feature>

     ...

</widget>
```

This extract describes that the installed application strongly requires the permission defined by AGL to access multimedia items and optionally requires permissions to access bluetooth and internet to be accessed by other applications.

It also declares that the permission urn:myplayer:permission:play might be necessary to access features provided to other applications. This permission is public and then not restricted to be granted by any distributor or authority.

5. Similar mechanisms on other systems

5.1. Tizen

Some links about security and privileges on Tizen:

- https://wiki.tizen.org/wiki/Security
- https://wiki.tizen.org/wiki/Category:Security

Tizen 2.4 documentation[4] defines the following 55 privileges for Web Applications (Mobile profile):

Tizen: Mobile Web Device API privileges			
Privilege	**Level**	**Display name**	**Description**
`http://tizen.org/privilege/account.read`	public	Reading accounts	The application can read accounts.
`http://tizen.org/privilege/account.write`	public	Managing accounts	The application can create, edit, and delete accounts.
`http://tizen.org/privilege/alarm`	public	Setting alarms	The application can set alarms and wake up the device at scheduled times.
`http://tizen.org/privilege/application.info`	public	Retrieving application information	The application can retrieve information related to other applications.
`http://tizen.org/privilege/application.launch`	public	Opening applications	The application can open other applications

4 Tizen privileges documentation
https://developer.tizen.org/development/getting-started/native-application/understanding-tizen-programming/security-and-api-privileges

Tizen: Mobile Web Device API privileges			
Privilege	**Level**	**Display name**	**Description**
			using the application ID or application control.
http://tizen.org/privilege/appmanager.certificate	partner	Getting application certificates	The application can retrieve specified application certificates.
http://tizen.org/privilege/appmanager.kill	partner	Closing applications	The application can close other applications.
http://tizen.org/privilege/bluetooth	public	Using unrestricted Bluetooth services	The application can perform unrestricted actions using Bluetooth, such as scanning for and connecting to other devices.
http://tizen.org/privilege/bluetoothmanager	platform	Managing Bluetooth system settings	The application can change Bluetooth system settings related to privacy and security, such as the visibility mode.
http://tizen.org/privilege/bookmark.read	platform	Reading bookmarks	The application can read bookmarks.
http://tizen.org/privilege/bookmark.write	platform	Managing bookmarks	The application can create, edit, and delete bookmarks.
http://tizen.org/privilege/calendar.read	public	Reading calendar	The application can read events and tasks.
http://tizen.org/privilege/calendar.write	public	Managing calendar	The application can create, update, and delete events and tasks.
http://tizen.org/privilege/call	public	Making phone calls	The application can make phone calls to numbers when they are tapped without further confirmation.
http://tizen.org/privilege/callhistory.read	public	Reading call logs	The application can read call log items.
http://tizen.org/privilege/callhistory.write	public	Managing call logs	The application can create, update, and delete call log items.
http://tizen.org/privilege/contact.read	public	Reading contacts	The application can read your profile, contacts, and contact history. Contact history can include social network activity.

Tizen: Mobile Web Device API privileges			
Privilege	**Level**	**Display name**	**Description**
`http://tizen.org/privilege/contact.write`	public	Managing contacts	The application can create, update, and delete your profile, contacts, and any contact history that is related to this application. Contact history can include social network activity.
`http://tizen.org/privilege/content.read`	public	Reading contents	The application can read media content information.
`http://tizen.org/privilege/content.write`	public	Managing contents	The application can create, update, and delete media content information.
`http://tizen.org/privilege/datacontrol.consumer`	public	Accessing exported data	The application can read data exported by data control providers.
`http://tizen.org/privilege/datasync`	public	Syncing device data	The application can synchronize device data, such as contacts and calendar events, using the OMA DS 1.2 protocol.
`http://tizen.org/privilege/download`	public	Downloading through HTTP	The application can manage HTTP downloads.
`http://tizen.org/privilege/filesystem.read`	public	Reading file systems	The application can read file systems.
`http://tizen.org/privilege/filesystem.write`	public	Writing to file systems	The application can write to file systems.
`http://tizen.org/privilege/healthinfo`	public	Reading health related information	The application can read the user's health information gathered by device sensors, such as pedometer or heart rate monitor.
`http://tizen.org/privilege/ime`	public	Providing input methods	The application can provide users with a way to enter characters and symbols into an associated text field.
`http://tizen.org/privilege/led`	public	Managing LEDs	The application can turn LEDs on or off, such as the LED on the front of the device and the camera flash.

Tizen: Mobile Web Device API privileges			
Privilege	**Level**	**Display name**	**Description**
`http://tizen.org/privilege/location`	public	Using user location	The application can read the user's location information.
`http://tizen.org/privilege/mediacontroller.client`	public	Controlling media player	The application can receive information about currently playing media from applications that are allowed to send it, and can control those applications remotely.
`http://tizen.org/privilege/mediacontroller.server`	public	Accepting remote controls	The application can send information about currently playing media to applications that are allowed to receive it, and can be controlled remotely by those applications.
`http://tizen.org/privilege/messaging.read`	public	Accessing messages	The application can retrieve messages from message boxes or receive messages.
`http://tizen.org/privilege/messaging.write`	public	Writing messages	The application can write, send, sync, and remove text messages, multimedia messages, and emails.
`http://tizen.org/privilege/networkbearerselection`	partner	Selecting network connection	The application can request and release a specific network connection.
`http://tizen.org/privilege/nfc.admin`	public	Managing NFC general settings	The application can change NFC settings, such as switching NFC on or off.
`http://tizen.org/privilege/nfc.cardemulation`	public	Using NFC card emulation mode	The application can access smart card details, such as credit card details, and allow users to make payments through NFC.
`http://tizen.org/privilege/nfc.common`	public	Using NFC common features	The application can use common NFC features.
`http://tizen.org/privilege/nfc.p2p`	public	Pushing NFC messages	The application can push NFC messages to other devices.

Tizen: Mobile Web Device API privileges			
Privilege	**Level**	**Display name**	**Description**
`http://tizen.org/privilege/nfc.tag`	public	Reading/writing to NFC tags	The application can read and write NFC tag information.
`http://tizen.org/privilege/notification`	public	Providing notifications	The application can show and hide its own notifications and badges.
`http://tizen.org/privilege/package.info`	public	Receiving package information	The application can retrieve information about installed packages.
`http://tizen.org/privilege/packagemanager.install`	platform	Managing packages	The application can install or uninstall application packages.
`http://tizen.org/privilege/power`	public	Managing power	The application can control power-related settings, such as dimming the screen.
`http://tizen.org/privilege/push`	public	Receiving push notifications	The application can receive notifications from the Internet.
`http://tizen.org/privilege/secureelement`	public	Accessing secure elements	The application can access secure smart card chips, such as UICC/SIM, embedded secure elements, and secure SD cards.
`http://tizen.org/privilege/setting`	public	Accessing user settings	The application can change and read user settings.
`http://tizen.org/privilege/system`	public	Reading system information	The application can read system information.
`http://tizen.org/privilege/telephony`	public	Accessing telephony information	The application can retrieve telephony information, such as the network and SIM card used, the IMEI, and the statuses of calls.
`http://tizen.org/privilege/volume.set`	public	Adjusting volume	The application can adjust the volume for different features, such as notification alerts, ringtones, and media.

Tizen: Mobile Web W3C/HTML5 API privileges		
Privilege	**Level**	**Description**
`http://tizen.org/privilege/internet`	public	The application can access the Internet using the WebSocket, XMLHttpRequest Level 2, Server-Sent Events, HTML5 Application caches, and Cross-Origin Resource Sharing APIs.
`http://tizen.org/privilege/mediacapture`	public	The application can manipulate streams from cameras and microphones using the getUserMedia API. **Privilege behavior:** • In the local domain, if this privilege is defined, permission is granted. Otherwise, execution is blocked. • In the remote domain, if this privilege is defined, pop-up user prompt is used. Otherwise, execution is blocked.
`http://tizen.org/privilege/unlimitedstorage`	public	The application can use the storage with unlimited size with the File API: Directories and System, File API: Writer, Indexed Database, and Web SQL Database APIs. **Privilege behavior:** • In the local domain, if this privilege is defined, permission is granted. Otherwise, pop-up user prompt is used. • In the remote domain, pop-up user prompt is used.
`http://tizen.org/privilege/notification`	public	The application can display simple notifications using the Web

Tizen: Mobile Web W3C/HTML5 API privileges		
Privilege	**Level**	**Description**
		Notifications API. **Privilege behavior:** • In the local domain, if this privilege is defined, permission is granted. Otherwise, pop-up user prompt is used. • In the remote domain, pop-up user prompt is used.
`http://tizen.org/privilege/location`	public	The application can access geographic locations using the Geolocation API. **Privilege behavior:** • In the local domain, if this privilege is defined, permission is granted. Otherwise, execution is blocked. • In the remote domain, if this privilege is defined, pop-up user prompt is used. Otherwise, execution is blocked.

Tizen: Mobile Web Supplementary API privileges		
Privilege	**Level**	**Description**
`http://tizen.org/privilege/fullscreen`	public	The application can display in the full-screen mode using the FullScreen API - Mozilla API. **Privilege behavior:** • If this privilege is defined, permission is granted without user interaction. Otherwise, permission is granted by user interaction.

5.2. Android

Some links related to Android applications security:

- a bit dated, but still accurate: https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/isec_securing_android_apps.pdf
- developer resource: http://source.android.com/devices/tech/security/index.html

The following table summarizes the permissions defined in Android (API level 24[5]): there are 132 permissions (deprecated ones removed).

Android Privileges	
Privilege	**Description**
ACCESS_CHECKIN_PROPERTIES	Allows read/write access to the "properties" table in the checkin database, to change values that get uploaded.
ACCESS_COARSE_LOCATION	Allows an app to access approximate location.
ACCESS_FINE_LOCATION	Allows an app to access precise location.
ACCESS_LOCATION_EXTRA_COMMANDS	Allows an application to access extra location provider commands.
ACCESS_NETWORK_STATE	Allows applications to access information about networks.
ACCESS_NOTIFICATION_POLICY	Marker permission for applications that wish to access notification policy.
ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks.
ACCOUNT_MANAGER	Allows applications to call into AccountAuthenticators.

5 https://developer.android.com/reference/android/Manifest.permission.html

Android Privileges	
Privilege	**Description**
ADD_VOICEMAIL	Allows an application to add voicemails into the system.
BATTERY_STATS	Allows an application to collect battery statistics
BIND_ACCESSIBILITY_SERVICE	Must be required by an AccessibilityService, to ensure that only the system can bind to it.
BIND_APPWIDGET	Allows an application to tell the AppWidget service which application can access AppWidget's data.
BIND_CARRIER_SERVICES	The system process that is allowed to bind to services in carrier apps will have this permission.
BIND_CHOOSER_TARGET_SERVICE	Must be required by a ChooserTargetService, to ensure that only the system can bind to it.
BIND_CONDITION_PROVIDER_SERVICE	Must be required by a ConditionProviderService, to ensure that only the system can bind to it.
BIND_DEVICE_ADMIN	Must be required by device administration receiver, to ensure that only the system can interact with it.
BIND_DREAM_SERVICE	Must be required by an DreamService, to ensure that only the system can bind to it.
BIND_INCALL_SERVICE	Must be required by a InCallService, to ensure that only the system can bind to it.
BIND_INPUT_METHOD	Must be required by an InputMethodService, to ensure that only the system can bind to it.
BIND_MIDI_DEVICE_SERVICE	Must be required by an MidiDeviceService, to ensure that only the system can bind to it.
BIND_NFC_SERVICE	Must be required by a HostApduService or OffHostApduService to ensure that only the system can bind to it.
BIND_NOTIFICATION_LISTENER_SERVICE	Must be required by an NotificationListenerService, to ensure that only the system can bind to it.
BIND_PRINT_SERVICE	Must be required by a PrintService, to ensure that only the system can bind to it.
BIND_QUICK_SETTINGS_TILE	Allows an application to bind to third party quick settings tiles.
BIND_REMOTEVIEWS	Must be required by a RemoteViewsService, to ensure that only the system can bind to it.

Android Privileges	
Privilege	**Description**
BIND_SCREENING_SERVICE	Must be required by a CallScreeningService, to ensure that only the system can bind to it.
BIND_TELECOM_CONNECTION_SERVICE	Must be required by a ConnectionService, to ensure that only the system can bind to it.
BIND_TEXT_SERVICE	Must be required by a TextService (e.g.
BIND_TV_INPUT	Must be required by a TvInputService to ensure that only the system can bind to it.
BIND_VOICE_INTERACTION	Must be required by a VoiceInteractionService, to ensure that only the system can bind to it.
BIND_VPN_SERVICE	Must be required by a VpnService, to ensure that only the system can bind to it.
BIND_VR_LISTENER_SERVICE	Must be required by an VrListenerService, to ensure that only the system can bind to it.
BIND_WALLPAPER	Must be required by a WallpaperService, to ensure that only the system can bind to it.
BLUETOOTH	Allows applications to connect to paired bluetooth devices.
BLUETOOTH_ADMIN	Allows applications to discover and pair bluetooth devices.
BLUETOOTH_PRIVILEGED	Allows applications to pair bluetooth devices without user interaction, and to allow or disallow phonebook access or message access.
BODY_SENSORS	Allows an application to access data from sensors that the user uses to measure what is happening inside his/her body, such as heart rate.
BROADCAST_PACKAGE_REMOVED	Allows an application to broadcast a notification that an application package has been removed.
BROADCAST_SMS	Allows an application to broadcast an SMS receipt notification.
BROADCAST_STICKY	Allows an application to broadcast sticky intents.
BROADCAST_WAP_PUSH	Allows an application to broadcast a WAP PUSH receipt notification.
CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
CALL_PRIVILEGED	Allows an application to call any phone number, including emergency numbers, without going

Android Privileges	
Privilege	**Description**
	through the Dialer user interface for the user to confirm the call being placed.
CAMERA	Required to be able to access the camera device.
CAPTURE_AUDIO_OUTPUT	Allows an application to capture audio output.
CAPTURE_SECURE_VIDEO_OUTPUT	Allows an application to capture secure video output.
CAPTURE_VIDEO_OUTPUT	Allows an application to capture video output.
CHANGE_COMPONENT_ENABLED_STATE	Allows an application to change whether an application component (other than its own) is enabled or not.
CHANGE_CONFIGURATION	Allows an application to modify the current configuration, such as locale.
CHANGE_NETWORK_STATE	Allows applications to change network connectivity state.
CHANGE_WIFI_MULTICAST_STATE	Allows applications to enter Wi-Fi Multicast mode.
CHANGE_WIFI_STATE	Allows applications to change Wi-Fi connectivity state.
CLEAR_APP_CACHE	Allows an application to clear the caches of all installed applications on the device.
CONTROL_LOCATION_UPDATES	Allows enabling/disabling location update notifications from the radio.
DELETE_CACHE_FILES	Allows an application to delete cache files.
DELETE_PACKAGES	Allows an application to delete packages.
DIAGNOSTIC	Allows applications to RW to diagnostic resources.
DISABLE_KEYGUARD	Allows applications to disable the keyguard if it is not secure.
DUMP	Allows an application to retrieve state dump information from system services.
EXPAND_STATUS_BAR	Allows an application to expand or collapse the status bar.
FACTORY_TEST	Run as a manufacturer test application, running as the root user.
GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service.

Android Privileges	
Privilege	**Description**
GET_ACCOUNTS_PRIVILEGED	Allows access to the list of accounts in the Accounts Service.
GET_PACKAGE_SIZE	Allows an application to find out the space used by any package.
GLOBAL_SEARCH	This permission can be used on content providers to allow the global search system to access their data.
INSTALL_LOCATION_PROVIDER	Allows an application to install a location provider into the Location Manager.
INSTALL_PACKAGES	Allows an application to install packages.
INSTALL_SHORTCUT	Allows an application to install a shortcut in Launcher.
INTERNET	Allows applications to open network sockets.
KILL_BACKGROUND_PROCESSES	Allows an application to call killBackgroundProcesses(String).
LOCATION_HARDWARE	Allows an application to use location features in hardware, such as the geofencing api.
MANAGE_DOCUMENTS	Allows an application to manage access to documents, usually as part of a document picker.
MASTER_CLEAR	Not for use by third-party applications.
MEDIA_CONTENT_CONTROL	Allows an application to know what content is playing and control its playback.
MODIFY_AUDIO_SETTINGS	Allows an application to modify global audio settings.
MODIFY_PHONE_STATE	Allows modification of the telephony state - power on, mmi, etc.
MOUNT_FORMAT_FILESYSTEMS	Allows formatting file systems for removable storage.
MOUNT_UNMOUNT_FILESYSTEMS	Allows mounting and unmounting file systems for removable storage.
NFC	Allows applications to perform I/O operations over NFC.
PACKAGE_USAGE_STATS	Allows an application to collect component usage statistics. Declaring the permission implies intention to use the API and the user of the device can grant permission through the Settings application.
PROCESS_OUTGOING_CALLS	Allows an application to see the number being dialed during an outgoing call with the option

Android Privileges	
Privilege	**Description**
	to redirect the call to a different number or abort the call altogether.
READ_CALENDAR	Allows an application to read the user's calendar data.
READ_CALL_LOG	Allows an application to read the user's call log.
READ_CONTACTS	Allows an application to read the user's contacts data.
READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
READ_FRAME_BUFFER	Allows an application to take screen shots and more generally get access to the frame buffer data.
READ_LOGS	Allows an application to read the low-level system log files.
READ_PHONE_STATE	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any PhoneAccounts registered on the device.
READ_SMS	Allows an application to read SMS messages.
READ_SYNC_SETTINGS	Allows applications to read the sync settings.
READ_SYNC_STATS	Allows applications to read the sync stats.
READ_VOICEMAIL	Allows an application to read voicemails in the system.
REBOOT	Required to be able to reboot the device.
RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting.
RECEIVE_MMS	Allows an application to monitor incoming MMS messages.
RECEIVE_SMS	Allows an application to receive SMS messages.
RECEIVE_WAP_PUSH	Allows an application to receive WAP push messages.
RECORD_AUDIO	Allows an application to record audio.

Android Privileges	
Privilege	**Description**
REORDER_TASKS	Allows an application to change the Z-order of tasks.
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	Permission an application must hold in order to use ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS.
REQUEST_INSTALL_PACKAGES	Allows an application to request installing packages.
SEND_RESPOND_VIA_MESSAGE	Allows an application (Phone) to send a request to other applications to handle the respond-via-message action during incoming calls.
SEND_SMS	Allows an application to send SMS messages.
SET_ALARM	Allows an application to broadcast an Intent to set an alarm for the user.
SET_ALWAYS_FINISH	Allows an application to control whether activities are immediately finished when put in the background.
SET_ANIMATION_SCALE	Modify the global animation scaling factor.
SET_DEBUG_APP	Configure an application for debugging.
SET_PROCESS_LIMIT	Allows an application to set the maximum number of (not needed) application processes that can be running.
SET_TIME	Allows applications to set the system time.
SET_TIME_ZONE	Allows applications to set the system time zone.
SET_WALLPAPER	Allows applications to set the wallpaper.
SET_WALLPAPER_HINTS	Allows applications to set the wallpaper hints.
SIGNAL_PERSISTENT_PROCESSES	Allow an application to request that a signal be sent to all persistent processes.
STATUS_BAR	Allows an application to open, close, or disable the status bar and its icons.
SYSTEM_ALERT_WINDOW	Allows an app to create windows using the type TYPE_SYSTEM_ALERT, shown on top of all other apps.

Android Privileges	
Privilege	**Description**
TRANSMIT_IR	Allows using the device's IR transmitter, if available.
UNINSTALL_SHORTCUT	Allows an application to uninstall a shortcut in Launcher.
UPDATE_DEVICE_STATS	Allows an application to update device statistics.
USE_FINGERPRINT	Allows an app to use fingerprint hardware.
USE_SIP	Allows an application to use SIP service.
VIBRATE	Allows access to the vibrator.
WAKE_LOCK	Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming.
WRITE_APN_SETTINGS	Allows applications to write the apn settings.
WRITE_CALENDAR	Allows an application to write the user's calendar data.
WRITE_CALL_LOG	Allows an application to write (but not read) the user's call log data.
WRITE_CONTACTS	Allows an application to write the user's contacts data.
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.
WRITE_GSERVICES	Allows an application to modify the Google service map.
WRITE_SECURE_SETTINGS	Allows an application to read or write the secure system settings.
WRITE_SETTINGS	Allows an application to read or write the system settings.
WRITE_SYNC_SETTINGS	Allows applications to write the sync settings.
WRITE_VOICEMAIL	Allows an application to modify and remove existing voicemails in the system.