Tizen security, lessons learnt

Abstract

The document provides a feedback on lessons learned through the implementation of Tizen security framework in version 2 & 3. Tizen security framework mostly relies on: Smack for rules and Cynara for privileges, completed by a mix of DBus, application manager and security manager for the orchestration.

The document starts with an introduction of Tizen Security Model before presenting lessons learnt from its implementation. Reader interested in further information will find an annex with more details about Tizen security concepts, as well as a few pointers to external references.

Version 1.0

September 2015

Table of contents

Document revisions

Date	Version	Designation	Author
23 sept. 2015	0.1	Proposal	José Bollo
25 sept. 2015	1.0	Initial release	Fulup Ar Foll

1. About Tizen

Tizen is an effort mainly promoted by Samsung and Intel to deliver a Linux distribution for connected devices: cars (IVI), phones, televisions, watches.

During the year 2014, Tizen greatly evolved to address IVI requirements. Since version-3 Tizen is multi-user and multi-seat aware. The security model of Tizen is baked in and fully integrated. It is based on a mix of Trusted boot, Smack, security-manager, Cynara, DBus, framework and Cynara enabled services.

- *Note: when no version is specified this document relates to Tizen-V3.*
- *For further info about Tizen: https://en.wikipedia.org/wiki/Tizen.*

2. Tizen security overview

Tizen distinguishes two aspects of the security: "system security" managed with Smack and "application security", including private user data protection, managed through Tizen Framework.

2.1. System Security

The "System Security" establishes the foundation of any further security aspect. It is controlled by device manufacturers. Multiple objectives: ensure the integrity of the base system, avoid system misuse, provide foundations for applications security.

"System Security" relies on Trusted boot, Smack and DAC (discretionary access control).

To be fully trusted and prevent off-line attacks, the system should include a TPM (Trusted Platform Module) and the IMA+EVM (Integrity Measurement Architecture + Extended Validation Module).

2.2. Application Security

Security of applications applies to: resilience against failure; coercion to respect manifest contract; isolation of application data; respect of the user privacy.

The security of applications is handled through a smart mix of: Smack for sandboxing/isolating applications, DAC for discretionary access control, and a database called Cynara to store application privileges.

The security of applications is implemented by the "Tizen security framework".

"Tizen security Framework" is a key component. It manages installation and removal of applications on the system. It provides a launcher to set-up applications environment in conformance to security specifications.

Tizen/IVI is a challenging system because: it is multi-seat; it has few real-time constraints and it should implement multiple policy rules like: sounds, windows placement, alerts,...

One of Tizen targeted configuration includes 4 displays and 4 zones. It means that 4 different users may interact at the same time on a single system. In that context, the system must guarantee the privacy of each user, but it should also allow interactions and sharing in between users.

The user may face two classes of applications. Applications that are installed by the manufacturer and applications installed later by the user or its children or in worst case by a pirate! The main goal of Tizen security is to provide a safe execution model, that prevents degradation of the system, unwanted modifications or stealing of data.

2.3. The Tizen Security Framework

Beside Smack, the achievement of security for applications relies on following programs: Security-Manager, Cynara, and application launcher. These programs all together compose the "Tizen Security Framework".

Security Framework of Tizen does: install and uninstall package; launch & kill applications; get list and detail status of running applications; and last but not least manage application privileges.

The Cynara database records application's privileges for users. Services may query Cynara to check their clients abilities.

The Security-Manager is called during installation and removal of packages. It computes application's labels. It tags installed files with computed labels. It fills Cynara database for the installed application. For the uninstallation, it reverts the process. The security-manager is also called every time an application is launched.

3. Lessons learnt

3.1. Benefits of using Smack

Smack tries to keep things simple. Smack rules aim to be clear and easy to understand. For these reasons, learning Smack is normally fast and should quickly become natural and intuitive.

Tizen/v3 proposes a ready to go security model, designed by Intel and Samsung security architects under the supervision of Casey Schaufler the author of Smack.

Smack comes with a set of commands and library toolkit, as well Smack enabled packages like “Binutils” and “Linux-Utils”. Some others like “Busybox” do not support Smack, hopefully this last one can easily be replaced by “ToyBox” that implements most of Smack features, only “cp” and “ps” miss and they are on the short term “to do” list. In worst case, when something is missing, original Smack command toolkit can still be used.

When moving from Tizen/v2 to v3, a significant effort to reduce the number of Smack rules happened. Current version of Tizen/IVI holds around 30 rules, 10 of them targeting specifically the automotive message broker. The reduced number of predefined rules makes easy for designers to complement the model while keeping complexity under control.

Finding the right balance between complexity and flexibility remains an open challenge. With 30 rules in total, Tizen/IVI remains easy enough for security team to master the system, this as well during its initial design than for its longer term maintenance or evolution.

3.2. Removing privileges from Smack

In Tizen/v2, privileges were managed using Smack. This led to a fat database of Smack rules. The drawbacks were: slower boot time, unmanageable rules. As an example, a simple phone without much applications had over 30 000 rules.

To keep it simple, the management of privileges should not rely on Mandatory Access Control. This was the main motivation for the introduction of Cynara in Tizen/v3.

3.3. Keep authorisation model simple

Tizen privileges can be granted dynamically when application needs them. This significantly complicates implementations and is not very useful. In fact this should be avoided as much as possible.

The privilege manager should be the only way of changing authorized privileges of applications after installation. It simplifies system design and improves end-user experience.

Also, a distinction has to be made between: a privilege denied because end-user doesn't grant it and a privilege denied because the application did not request it in its manifest. In the latter case, the application should be jailed because it doesn't respect its contract. This distinction is a need for application and system designers to emit clear diagnostics to end-users.

3.4. Importance of grouping privileges

If Tizen/v2 had approximately above 100 privileges, despite a significant simplification effort, Tizen/v3 remains above 70 privileges[1].

From the developer point of view, on one hand the more privileges you have the better it is. More privileges translate in more detailed authorizations and allow targeted objects to appear more clearly. On the other hand maintenance might quickly become a nightmare.

From the end-user point of view, on one hand the more privileges we have, the more privacy protection and secured applications he gets. Conversely, users might be bothered by long list of privileges and would almost certainly start to allow most of them blindly.

One of the drawback of current Tizen privileges structure comes from its flat organization model. Privilege hierarchy could help to provide a better balance in between complexity and flexibility. 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 specifically request it.

3.5. Importance of signatures

Every installed application package should contain valid signatures[2]. The embedded signature should be used to set level of privileges of corresponding applications. 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.

1 The current version of Android defines 135 permissions grouped in 9 items.
2 http://www.w3.org/TR/widgets-digsig/

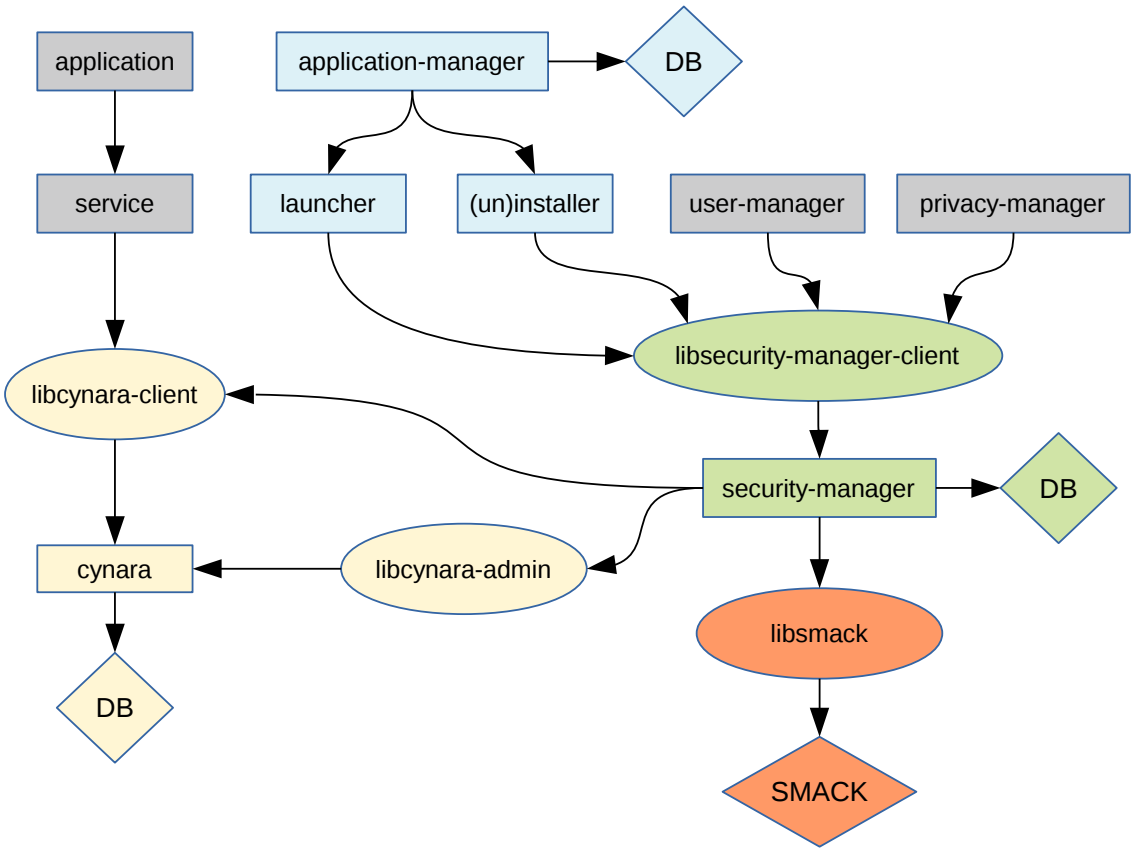
“partner” is an intermediary that may have some form of extended privileges, but with a reduced scope compared to “platform”. Nevertheless even if this level exist under Tizen/v3 it is not used.

“public” is the lowest one. An application of this level is never authorized to use the privileges that are of level platform or partner.

Tizen uses this model for both HTML5 and native applications.

Tizen mechanism or something equivalent to assert the origin and validity of a package and select corresponding level of privileges before its installation is a “MUST have” feature.

Tizen's Security Framework is made of many components: Security-Manager, Cynara, Application-Manager, Privacy-Manager, User-Manager. Here after a picture summarising the architecture.



While this model works, it is far from being perfect. On one hand, we have multiple databases with duplication of information that may eventually raise synchronisation challenges. This problem of duplication of user data is obvious for Security and Application Manager and at a lesser extent is also true for Cynara database. The second issue is the inconsistency of data localisation. While Application Manager data

are handled per user and stored within his home directory, Cynara and Security Manager rely on a central repository. Depending on how a user is removed this may create inconsistency with a user still existing at global database level while he does not exist any more on the system.

The amount of work to merge Cynara, Security-Manager and the Application-Manager is far from being negligible, but it would simplify security management. At least every data related to a given user should be stored in his home directory.

3.6. Security manifest and packaging

Tizen/v2 leverages RPM security plugin to set Smack labels and thus enforces the security of installed packages. Each package had to have a manifest[3] file explaining its own security domain and links to other domains. This model was limited to Smack and was only supported by mainstream RPMv4 but not by RPM5 the default for yocto.

In Tizen/V3, most manifests have been reset and security rules have been transferred to postinstall or runtime. This model provides more flexibility and does not depend on a specific version of RPM packaging system.

It is nevertheless important to note that only platform packages are concerned by this package mechanism, ie: Weston, PulseAudio, Bluez, … On the other hand, applications handled directly by the Tizen "Application-Framework" like: Media-Player, Games,... are not concerned and retrieve their policies directly from Cynara and the Security-Manager.

Keeping security set-up independent from Packaging tool is important. Nevertheless, the concept of a manifest that statically describes the security requirements for concerned package is a very nice feature to have.

3.7. NfTable/Netfilter should be used

Smack uses CIPSO to transmit labels through Internet. CIPSO is not safe and can easily be faked. Furthermore, it is not available on IPv6. As a result, CIPSO does not look a valid route.

Netfilter has existing features to filter Smack tagged packets. Such a feature could potentially be used to configure netfilter tables based on the permission to access internet. Nftables are to be used rather than iptables for their IPV6 implementation.

Despite that such feature exist, Tizen/v3 doesn't implement netfiltering. It might be viewed as a lack and is a potential item for a "to do" list.

3 The reader should be aware that manifest files are linked to RPM through RPM security plugin. It should not be confused with manifest of Tizen's application.

3.8. Content security policy is impossible

The W3C defines the content security policy[4] that rules accesses to internet resources based on their names. Using the name makes the enforcement of policy very difficult because there is no one to one match between the name and the IP address. Also, in the case of HTTPS, the truly requested name is encrypted. Thus, content security policy can not be implemented through Smack or netfilter. This is only valid for a certified program like a web browser.

3.9. Constraint with privileges check inside DBus

DBus was patched to seamlessly call Cynara and transparently check privileges. In the past, Tizen team had also investigated a model based on stubs on top of services but it wasn't held.

On the paper, this model looks good because it allows the integration of services without modification. In reality, it isn't as trivial because Cynara is allowed to 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. First drawbacks: DBus daemon has to be patched to interact with Cynara. Second: user consent may conflict with default DBus time-out. Nevertheless, in the real world, this is not an issue because privileges are set only once at installation time.

Some discussion on this topic happened on DBus mailing list, and Tizen's patch was submitted. Currently the situation is not crystal clear, but most probably proposed Tizen patch will not enter DBus. It should also be check if porting the patch to sd-bus is interesting or not.

So advantage of having a “transparent” solution has to be put in balance with maintaining a patch.

3.10. Containerisation needs more work

During Tizen/v3 studies, a specific version of the application launcher was introduced to start applications within a light container. While container cost from both startup time and resource usage remained acceptable, this facility was finally not introduced in Tizen/v3.

Running applications in a dedicated container has significant security advantages. It becomes possible to implement global security rules at container level, while keeping details of security within the application container itself. During Tizen/v3 test, Cynara was not used and only Smack label have been tested. This model increases moderately the number of Smack labels and keeps complexity acceptable.

4 http://www.w3.org/TR/CSP

As stated before, this solution was not selected for Tizen/v3. Nevertheless the container model remains a valuable solution. To implement such a model in production, it looks important to support Smack rules not only outside of the container but also inside. This would require a new version of `unshare(2)` that supports Smack isolation. Current proposal is to merge Smack namespace with user namespace[5]. Samsung worked on the subject but until now it is not in upstream kernel.

3.11. Elimination of polkit in favor of Cynara

During Tizen/v3 studies and evaluation, we looked at Polkit as a potential replacement for Cynara. Nevertheless this solution appeared far too slow and was quickly abandoned.

On the other hand, despite some limitation in multi-user mode, Cynara's return of experience is good, even if some improvements to smooth existing limitations would be more than welcome.

3.12. Filesystem must have extended attributes

The Smack security relies on the extended attributes. For this reason, the chosen filesystem to use must support the extended attributes.

For NAND or NOR memory, it may be a problem because unfortunately most of well known flash optimised filesystems do not support them[6]. JFFS2 supports extended attributes.

3.13. No initrd

Because Smack needs extended attributes for files, it is not possible to use an init ramdisk that don't support them. This lack comes from cpio format used to serialize ramdisk to file.

While a patch exists and was proposed to add support of extended attributes to initrd/cpio, as of today, the best solution remains to boot directly from the disk.

3.14. Do not under estimate security cost

Implementing security features might require long and hard work. Any non existing security feature should be carefully analysed before the 'go/no-go” decision. The effective benefit in term of: functionality, security and complexity for developer, maintainer and end-users should justify the long and hard work needed to implement it.

5 https://lkml.org/lkml/2015/5/21/299
6 https://en.wikipedia.org/wiki/Flash_file_system

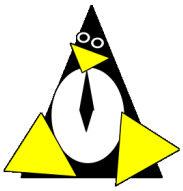
A good example of over-engineering made for Tizen: BlueTooth daemon was patched to improve isolation of user data. The objective was to prevent that an object paired by a given user could be accessible by an other. This requirement broke mainstream implementation that globally shares connections.

The effort required to implement this feature was far too important compared to the benefit, and the equivalent feature for WIFI while started was finally abandoned.

Never forget that a trade off has to be made and that priorities have to be defined.

4. Annexes

4.1. About Smack



Smack stands for Simplified Mandatory Access Control Kernel. It is implemented as an upstream Linux Security Module (LSM) since kernel v2.6.25, nevertheless Tizen's team had to wait until kernel v3.14.20 for Smack to fulfilled Tizen security requirements. Smack like SeLinux or AppAsmor is one of few existing standard frameworks available to operate Linux/LSM.

For Smack, any object has a label. Objects are processes, files, sockets, mapped memory, devices, pipes. Smack is set up with rules describing possible actions. Actions are read, write, append, execute, lock, transmute[7].

A Smack rule is simple. It says what action a given process is allowed/not-allowed on a target object. Here after rule says that any process running with label "`System`" can "read, write, execute" any object with the label "`User`".

```
System      User      rwx
```

Files created by a process are labelled reusing existing process label. An exception is the "transmute" action, where files receive the directory label. The transmutation must be allowed by a rule.

For the details see https://www.kernel.org/doc/Documentation/security/Smack.txt

4.2. Tizen Terminology

Reader should be warned that Tizen redefines few common words. Their meaning within Tizen context may change slightly and could create some confusion. Here after few important definition of Tizen's security terminology.

4.2.1. Package

Tizen/v3 uses the word "package" for a set of software that can be installed on the system. These packages must be distinguished from what is usually called packages by system integrators. Here the word packages doesn't refer to any RPM file (coming from "red hat package manager"). It refers closer to items looking like widget (WGT) files (as documented by http://www.w3.org/TR/widgets).

A package has the following properties useful for the Tizen security framework:

7 Another action called "bringup" exists but is recent and special (dedicated to regulation).

- a signature made by its author;
- an identifier;
- a set of privileges that matches the restricted services needed by the applications of the package.

The signature mainly identifies the author of the packages. It may also contain signatures of distributors. Some distributors signature are needed to unlock some privileges. Based on this model, Tizen implements 3 sets of privileges: platform, partner, public.

The identifier of the package should be unique. It is computed by the Tizen SDK at the time where the package is built. Thus, the probability of a collision of identifier is not null. In that case, the system refuses to install a second package of the same identifier but signed by an other author. Conversely, a package with the same identifier and the same author is an upgrade.

The identifier of the package is used to create two Smack labels for the package. These Smack labels are used for sharing data between the applications of the package either in read-only or in read-write. Example, the identifier `pkgid` **would have labels** `User::Pkg::pkgid` **and** `User::Pkg::pkgid::RO`.

4.2.2. Application

Tizen/v3 uses the word "application" for a piece of software that can be executed. An application can be a native application (written in your favourite language), an HTML5 application (a widget written using HTML+CSS+JAVASCRIPT) or it can be a hybrid application. Thanks to the use of Smack, one of the advantages of Tizen is that you can write an application using, for example, the python language or BASH.

An application has the following properties useful for the Tizen security framework:

- a name that is unique within a package;
- a package from where it comes,

The name of the application mixed with its package identifier is used to create the Smack label for the application. Example, the application of name `app` **of the package of identifier** `pkgid` **would have the label** `User::App::pkgid.app`**.**

4.2.3. Privilege

Any user installing an application[8] expects that application to do what it claims and nothing more. A cautious user will not install a calculator application that either needs to send SMS, to read its address book, or to use internet access.

Tizen/v3 uses the word "privilege" to denote the ability of performing a set of actions. For example, reading mails is an ability subject to privilege.

An application requires through its manifest the privileges corresponding to the ability that it needs. During installation, the installer scans the manifest and asks the user to grant the requested privileges. 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 (this fine tuning is probably specific to Tizen):

- **Blanket Prompt**: User is prompted for confirmation the first time the API function is called by the widget, but once confirmed, prompting is never again required.
- **Session Prompt**: User is prompted once per session.
- **One-Shot Prompt**: User is prompted each time the restricted API is invoked.
- **Permit**: Use of the device capability is always permitted, without asking the user.
- **Deny**: Use of the device capability is always denied.

Technically, a privilege is just a name. Tizen uses uri. Example:

```
http://Tizen.org/privilege/contact.write.
```

The count of privileges is high: approximately 100 for Tizen/v2 and, currently, 73 for Tizen/v3 (see https://wiki.Tizen.org/wiki/Security:Tizen_3.0_Core_Privileges).

The implementation of privileges changed greatly between Tizen/v2 and Tizen/v3. For Tizen/v2 privileges were enforced by Smack rules. This was discontinued because:

- unusable for multi-user where authorisations may differ from a user to an other user because Smack database is common to all users;
- the count of rules implied is very huge[9] and it has negative side effect like increasing boot time;
- Smack acts on resources (files, sockets, processes, memory) not on API nor behaviour so there is a kind of mismatch between what Smack does and what it is used for.

8 Technically the user isn't installing an application but a package that contains it (and that can also contain companion applications, example office suite).

9 For Tizen/v3 IVI, the count of Smack rules is approximatly equal or less than 33+16n where n is the count of applications (not packages that is less). For n=100 applications, system would have 1633 rules. A simple Tizen/v2 device have more rules.

Tizen/v3 enforces the respect of privileges through Cynara. Cynara is a secured database that stores the privileges associated to applications for the users. It records 4-uples [user, application, privilege, authorisation] and also manages the aspects linked to the session.

The service that implements the set of actions linked to a privilege must check Cynara to know if it can serve its client, if the client has the privilege.

For services called through DBus, a patched DBus daemon can check the privileges and ask Cynara for the service. It allows the integration without neither wrapping nor modifying the service.

In some cases, for efficiency or simplicity, the use of a Cynara is replaced by a DAC control of the group of the client. The main example is the use of the camera. In this case, the launcher adds the groups needed for granted privileges.

The other main reason to use a launcher is to enforce the application to run with its Smack label. In the case of HTML5 applications, the running process is always the same but it must run with the label of the application that it interprets. This is the job of the launcher to set that label.

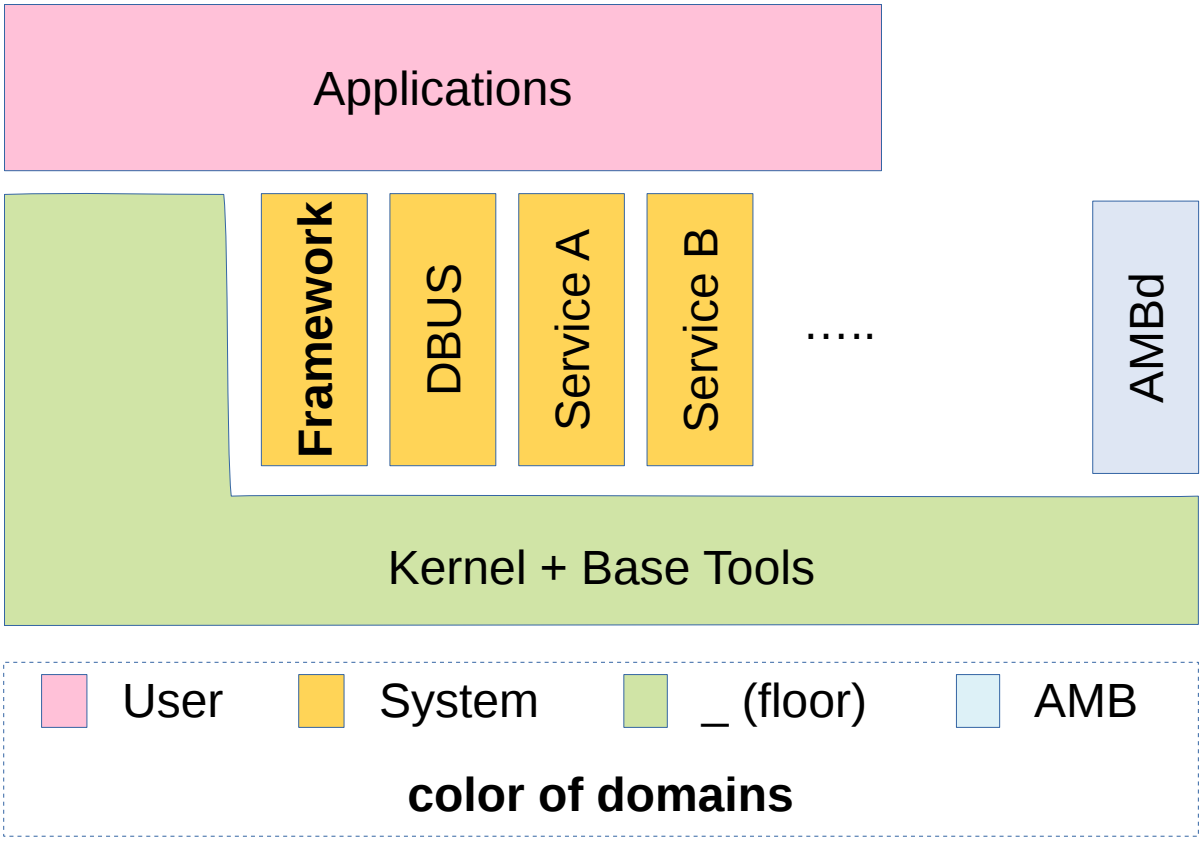
There is a strong implication in using Cynara. The call to Cynara for checking if privilege is allowed can not be made by the client. It must be made by the service because believing the client is unsafe. Thus the checking can not be made in client library. The service that receives a request from its client via UDS (Unix Domain Socket) also receives the Smack label of its client in a sure and reliable way. This received label is used for queries to Cynara.

It have to be known that some privileges are linked to legal issues in some countries.

4.2.4. Domain

Tizen/v3 IVI structures the Smack system security in 4 coarse domains: _ (floor), `System`, `User` and `AMB`. Briefly, they are used, in the order, for the base system, the services, the application and the automotive services.

Having only these 4 domains is too coarse. But these 4 domains make sense, are conceptually well separated and are naturally matching the need. Thus each domain is represented by more than just one label.



_ (floor) is the domain that labels kernel processes and all files and programs being in the foundation of the system. This label allows to all other processes the read and execute accesses but not the write access. It includes the labels _, * and ^.

`System` is the domain for the services. Systemd runs with this label and starts any services with this label unless otherwise specified. It includes the labels `System`, `System::Run`, `System::Shared` and `System::Log`.

`User` is the domain for the services bound to the user and for the applications. It allows to communicate with services of the domain System but it doesn't allow to read their data. It includes the labels `User`, `User::Home`, `User::App::Shared` and other labels of applications and packages.

AMB is dedicated to IVI systems. It includes the labels AMB, AMB::readall, AMB::writeall and AMB::machinegun.

Details here https://wiki.Tizen.org/wiki/Security:SmackThreeDomainModel .

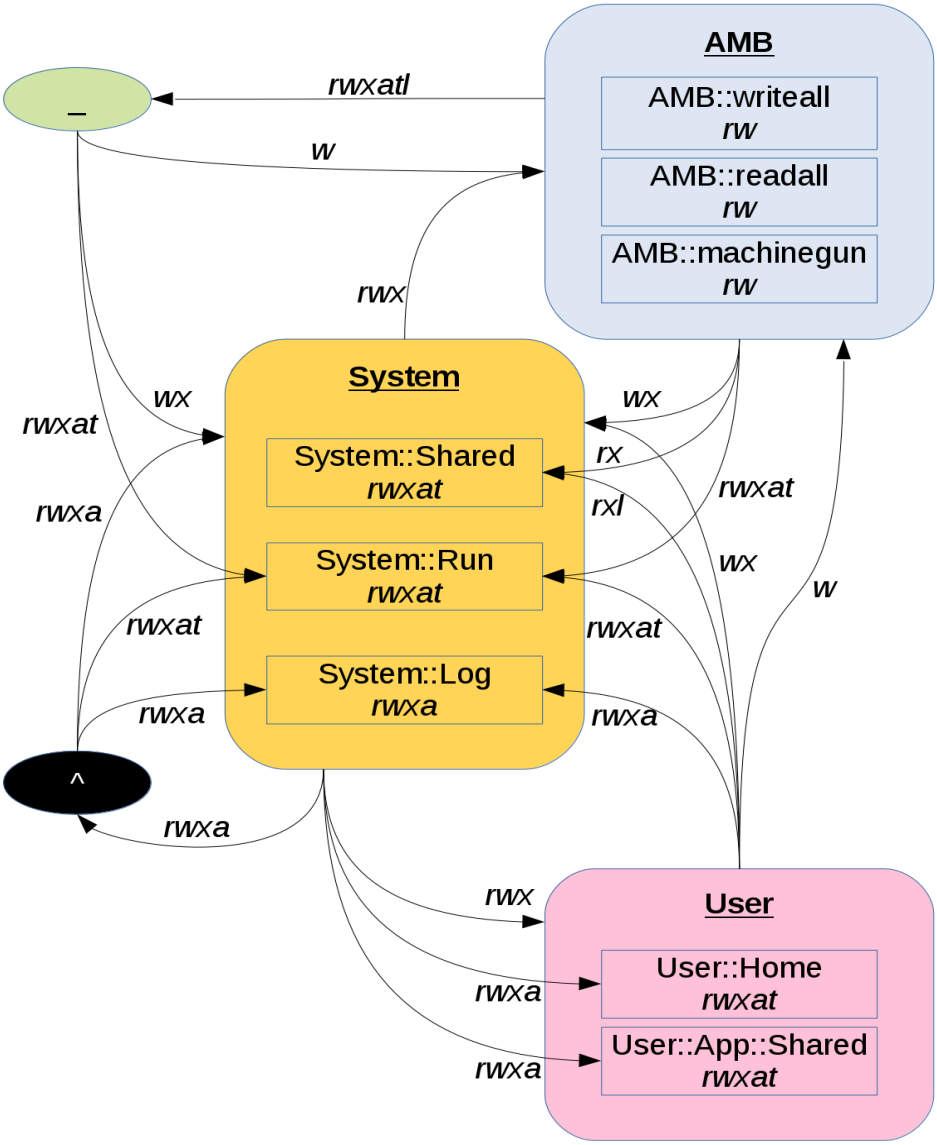
4.3. Picture of Tizen/IVI Smack model

Here are the 29 rules defining the Tizen/IVI policy of the system.

```
AMB      AMB::machinegun     rw
AMB      AMB::readall        rw
AMB      AMB::writeall       rw
AMB      System              wx
AMB      System::Run         rwxat
AMB      System::Shared      rx
AMB      User                w
AMB      _                   rwxatl
System   AMB                 rwx
System   System::Log         rwxa
System   System::Run         rwxat
System   System::Shared      rwxat
System   User                rwx
System   User::App::Shared   rwxa
System   User::Home          rwxa
System   ^                   rwxa
```

```
User     AMB                 w
User     System              wx
User     System::Log         rwxa
User     System::Run         rwxat
User     System::Shared      rxl
User     User::App::Shared   rwxat
User     User::Home          rwxat
^        System              rwxa
^        System::Log         rwxa
^        System::Run         rwxat
_        AMB                 w
_        System              wx
_        System::Run         rwxat
```

These rules are graphically represented below:



5. Links

The main page for Smack is here http://schaufler-ca.com/

A video introducing to Smack and explaining issues of Tizen2: http://dai.ly/x1cliqt its companion slide show is here: https://archive.fosdem.org/2014/schedule/event/Smack/

The main pages for Tizen security on Tizen's wiki are:

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

About CIPSO: https://tools.ietf.org/html/draft-ietf-cipso-ipsecurity-01 and http://csrc.nist.gov/publications/fips/fips188/fips188.pdf and an further reading is available in https://tools.ietf.org/html/rfc7569

Aside research about enforcement of privileges can be found here https://archive.fosdem.org/2015/schedule/event/sec_enforcement/

For android security: http://source.android.com/devices/tech/security/index.html

Widget and their signature: http://www.w3.org/TR/widgets/ and http://www.w3.org/TR/widgets-digsig/

About RPM you have the choice between http://rpm.org/ and http://rpm5.org/ The later being a fork.

The content security policy is http://www.w3.org/TR/CSP2/

The Tizen's privileges (version 2,x) are here https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/security-and-api-privileges and https://developer.tizen.org/development/api-references/web-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.web.apireference/html/web_api_reference.htm

The file systems https://en.wikipedia.org/wiki/Flash_file_system

About UBIFS https://lkml.org/lkml/2015/8/19/483

For extended cpio https://lkml.org/lkml/2015/1/7/679

Smack and namespace https://lwn.net/Articles/623375/ and https://lwn.net/Articles/652320/ and https://lkml.org/lkml/2015/5/21/299