Security Framework for Constraining Application Privileges

Łukasz Wojciechowski
l.wojciechow@partner.samsung.com
Samsung Electronics Poland

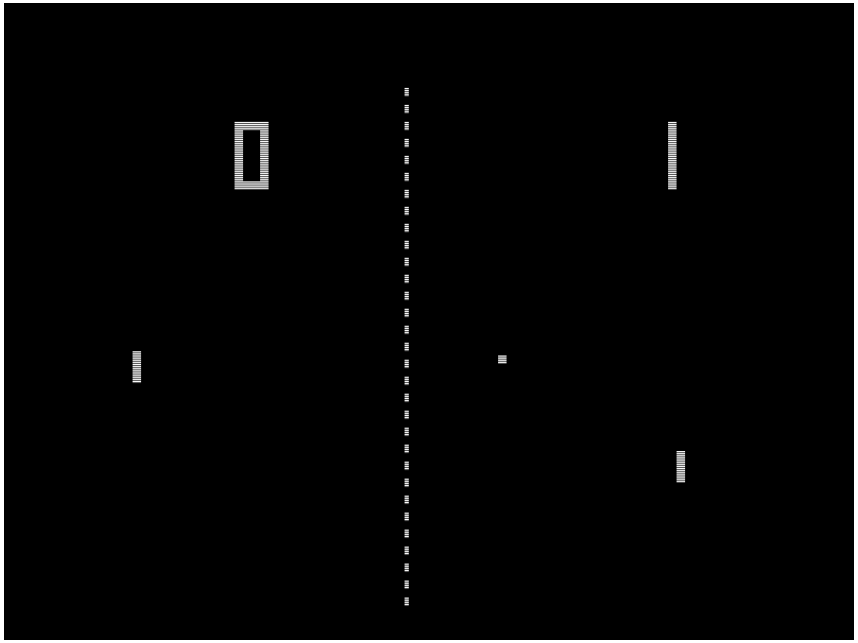
Agenda

- Needs for security
- Idea
- Security Framework
  - Isolation
  - Setup
  - Run
- Non trivial cases
- Auditing
- Containers
- Summary
- Q&A

Needs for security

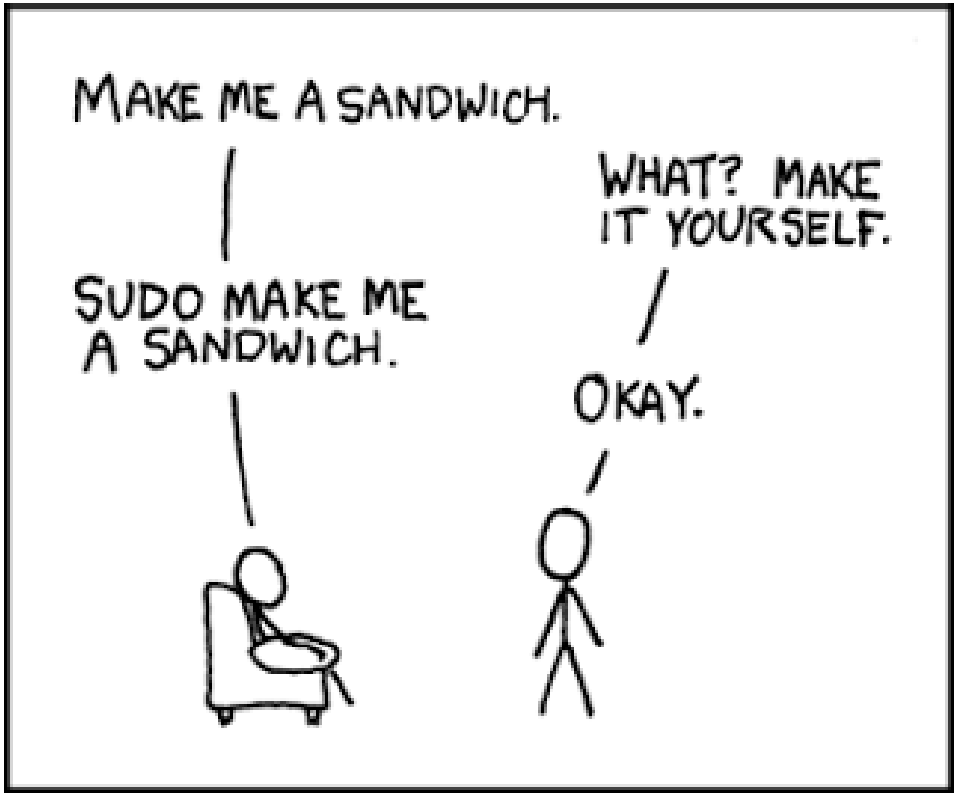
Real life scenario

- Imagine, you've found a super cool game
- Download
- Install
- Run ...



Is there any protection?

- Yes, there is some.
- System resources are not available to common users
- But ...



Idea

Resources and privileges

EMAIL

- Read, send emails
- Manage account

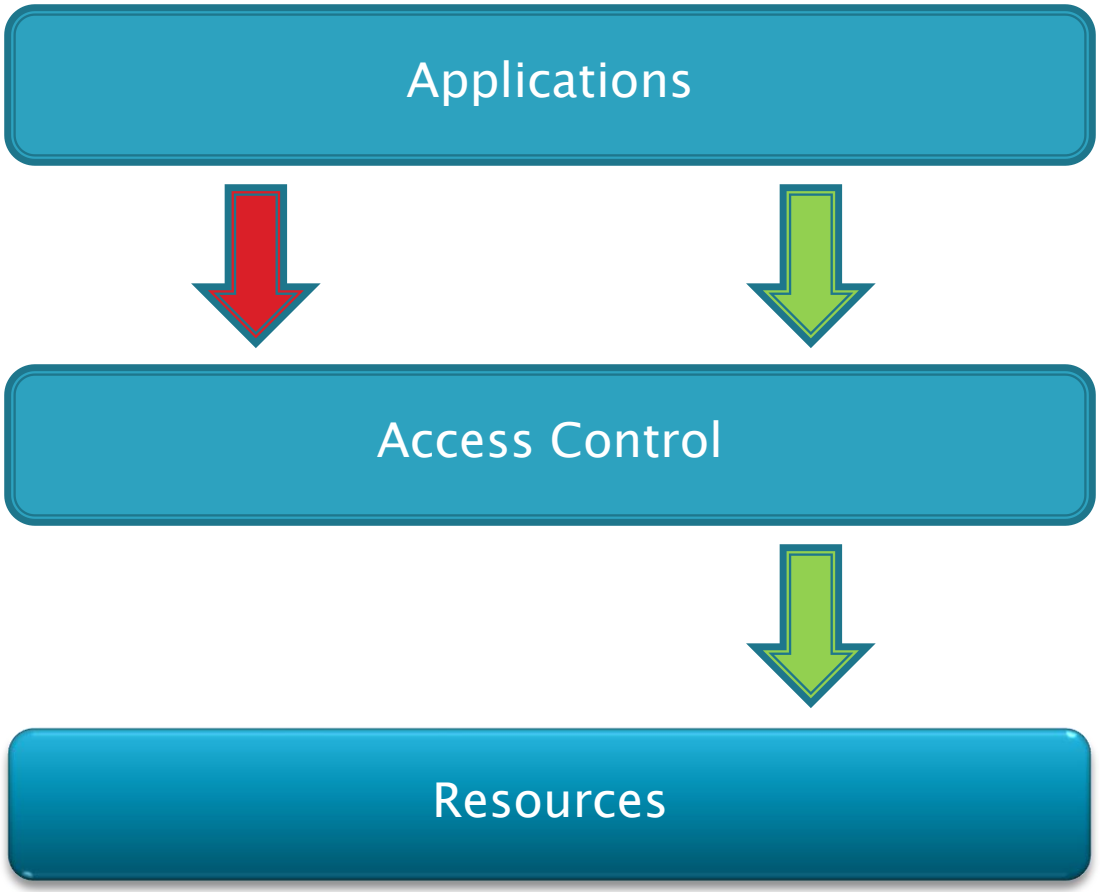
CAMERA

- Take photo
- Record

INTERNET

- Access some IP
- Use protocol

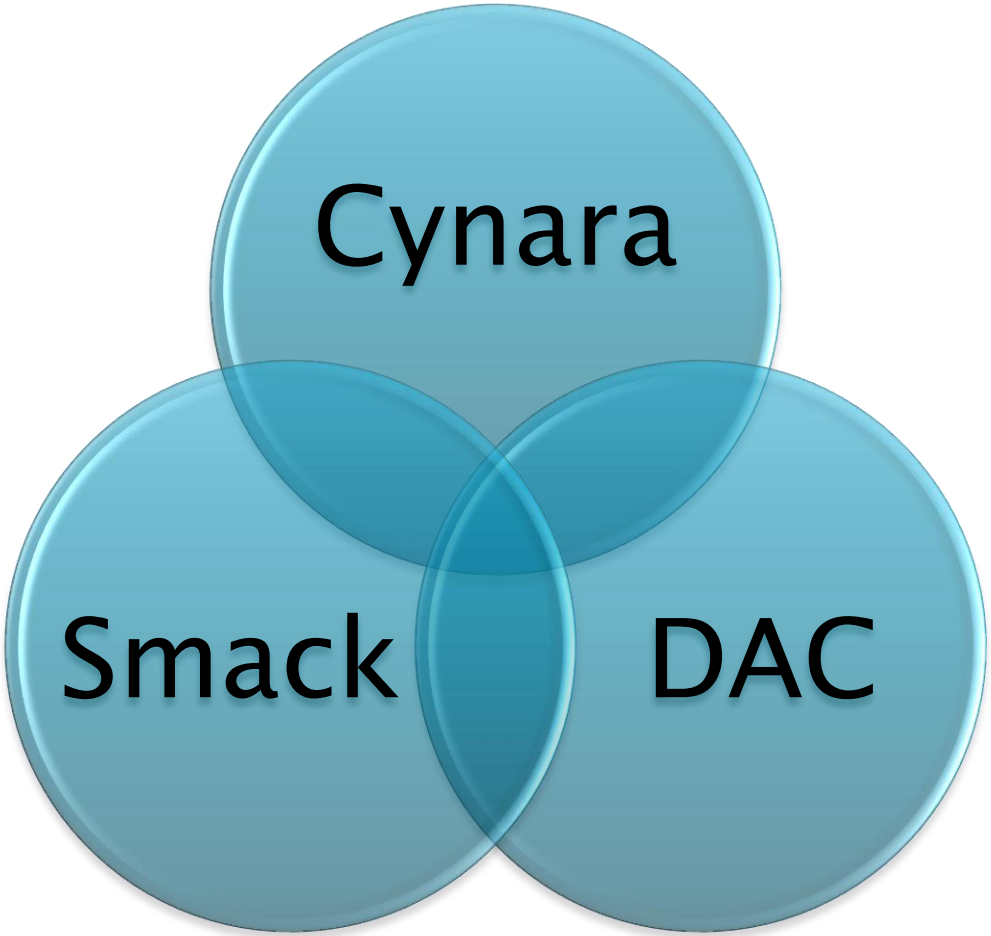
Access control



Security Framework

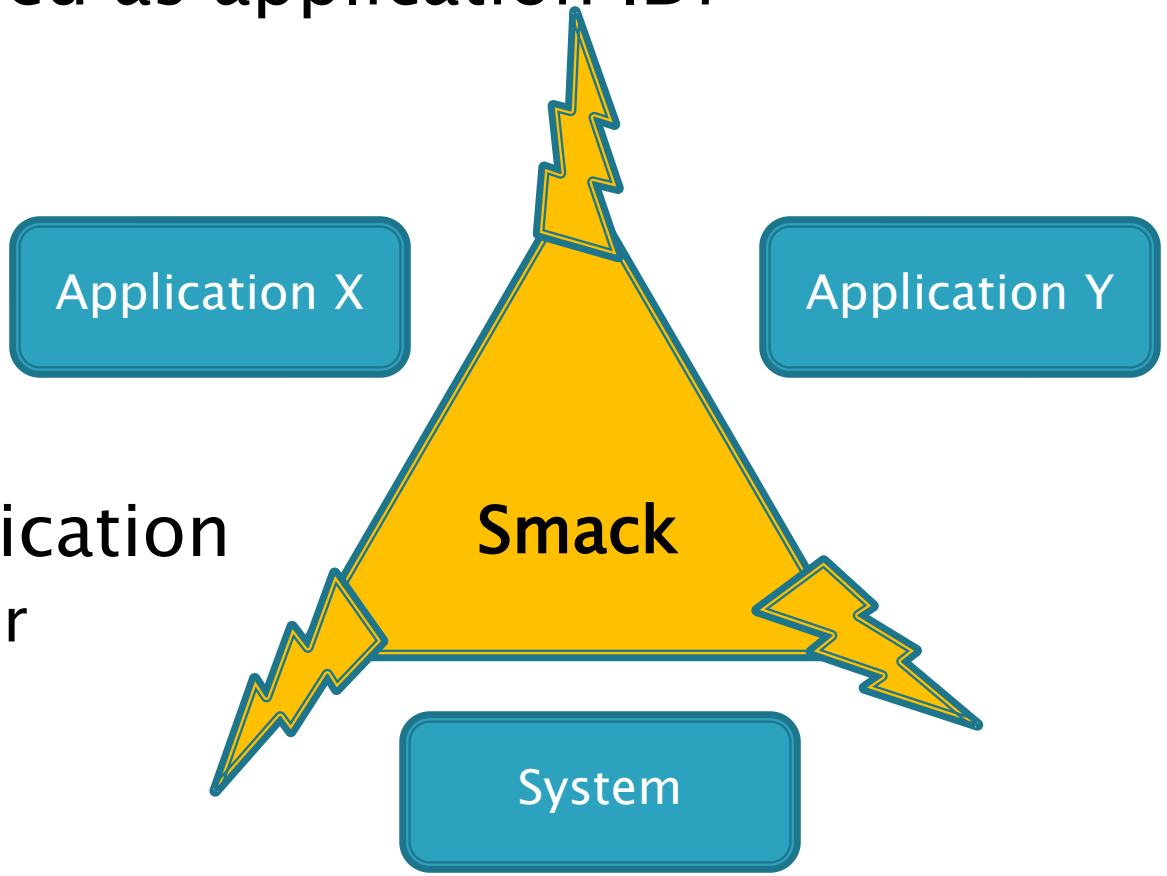
Step 1. Isolation

3 Pillars of Security



Smack

- Smack label used as application ID:
  - Easy to get
  - Unspoofable
- Separates application
  - From each other
  - From system



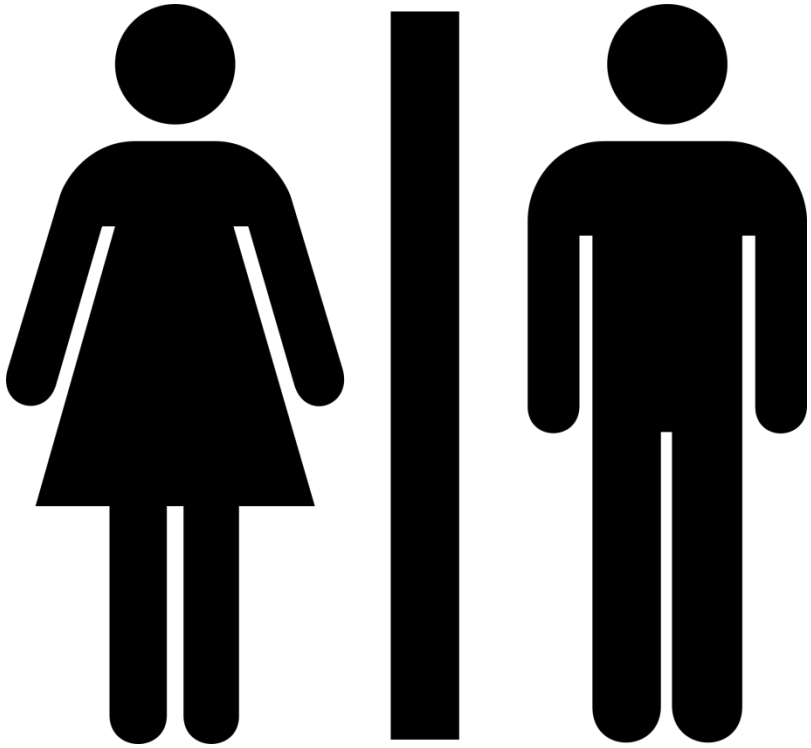
Smack: 3-Domain Policy

Floor	System	User
• RO system directories • Kernel helpers	• /run, /dev, /var/log • System services	• Home directories • Launcher and user services

Domains are sets of labels with common prefix.

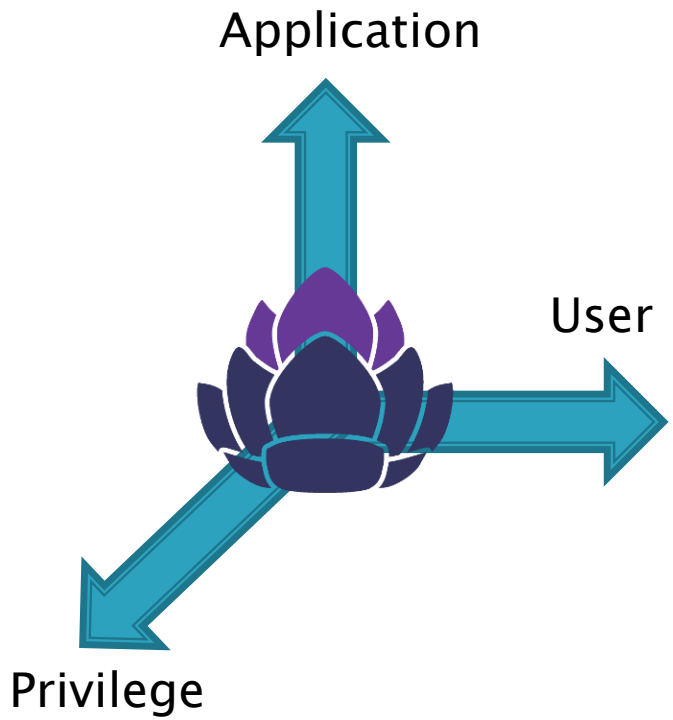
https://wiki.tizen.org/wiki/Security:SmackThreeDomainModel

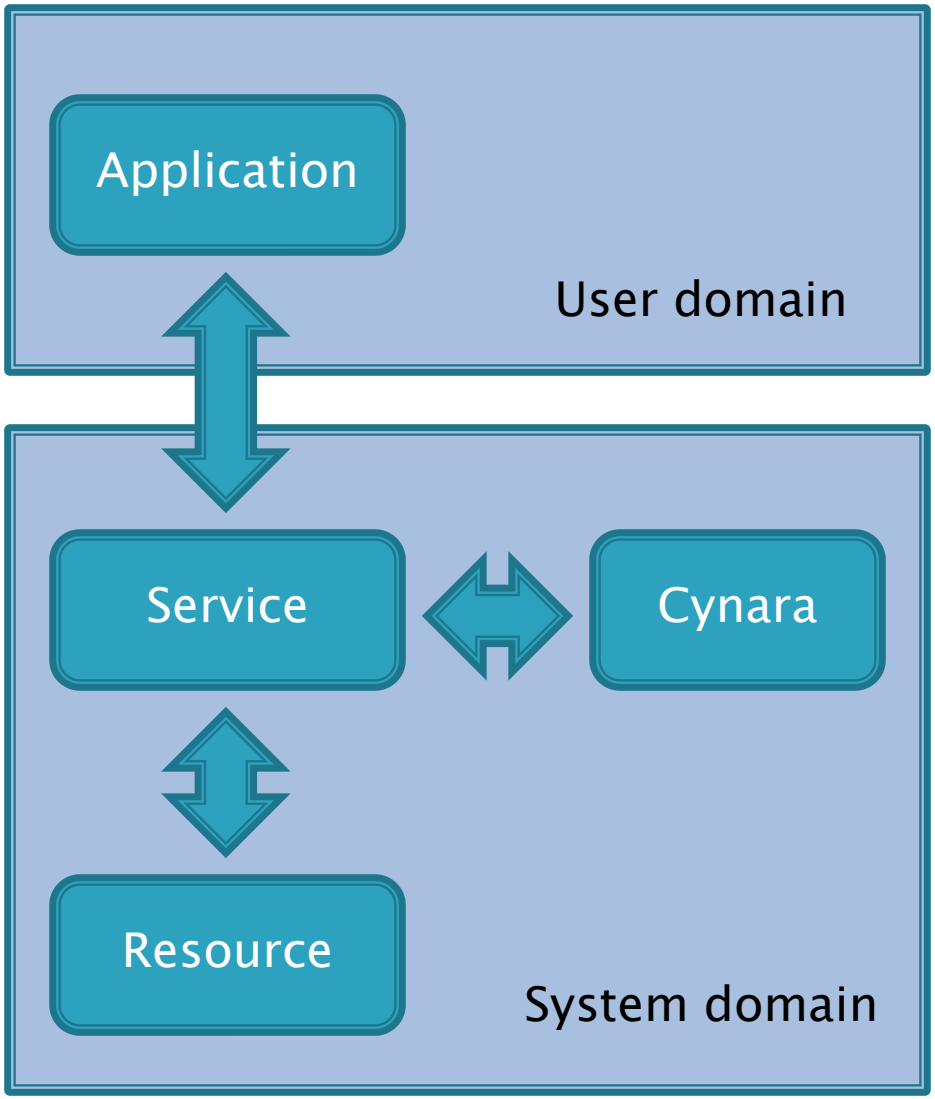
DAC – users separation



Same application running in context of different users
is separated with classic Linux users and groups mechanism.

Cynara – policy checker



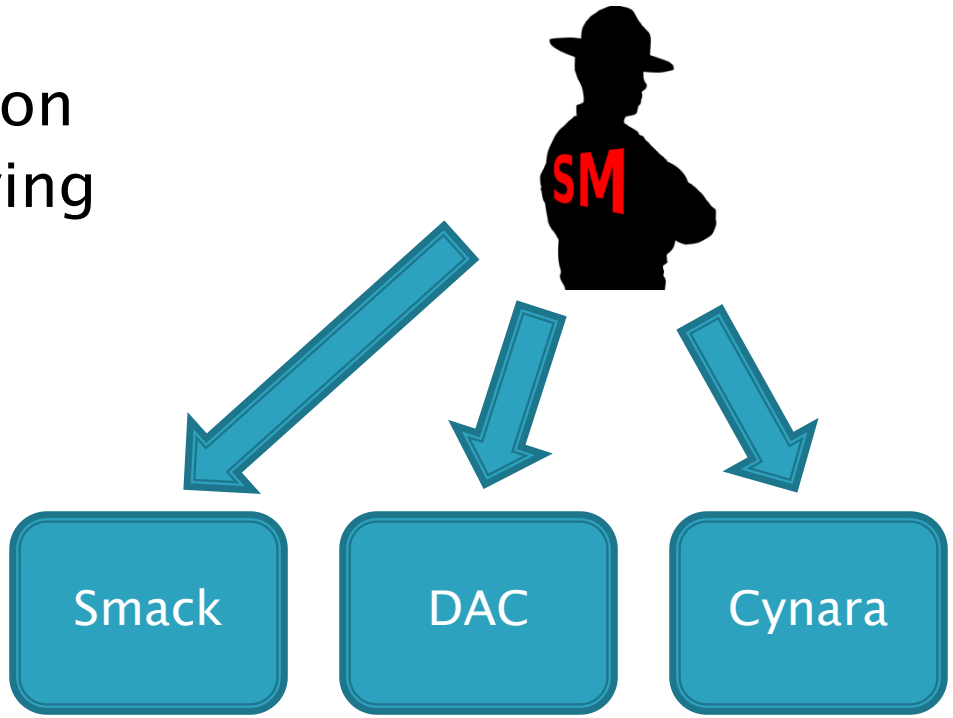


Security Framework

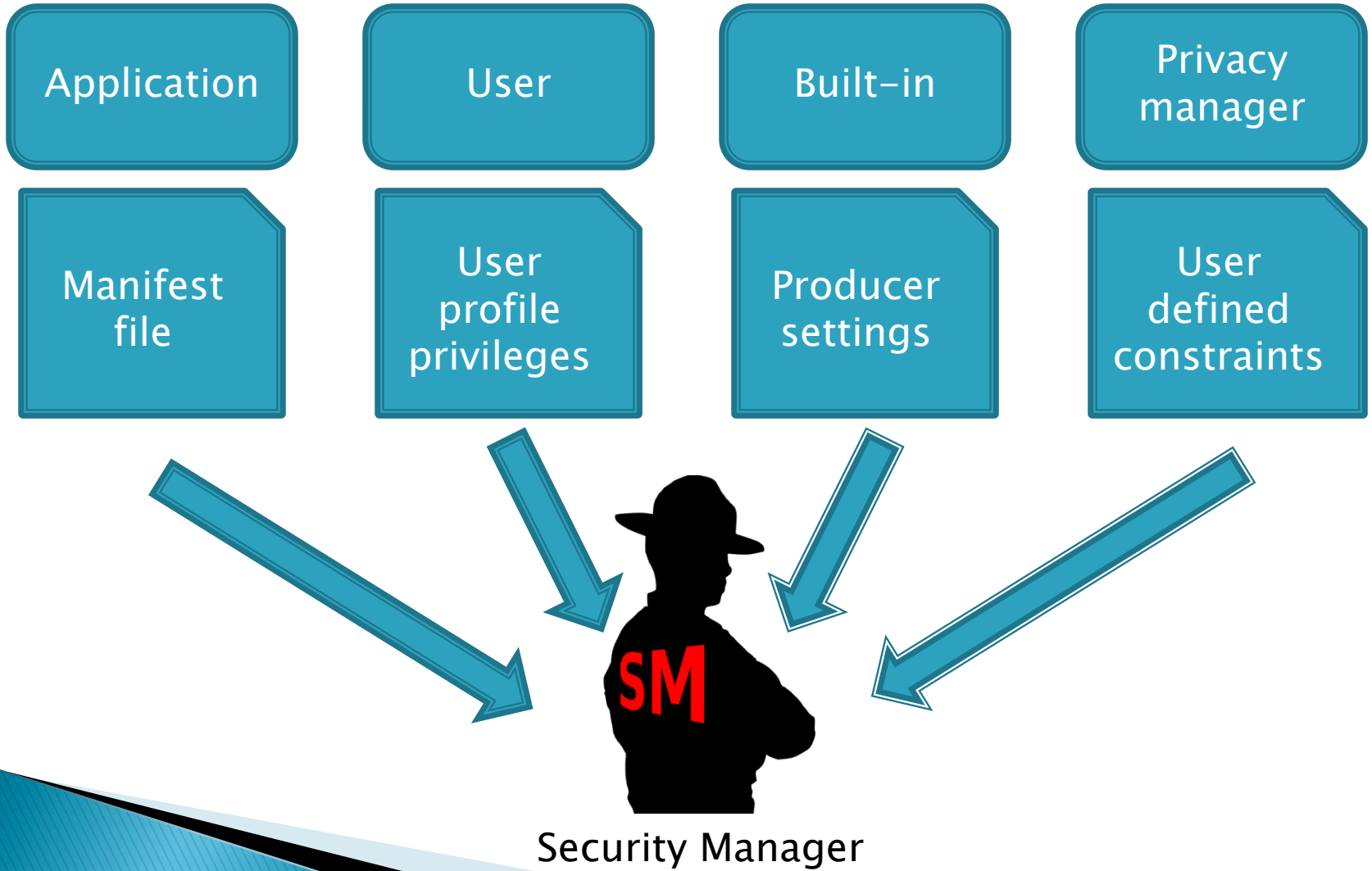
Step 2. Setup

Security Manager

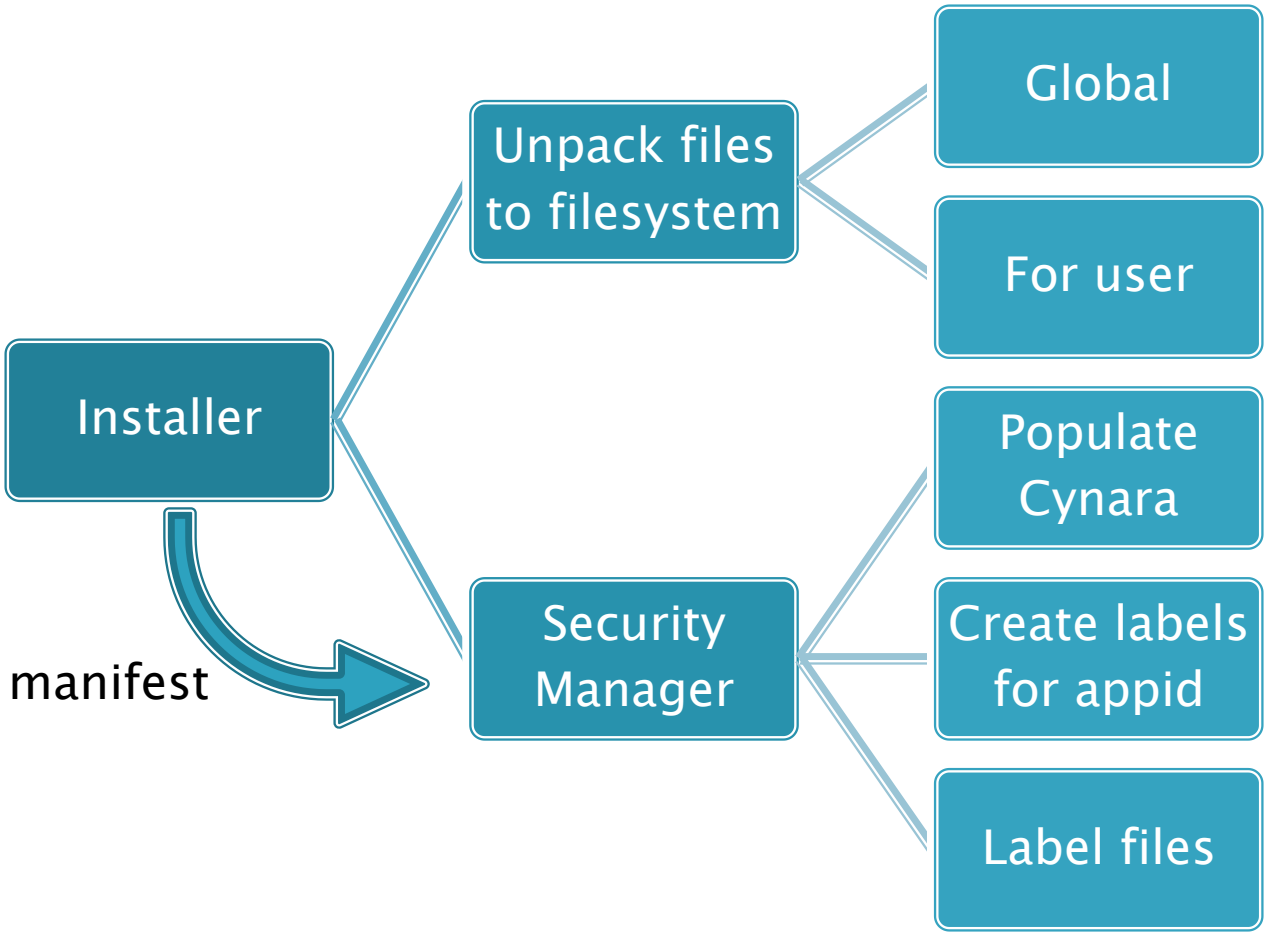
- Privileged service
- Sets launched applications' security context
- The only service allowed to setup security policy
  - Application installation
  - User adding / removing
  - Loading predefined manufacturer policy
  - Runtime policy reconfiguration



Sources of policies



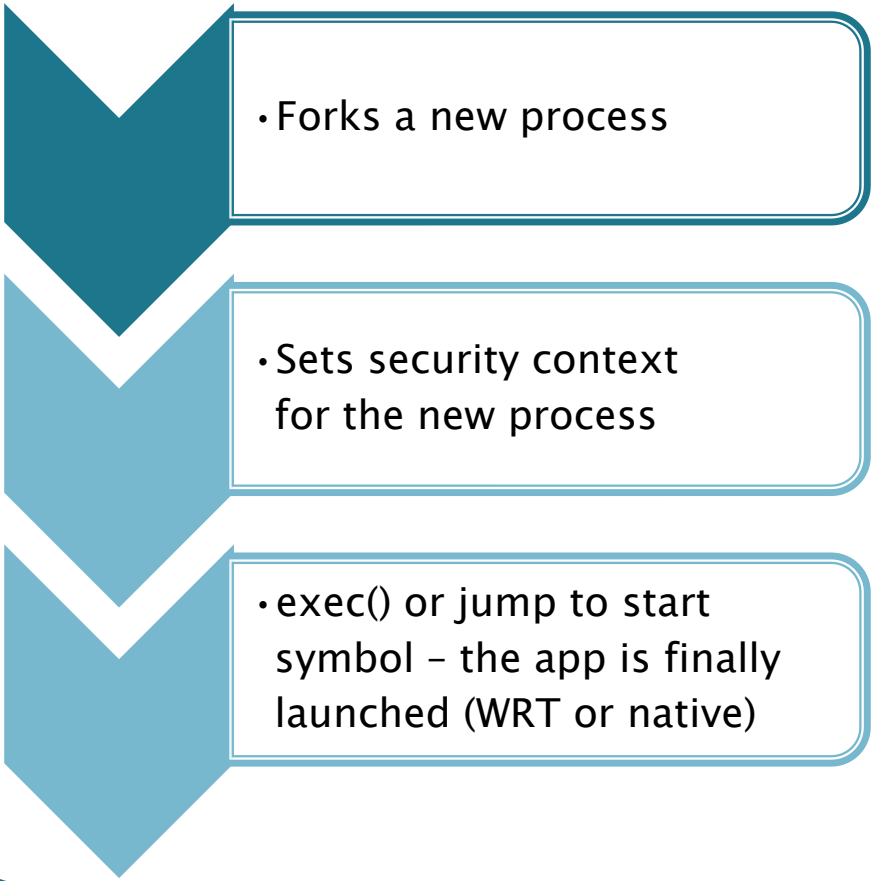
Package installation process



Security Framework

Step 3. Run

Application launching process



Launcher – privileged user service
(CAP_MAC_ADMIN)
runs with current user UID

SecurityManager – library
Security context is:
a Smack label
a set of effective UNIX groups

Application runs with label
so it can be uniquely recognized
and easily checked against Cynara policy

https://wiki.tizen.org/wiki/Multi-user_AMD

Runtime: accessing service

Application
ran with a proper Smack label
by some user
needs **GPS location**

Service maintaining restricted resource checks in **Cynara**, if access should be granted

App ID is a **Smack label**

Uid

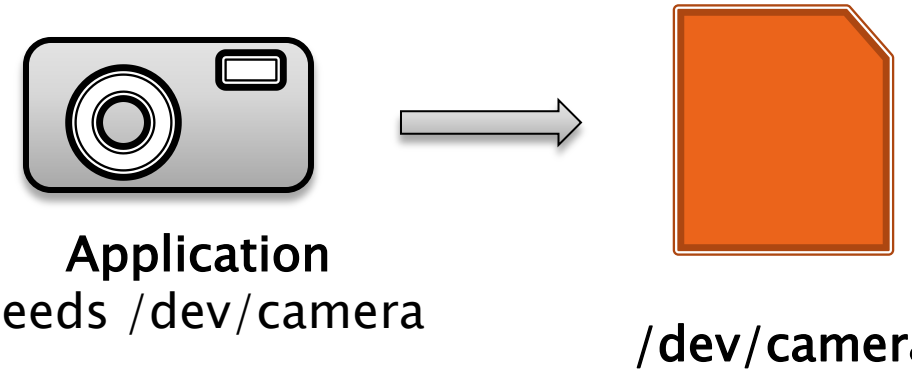
One of **privileges** in TIZEN 3.0,
e.g. http://tizen.org/privilege/location

https://wiki.tizen.org/wiki/Security:Tizen_3.0_Core_Privileges

Non trivial cases

Direct access resources

- Some resources cannot be wrapped with services (mainly because of performance)
- Solution = DAC groups



Application
needs /dev/camera

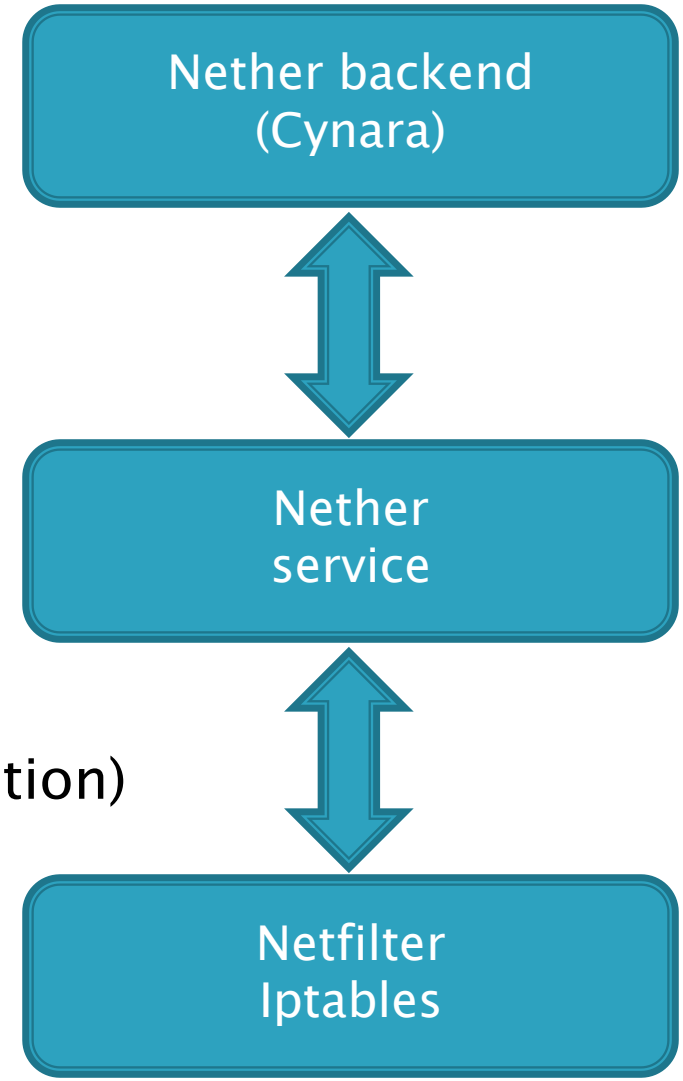
/dev/camera

DAC checks (Linux kernel itself), if process of application belongs to propper group

The groups are applied on every launch of application by Launcher and Cynara check is involved

Internet privilege

- Handled with **Nether**
- Configures iptables rules:
  - Mangle table rules for passing packets to user space and marking (with integer code)
  - Filter table rules for auditing, accepting or rejecting
  - Filtered packets:
    - TCP (1st packet in every connection)
    - UDP
    - ICMP



Auditing

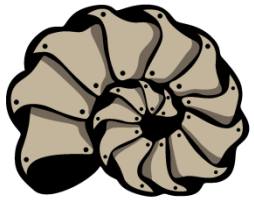
Nice-lad

- Narcissistic, Incredible, Completely Exceptional Logger of Access Denials
- Audisp plugin fed with audit event
- Aggregates and filters security related events
- Supported subsystems:
  - DAC denial on given group
  - Smack denials
  - Cynara denials
  - Netfilter denials (supported by Nether)

Containers

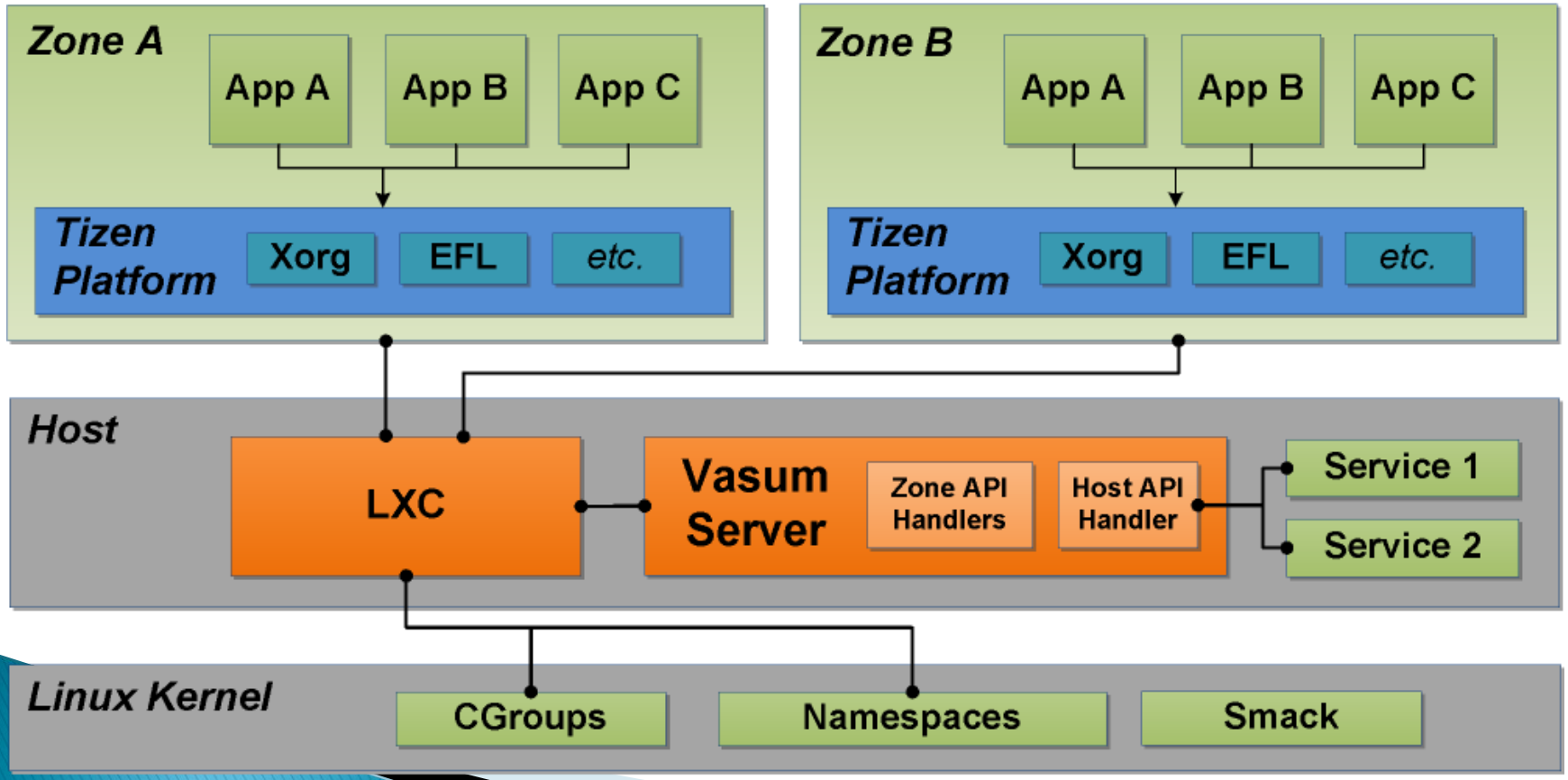
Vasum

Vasum



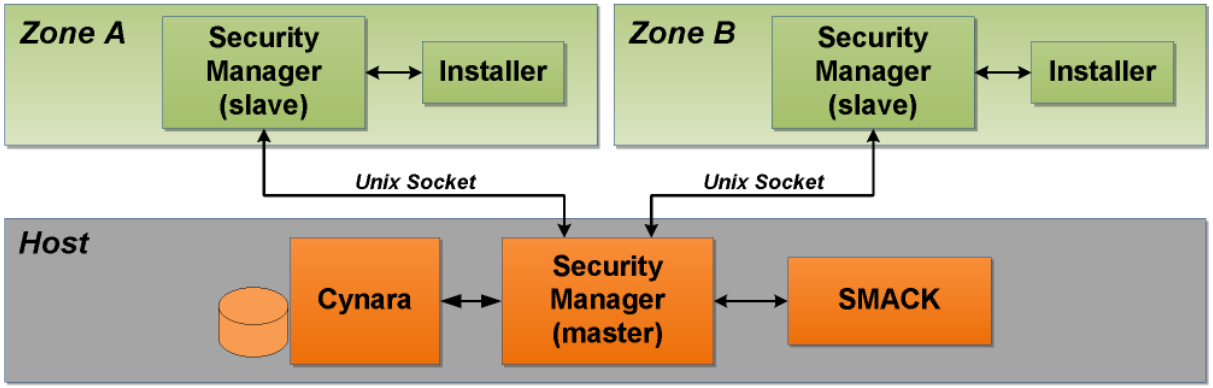
Vasum

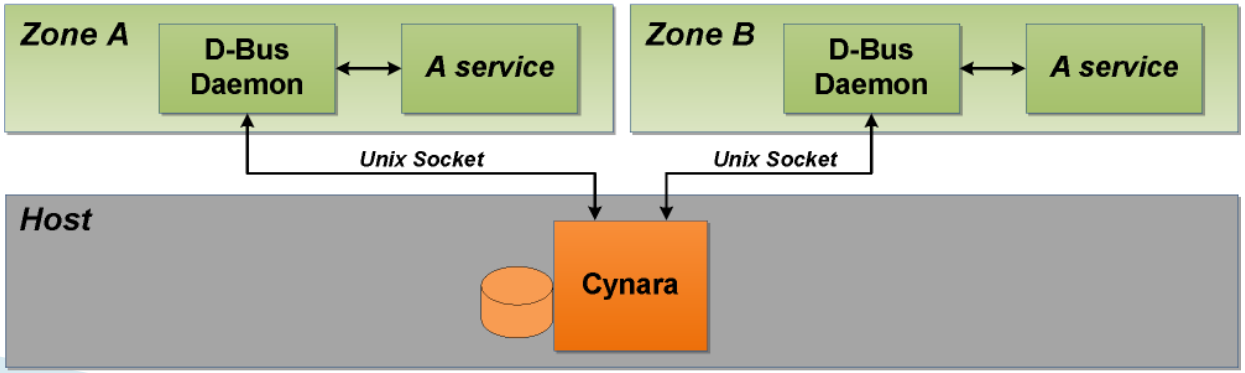
- Environment separation mechanism based on Linux Containers (LXC)



Security Framework

- Security framework integrated with Provisioning mechanism





Smack namespace – requirements

- Capabilities:
  CAP_MAC_ADMIN
  CAP_MAC_OVERRIDE
  should work inside the container

- Different containers shouldn't share Smack policies

Smack namespace – solution

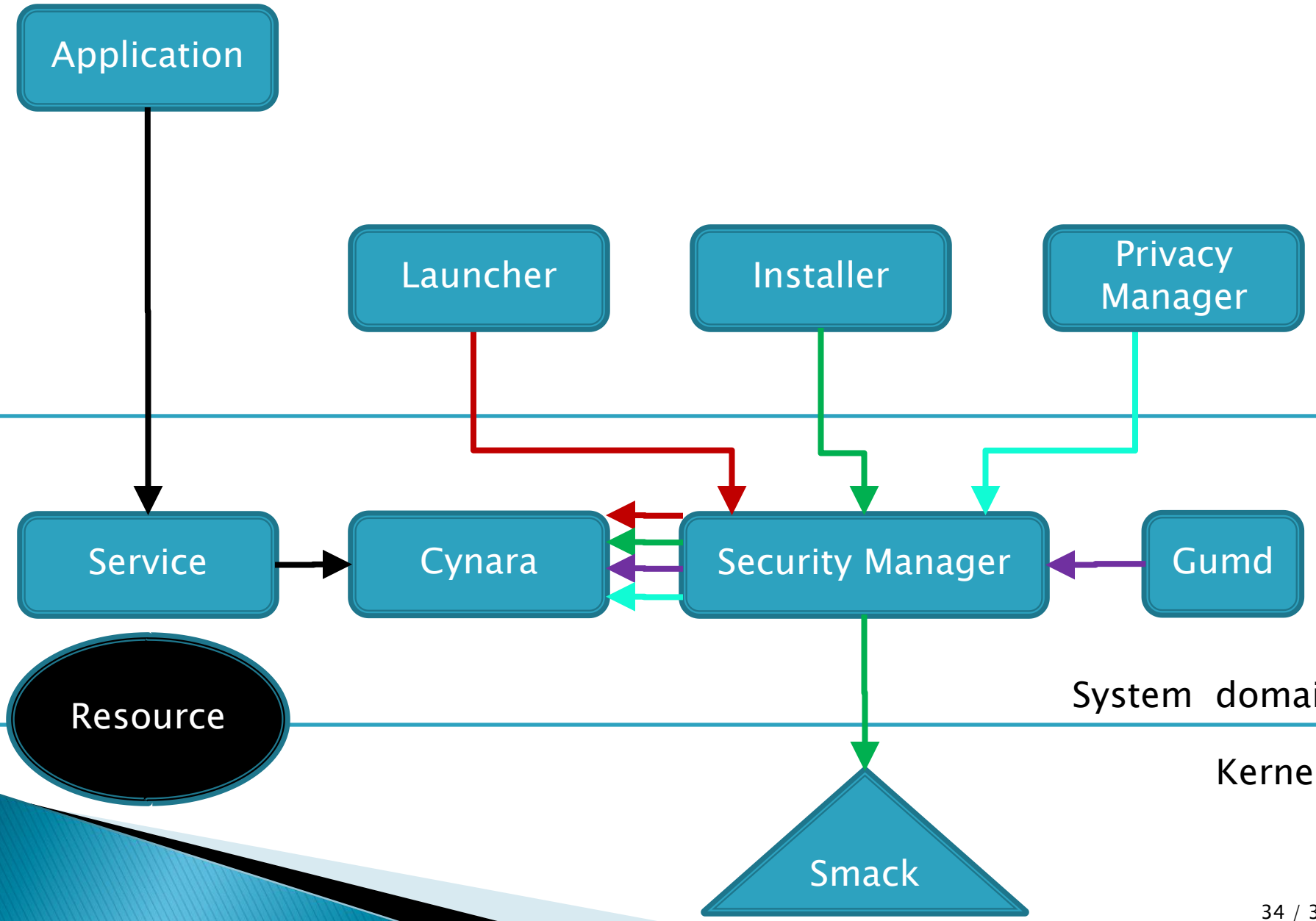
- Smack label mapping
  (implemented inside the Smack LSM)
- The map is tied to the User namespace
- The map is filled from the init ns

- Processes interact with Smack normally
- Only operations on labels that have been explicitly mapped are allowed
- All requests to access an object with an unmapped label will be denied

Summary

Overview

- Isolation of applications with: DAC + Smack
- Resources available through services API
- Privilege control enforced in services with check in Cynara
- Security policy controlled by Security Manager integrated into crucial processes:
  - Installation
  - Launching
  - Privacy and user management


User domain
Application
Launcher
Installer
Privacy Manager
Service
Cynara
Security Manager
Gumd
Resource
System domain
Kernel
Smack

Special cases

- Resources accessed directly protected by DAC groups assigned during launch
- Internet privilege filtered by Nether
- All security logs gathered by nice-lad
- Vasum allows easy creation of separate environments

Modules

- All modules available on both github.com and tizen.org
  - https://github.com/Samsung/security-manager
  - https://github.com/Samsung/nether
  - https://github.com/Samsung/nice-lad
  - https://github.com/Samsung/vasum
  - https://github.com/Samsung/cynara

Thank You

Questions ?

Images

- http://wulkana.republika.pl/polska.gif
- https://upload.wikimedia.org/wikipedia/commons/f/f8/Pong.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/8/87/Ic_quick_contacts_dialer_48px.svg/48px-Ic_quick_contacts_dialer_48px.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/2/2c/Web-browser-openclipart.svg/1024px-Web-browser-openclipart.svg.png
- https://pixabay.com/static/uploads/photo/2012/04/16/11/48/email-35636_640.png
- http://intergalacticrobot.blogspot.com/2009_06_01_archive.html
- https://upload.wikimedia.org/wikipedia/commons/thumb/b/b1/Email_Shiny_Icon.svg/256px-Email_Shiny_Icon.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/Instagram_Shiny_Icon.svg/500px-Instagram_Shiny_Icon.svg.png
- https://pixabay.com/static/uploads/photo/2012/04/26/14/14/internet-42583_640.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/Toilets_unisex.svg/2000px-Toilets_unisex.svg.png
- https://pixabay.com/static/uploads/photo/2014/04/02/11/14/police-305626_640.png
- https://wiki.tizen.org/w/images/5/52/Lad_overview.png
- https://wiki.tizen.org/w/images/d/d2/VasumDiagram_v1.png
- https://wiki.tizen.org/w/images/b/b5/Vasum_logo.png
- https://wiki.tizen.org/w/images/2/21/VasumCynaraPolicyCheck.png
- https://wiki.tizen.org/w/images/e/e6/VasumInstallation_v1.png

Smack namespace

- https://lwn.net/Articles/645403/
- https://lkml.org/lkml/2015/5/21/299

- Łukasz Pawelczyk
- l.pawelczyk@samsung.com

- Jan Olszak
- J.olszak@samsung.com

Application lifetime

Process	Application state	Security mechanisms		
		DAC	Smack	Cynara
Instalation	Installer unpacks files and manifest	Files are installed in proper dirs and rights are set	SM creates label for pkgid and labels files	SM populates Cynara's database
Launching	Launcher spawns new process	SM sets effective groups to provide access to special files (eg. devices)	SM sets apropriate label for new process	SM queries Cynara to check what groups should be applied
Runtime (1)	App requires access to some system service		Smack label uniquely identifies application	Service acquires label of app and checks in Cynara, if access should be granted
Runtime (2)	App requires direct access to a file or a raw device	Standard check of ownership	Standard check of access rules	

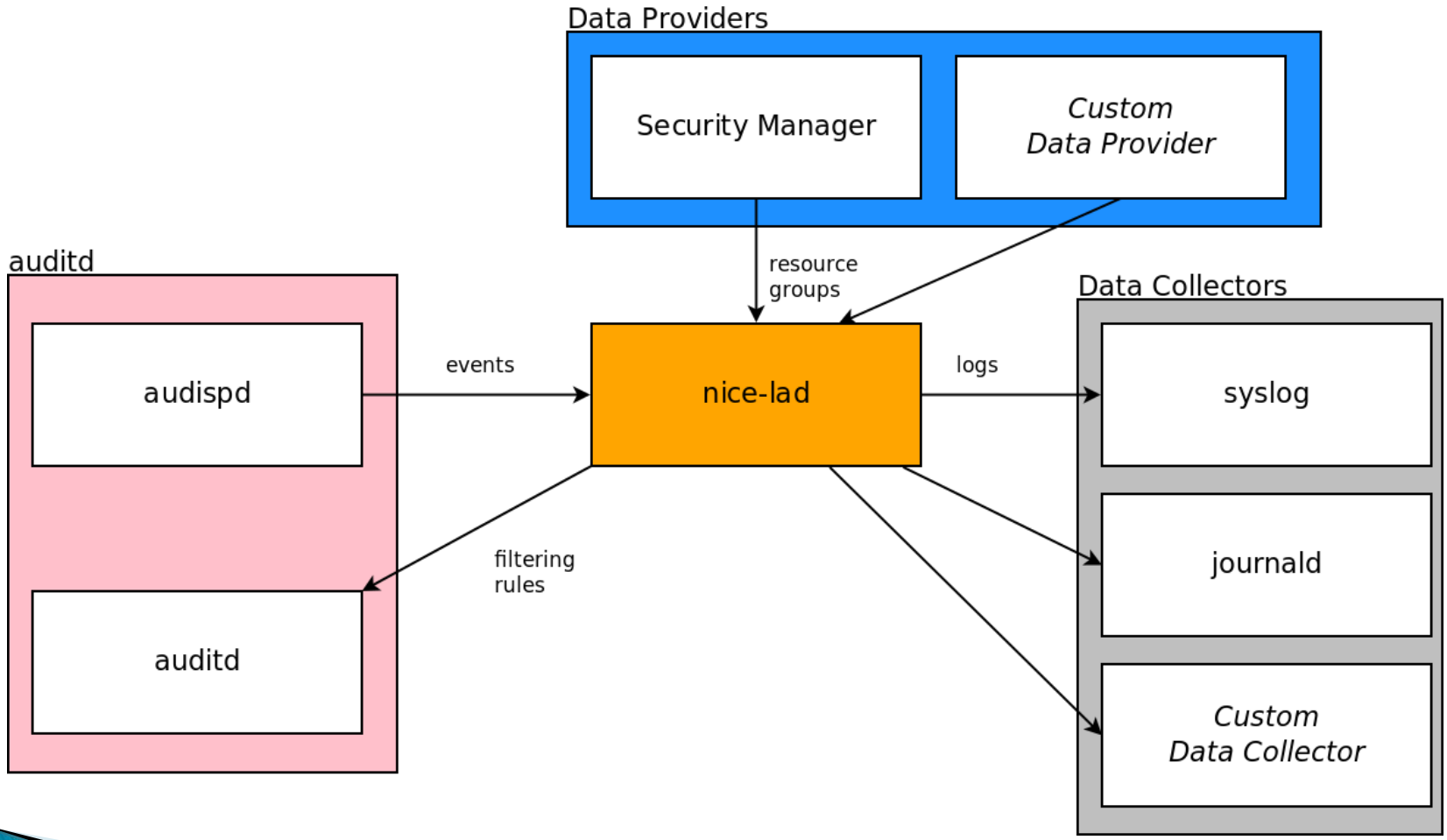
Cynara and DBus

```
<policy context="default">

  <check send_destination="com.example.service"
    send_interface="com.example.service.interface"
    send_member="SetAlarm"

  privilege="http://tizen.org/privilege/alarm.set"
  />

  <check send_destination="com.example.service"
    send_interface="com.example.service.interface"
    send_member="GetAlarm"

  privilege="http://tizen.org/privilege/alarm.get"
  />

</policy>
```

Nether details

- Uses NFQUEUE Netfilter mechanism
- Table: mangle
  - Nether, queue-0, queue-bypass -> mark (ACCEPT, DENY, ACCEPT + LOG)
- Table: filter
  - Nether-deny -> audit -> REJECT
  - Nether-accept+log ->audit -> ACCEPT

Nice-lad schema



Cynara is fast

Time*	PolKit	Cynara
Init + connect	12.37 ms	0.08 ms
Request + response	3.35 ms	0.15 ms
Policy check	14.45 ms	0.12 ms
Complete check	17.80 ms	0.27 ms

* measured on hardware equivalent of Samsung Galaxy S3