Mobile Simplified Security Framework Overview

Presented by:

Elena Reshetova

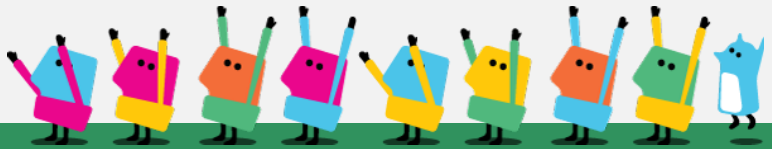
Casey Schaufler

Nokia Mobile Solutions

Outline

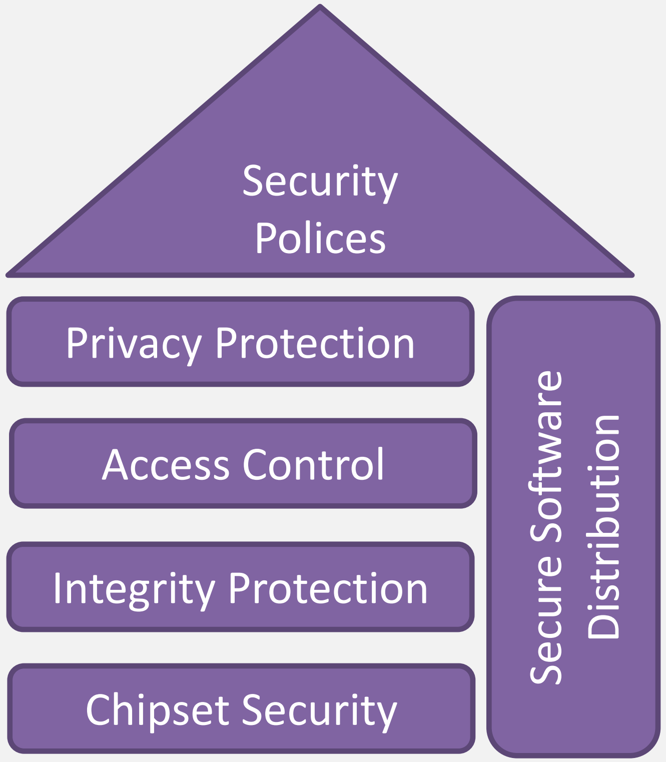
Mobile Simplified Security Framework (MSSF)

- MSSF Components
- MSSF Evolution
- Chipset Security
- Access Control
- Integrity Protection
- Privacy Protection
- Q&A



MSSF components

- Chipset Security
  - Provides secure cryptographic services and key management for higher levels
- Integrity protection
  - Ensures protection of TCB, applications and data
- Access Control
  - Limits application access to protected resources
- Privacy protection
  - Provides data integrity and confidentiality protection for applications
- MSSF relies on the secure software distribution model
  - Ensures the authentication of a package
  - Allows to manage remotely the security policy
- Security policies
  - Control points of the MSSF framework



MSSF Evolution

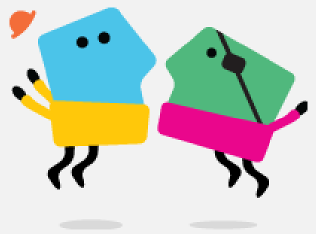
Mobile Simplified Security Framework

- MSSF v1 $\longrightarrow$
- MSSF v2

MeeGo 1.2

Differences

- A number of implementation changes
  - Platform based (Debian → RPM)
  - Feature based
    - Light-weight run-time file access control



Chipset Security

- Provides Trusted Execution Environment (TrEE)
  - Secure key management and cryptographic services
- Two main keys:
  - Root symmetric device specific key (RDSkey)
    - Used for local cryptography operations
  - Root Public Key (RPK)
    - Used to verify the software chain on the device
- Secure/authenticated boot
  - Verify integrity of the bootloader and SW image using RPK

MSSF Access Control - Definitions

- Protected resource
  - A virtual resource that needs limited access
    - Cellular functionality, Location information, Calendar, ..

- MSSF Resource token
  - String naming protected resource
    - Global: UserData, Cellular, Location, etc.
    - Package specific: my-package::access
  - Can be considered as new credential type in addition to UID, GID, GRP and POSIX capabilities

- Application must declare credentials it needs or provides in the Manifest file

Access Control – Reasons for moving to Smack

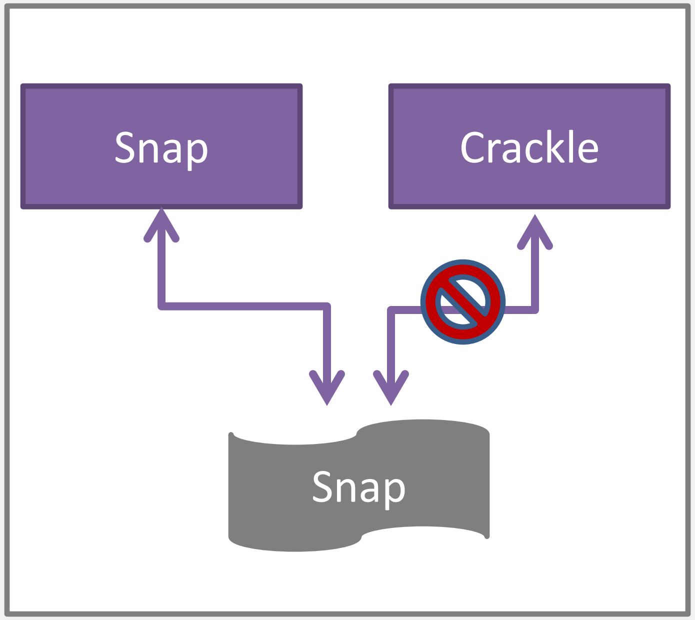
Simplified Mandatory Access Control Kernel

- Mainline Linux Security Module (LSM)
- Complete mandatory access control model
- Resource token implementation for MSSF v2

Smack Basics

Labels Must Match

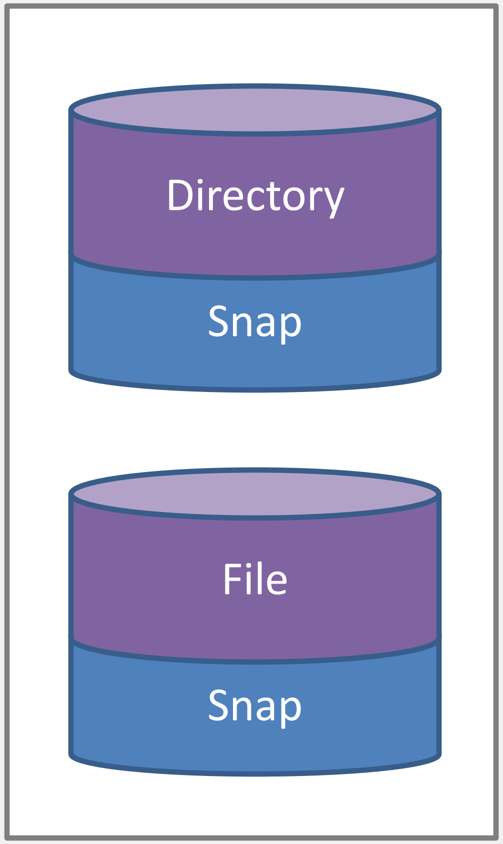
- Access requires that labels match
- Exceptions for system data
  - Floor "_"
  - Star "*"
- Exceptions may be specified
  - Subject Object Access

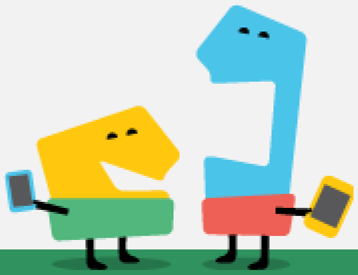


Smack File Access Control

Additional Restrictions

- Write access requires read access
- Access to attributes is also controlled
- Many operations require directory access
- Based on file system extended attributes

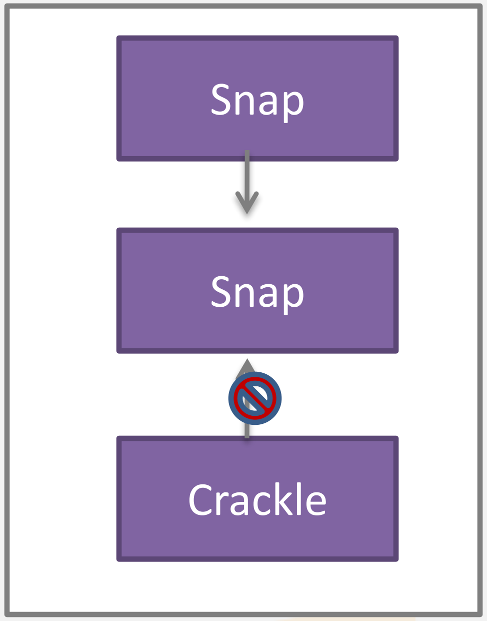




Smack Networking

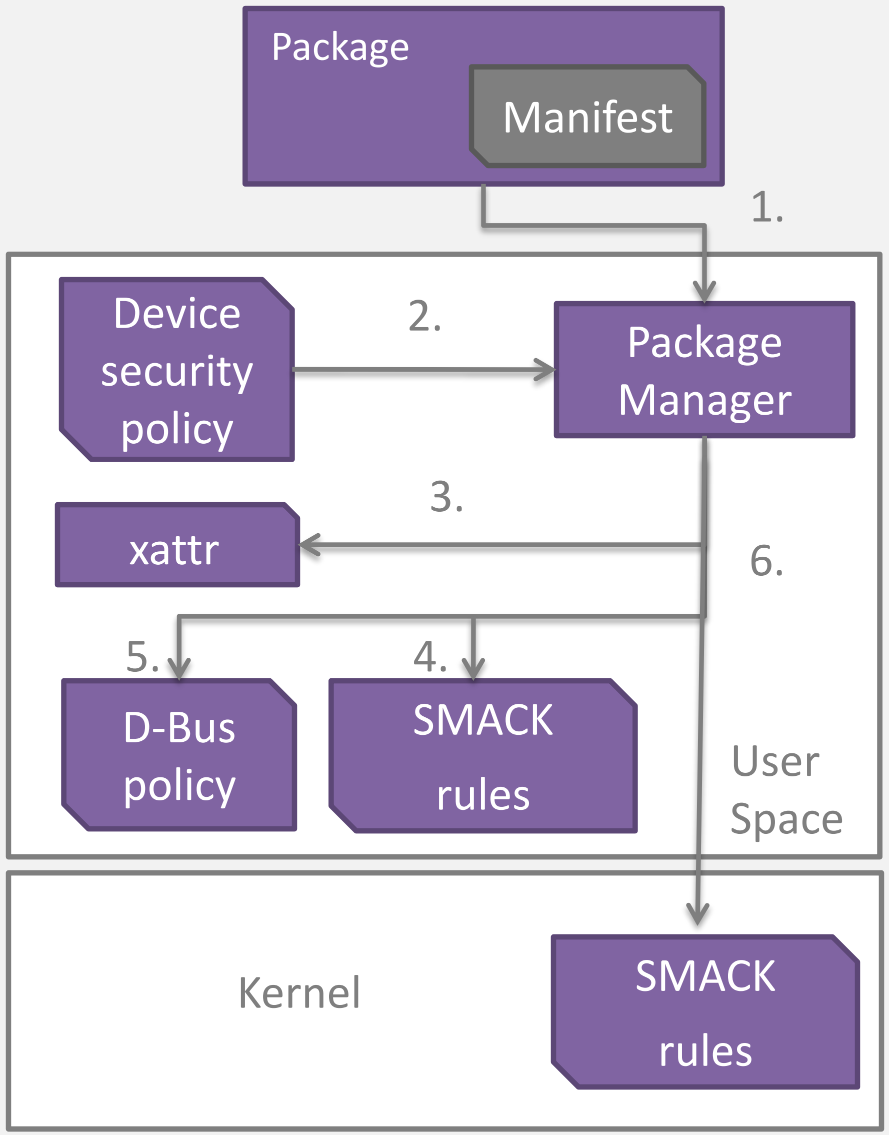
Networking As Interprocess Communication

- Sender must have write access to receiver
- Privileged process can set socket labels
- Packets are labeled
- Process can get label of the packet



Internal view - Installation

1. Application packaged with Manifest file comes to device
2. Package manager checks the Device Security policy
3. Package manager creates extended attributes
4. Package manager modifies the Smack rules
5. Package manager possibly modifies D-Bus policy
6. Package manager updates the Smack rules in kernel



Manifest mapping example

Manifest

- Server (comes from server-pkg) defines resource token UserData needed to access the server

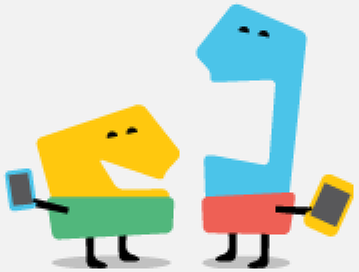
```
<aegis>
  <provide>
    <credential name="UserData" />
  </provide>
</aegis>
```

- Client declares that it requires tokens UserData and Cellular

```
<aegis>
  <request>
    <credential name="server-pkg::UserData"/>
    <credential name="Cellular"/>
    <for path="/usr/bin/udmanager"/>
  </request>
</aegis>
```

Smack rules

Subject	Object	Access
udmanager	Cellular	rw
udmanager	server-pkg::UserData	rw



Integrity Protection – IMA

Reasons for a change

- Mainline integrity protection module
- Usage of extended attributes

Features

- Stores a reference hash of a file in security.ima extended attribute
- Verifies integrity of a file based on reference hash in run-time
- Reference hash is automatically recalculated, when a file is modified (modification must be allowed by Access Control Framework)

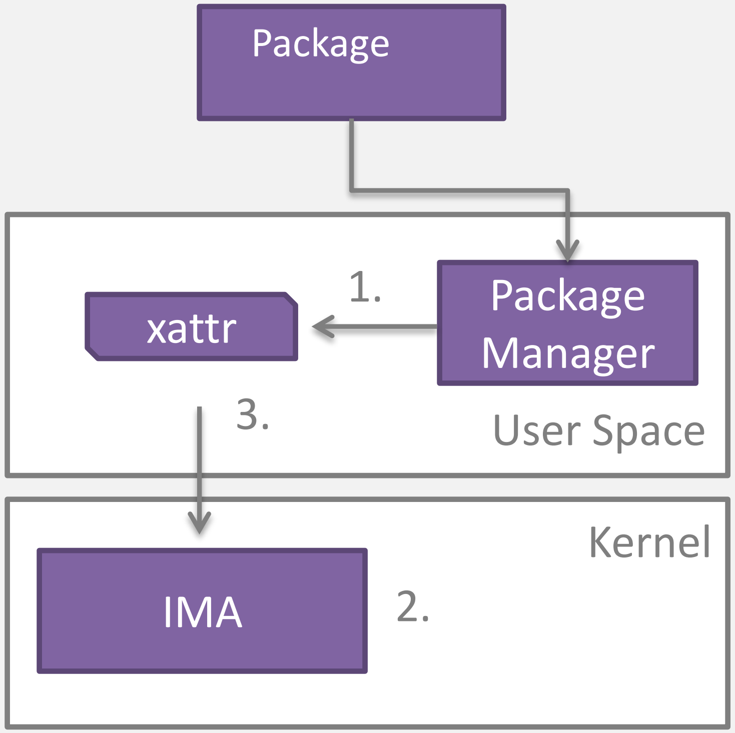
Integrity Protection - IMA

Application installation time

1. Package Manager updates the extended attributes with the reference hashes from the package

Application startup time

3. IMA calculates the hash of application binary
4. IMA compares it with the reference hash value loaded from extended attribute



Integrity Protection - EVM

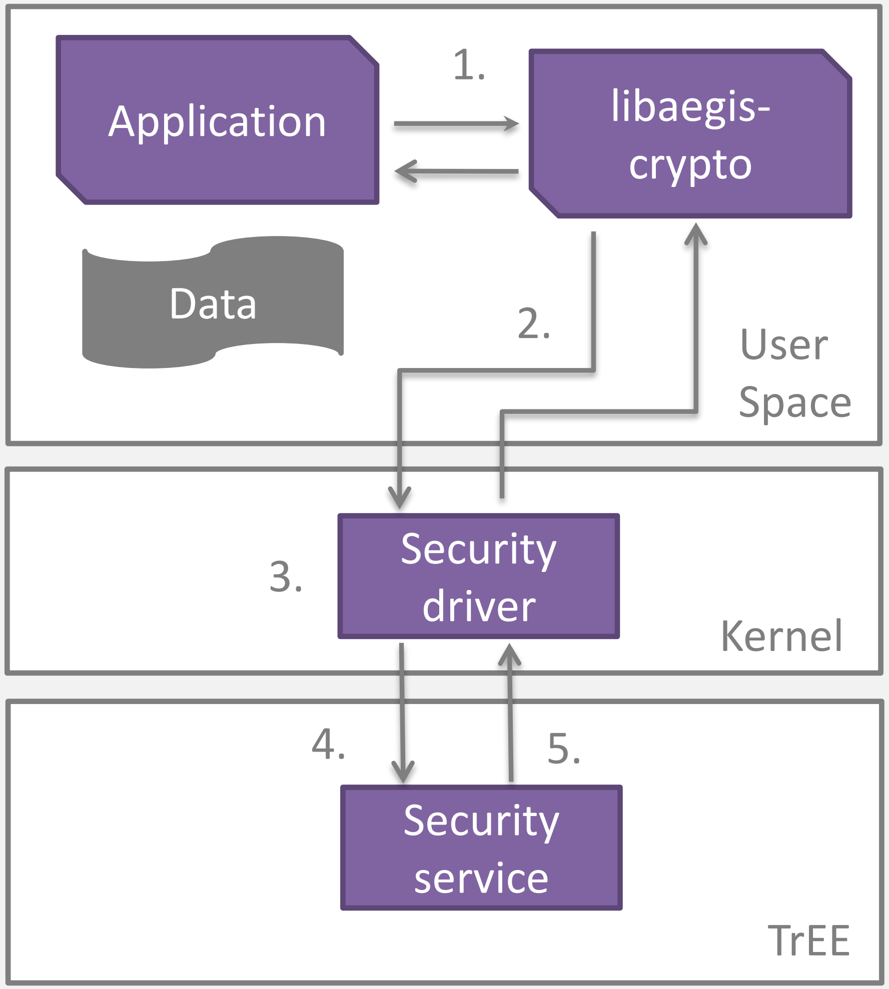
Extended Verification Module

- Offline protection for filesystem metadata
  - Owner, group & mode
  - Maintains a keyed hash across security attributes
    - security.ima
    - security.SMACK64
    - etc.
- Key is tighten to the Chipset security keys

Crypto Services

- Ensures integrity or confidentiality of data after installation
- Access to protected data is defined by either
  - Application specific key
    - K(AppID, RDSkey)
  - Shared key
    - K(Resource token, RDSkey)

- Interaction scenario example:
  1. Application calls libaegis-crypto to compute MAC on the data
  2. libaegis-crypto transfers request to a security driver
  3. Security driver verifies if application can perform the operation
  4. - 5. The MAC is computed and returned to the application



Questions?

What's next?

- Brian McGillion & Juhani Mäkelä "The cost of security, a developer's view."
- Ryan Ware "BOF session on MeeGo security"

Where do I find source code?

- Public project "Mobile Simplified Security Framework"

http://meego.gitorious.org/meego-platform-security/

I have much more questions!

elena.reshetova@nokia.com

casey.schaufler@nokia.com