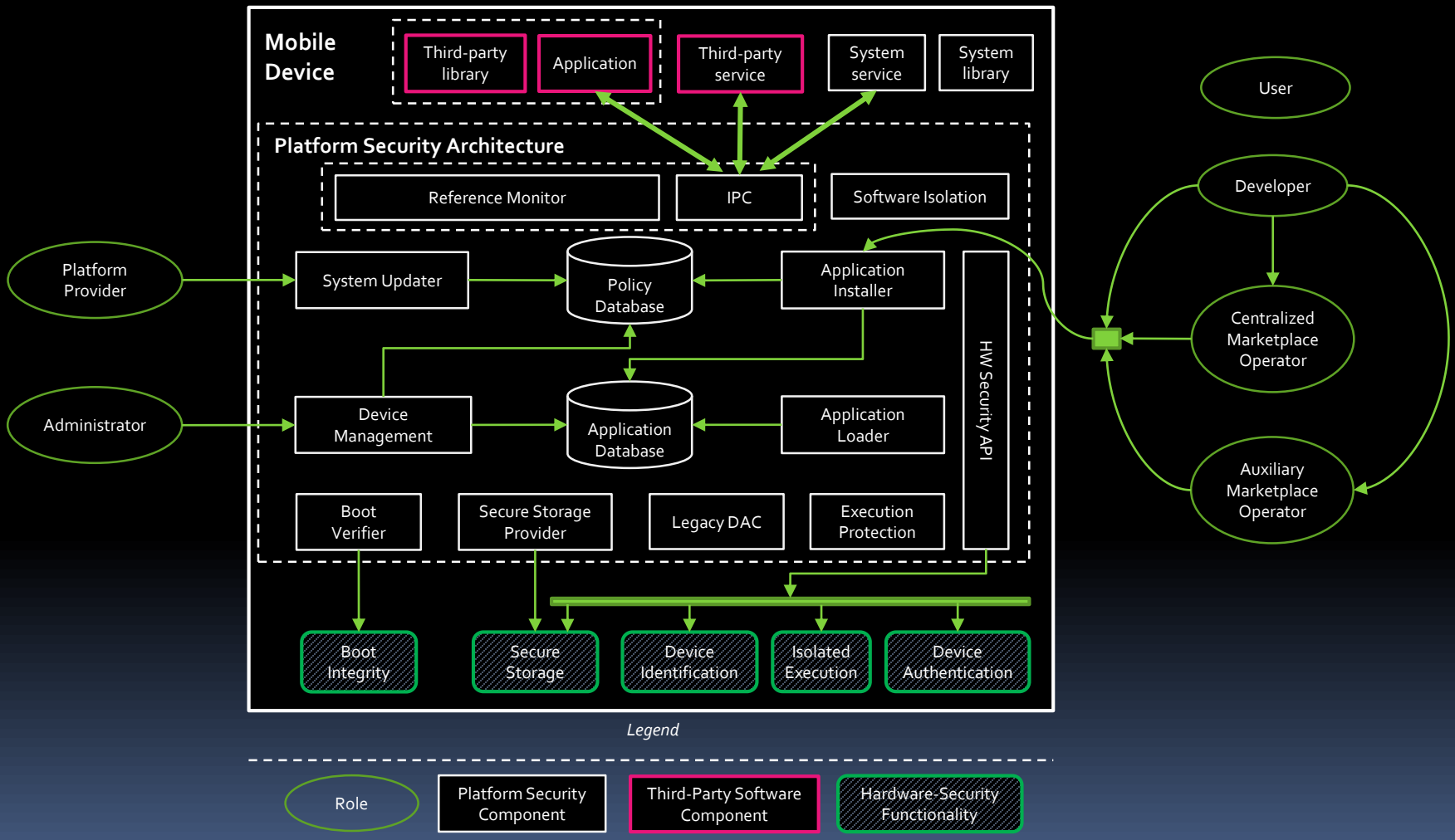
Lecture 2

PLATFORM SECURITY IN ANDROID OS

You will be learning:

- Android as a software platform
  - Internals and surrounding ecosystem
- Security techniques in Android:
  - Application signing
  - Application isolation
  - Permission-based access control
  - Hardware-based security features

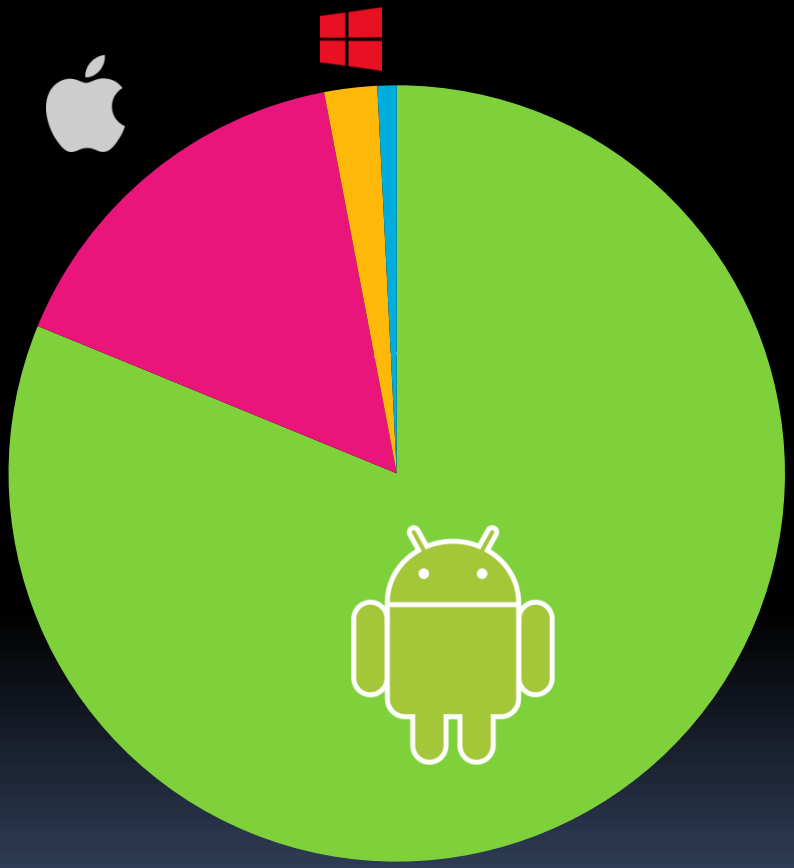
Mobile Software platform security



Mobile software platforms

- Which mobile platforms have you heard of?

Smartphone platforms



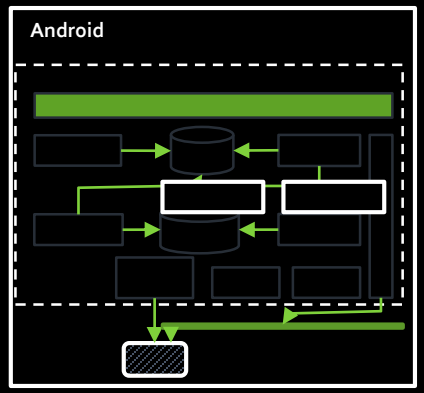
Android	81,2%
iOS	15,8%
Windows Phone	2,2%
Other	0,8%

(IDC Dec 2015, market share forecast 2015)

Android in a nutshell

- Linux-based (ARM, x86, x86_64, MIPS)
- Widely used for phones and tablets
  - Wearables, smart TVs, cameras, (handheld) gaming consoles, etc.
- Open-source software stack + closed source applications and services

Security goals

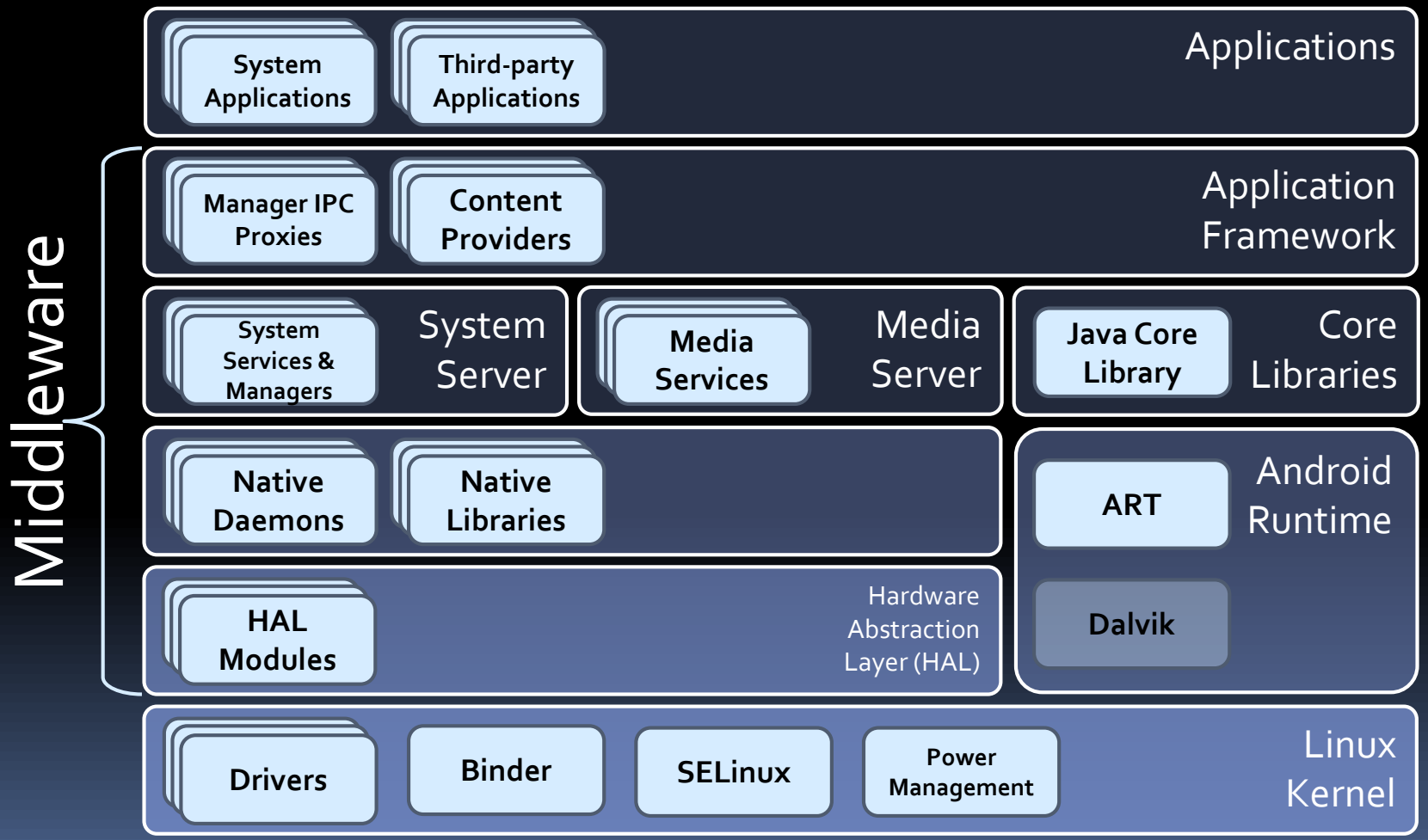


- Protect user data
- Protect system resources
- Provide application isolation

On terminology

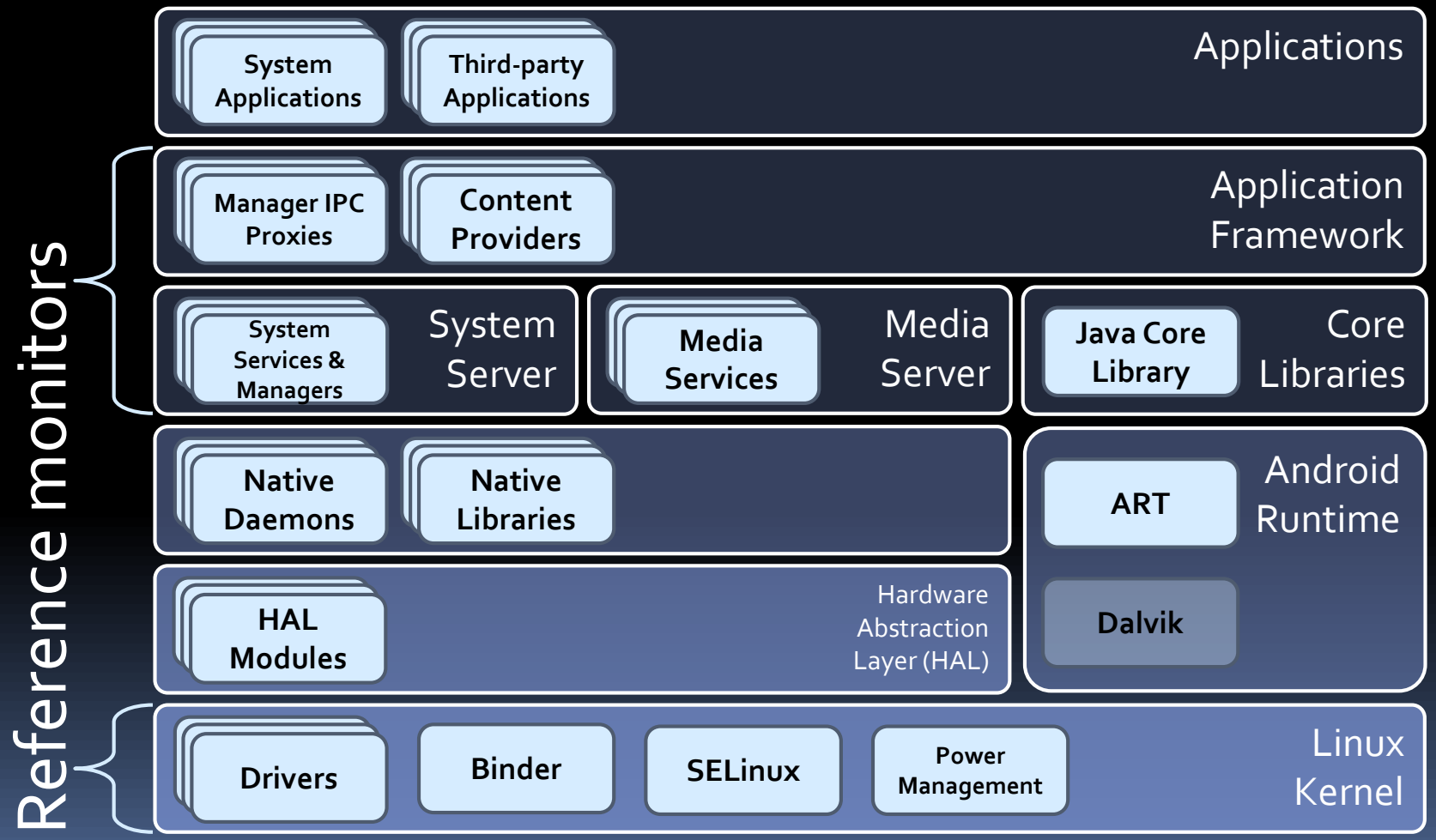
- Linux = the kernel
- “Desktop Linux” ≈ GNU / Linux
- Linux DAC = (Unix) file permissions
- Linux MAC = SELinux
- Permissions = Android app perms.

Android Software Stack



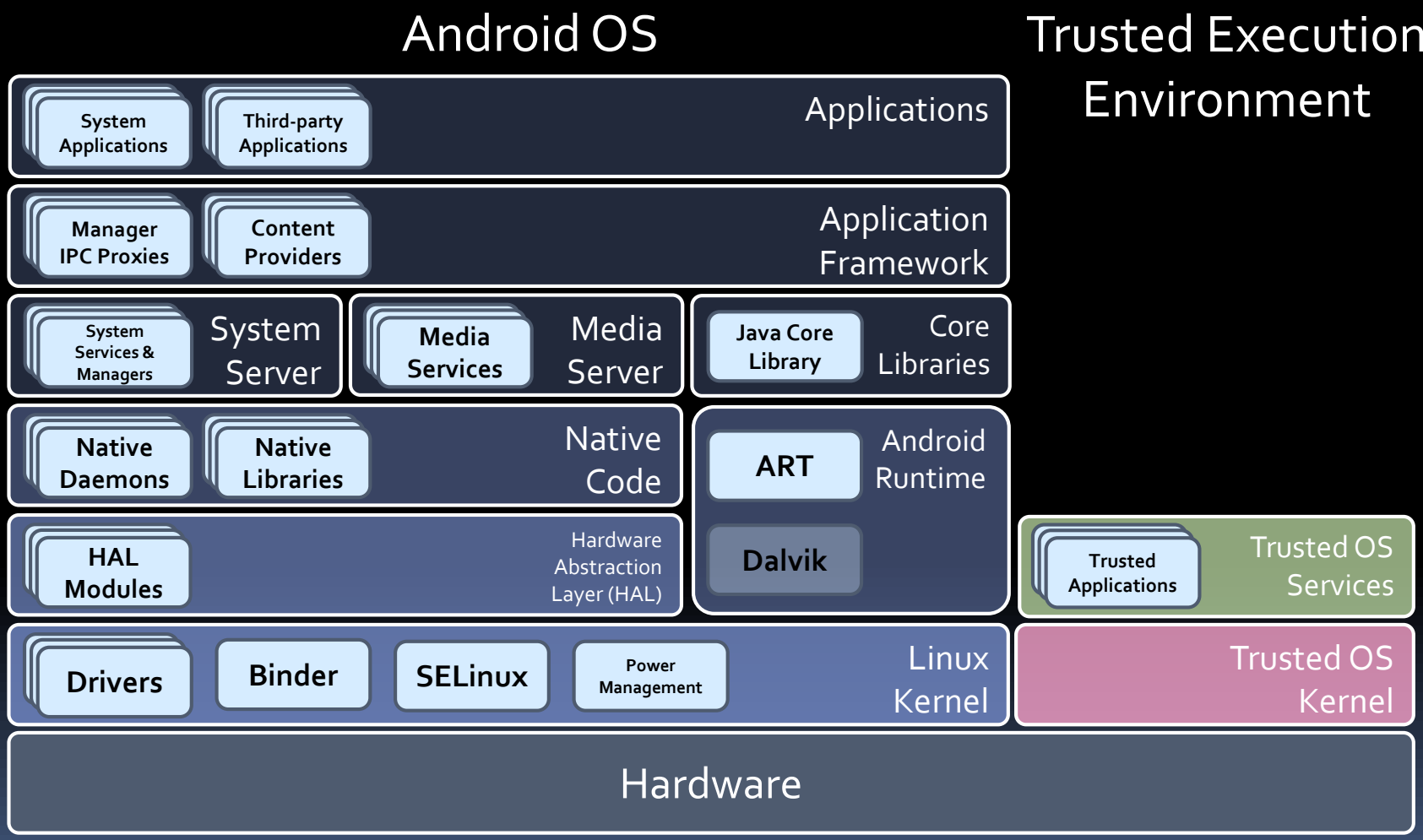
Android Open Source Project. Security. 2015

Android Software Stack



Android Open Source Project. Security. 2015

Android Software Stack



Google. Android for Work Security white paper. 2015

Application components

Activity • Visible UI

Intent

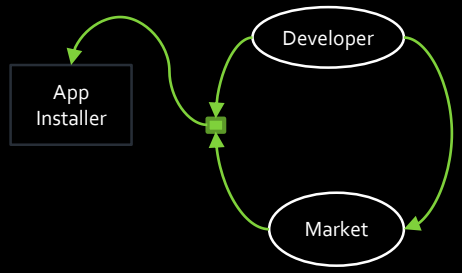
Service • Background job

Content Provider • Database

Broadcast Receiver

Enck, Ongtang, McDaniel. *Enck, Ongtang, McDaniel: Understanding Android Security*. *2009*
Android Open Source Project. *Application Security*. *2015*

Software distribution



- Apps from multiple sources
  - Google Play
  - Auxiliary marketplaces
  - Sideloading
  - Pre-installed software
- Marketplace services
  - Discovery
  - Purchase & Installation
  - User-submitted ratings / flagging
  - Malware scans (Google Bouncer)
  - Remote application installation & removal

Application signing

- Goal: same-origin policy for apps

Developer signs with private key

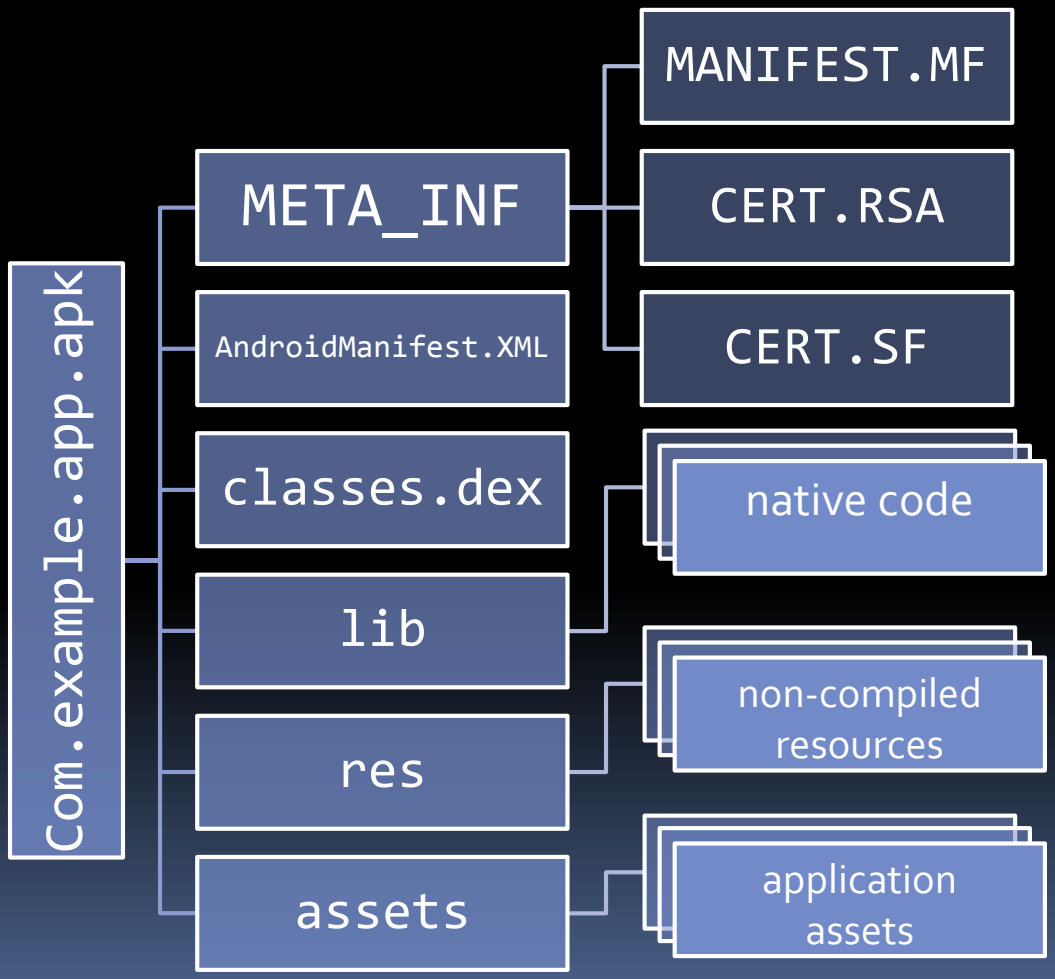
Signature verified with developer's public key

Application signing (cont.)

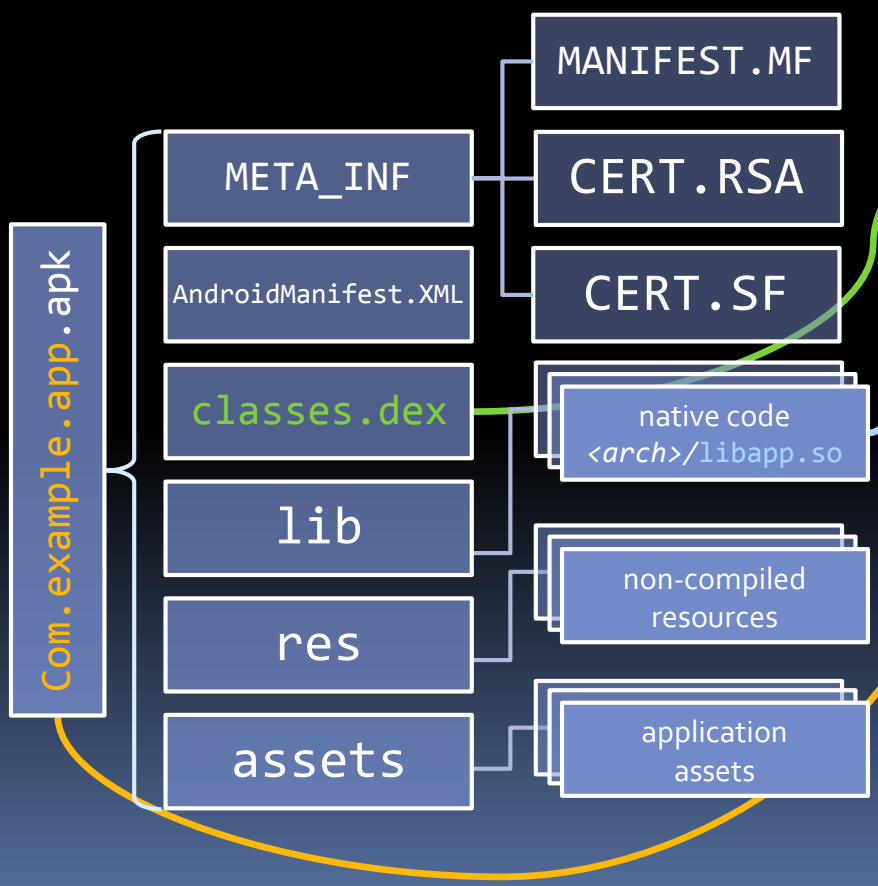
- For application packages (APKs)
  - Self-signed X.509 certificates (no PKI!)
  - Individual signature for each file part of APK
  - Package update requires same certificate
- For over-the-air updates (OTAs)
  - Signature over entire file stored in ZIP comment
  - Verified by OS and Recovery Mode
- System images must be signed (since 6.0)

Elenkov. Android's Security Architecture. 2015.

Android application packages

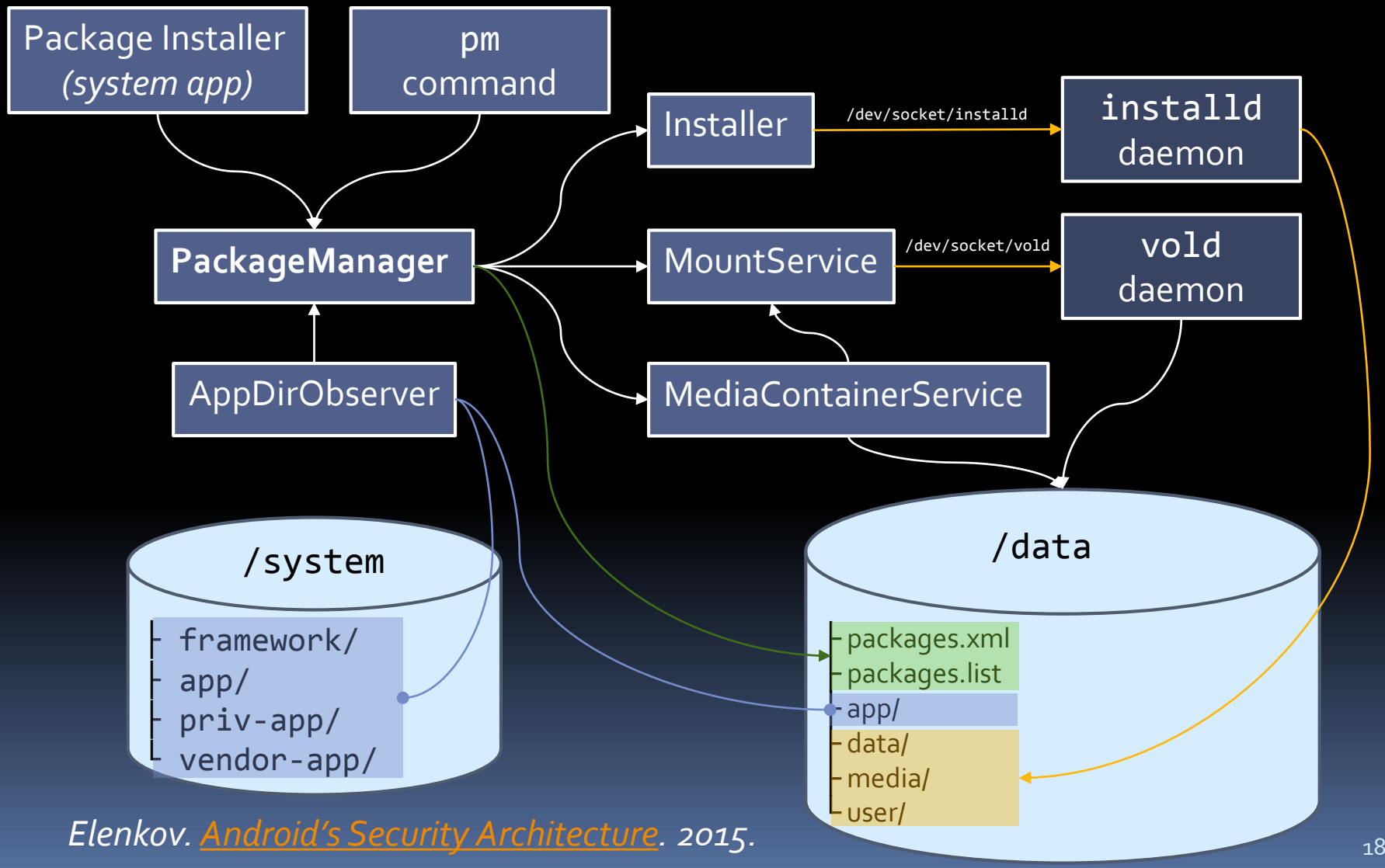


Package Installation



- Code and resources (common)
  - **/data/app/com.example.app/**
    - lib/*<arch>*/libapp.so
    - oat/*<arch>*/base.odex
    - base.apk
- Data (per user)
  - **/data/user/0/com.example.app/**
    - files/
    - databases/
    - shared_prefs/
  - **/data/user/1/com.example.app/**
    - …

Package management components



Elenkov. Android's Security Architecture. 2015.

Application isolation

- Goal: Applications cannot interfere with one another

Application isolation

Implementation on Android:

- Kernel: Process & memory protection
- Kernel: Linux DAC
- Kernel: Linux MAC (SELinux)
- Middleware: mediation of Binder IPC
- Applications run in separate Dalvik / ART virtual machine instances

Application Sandbox

- Each application assigned a Unix UID
  - One UID per user per application (since 5.0)
  - UID owns
    - Filesystem resources in `/data/user/<nr>/`
    - `Processes`
    - `Permissions (!)`
- Applications from same developer (= signed with same developer key) may share UID sandbox

Application isolation

- Linux DAC domain (UID)

Rooting

- Rooting applications exploit vulnerabilities in privileged system daemons to obtain shell
  - Note: bootloader unlocking intentionally supported by many OEMs
  - e.g. `fastboot oem unlock`

SE for Android

- Goal: System services and applications should not be able to deviate from their intended *modus operandi*

SE for Android (cont.)

- Implementation on Android:
  - Kernel-level MAC (SELinux) – Policies based on SELinux context
  - Middleware MAC (MMAC) – Policies based on package identity

Smalley, Craig. Smalley, Craig: Security Enhanced (SE) Android. 2013
Android Open Source Project. Security-Enhanced Linux in Android. 2015

SE for Android (cont.)

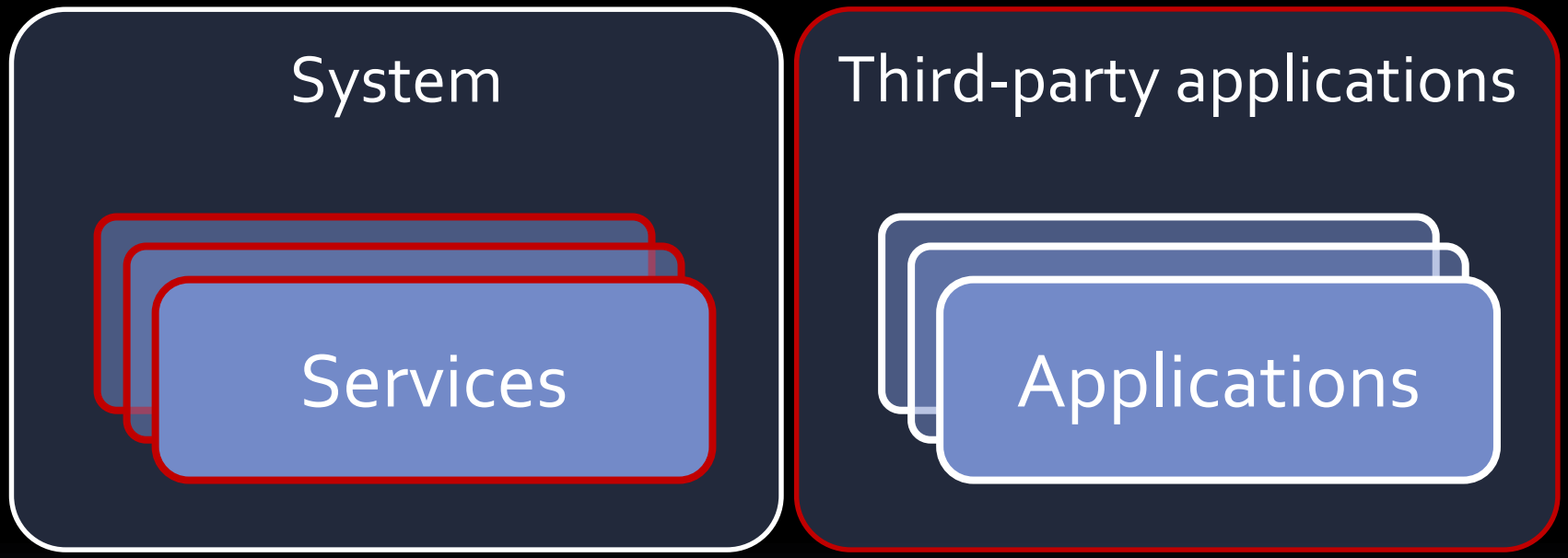
- Enforces MAC even for processes running with root/superuser privileges (since 4.4)
- Blocks many root exploits and misconfigurations
- Cannot protect against kernel exploits

Smalley, Craig. Smalley, Craig: Security Enhanced (SE) Android.2013
Android Open Source Project. Security-Enhanced Linux in Android. 2015

SE for Android

- *Domain* - Label for process(es)
- *Type* - Label for object(s)
- *Class* - Kind of object being accessed
  - (e.g. file, socket)
- *Permission* - Operation being performed
  - (e.g. read, write)

Application isolation (cont)

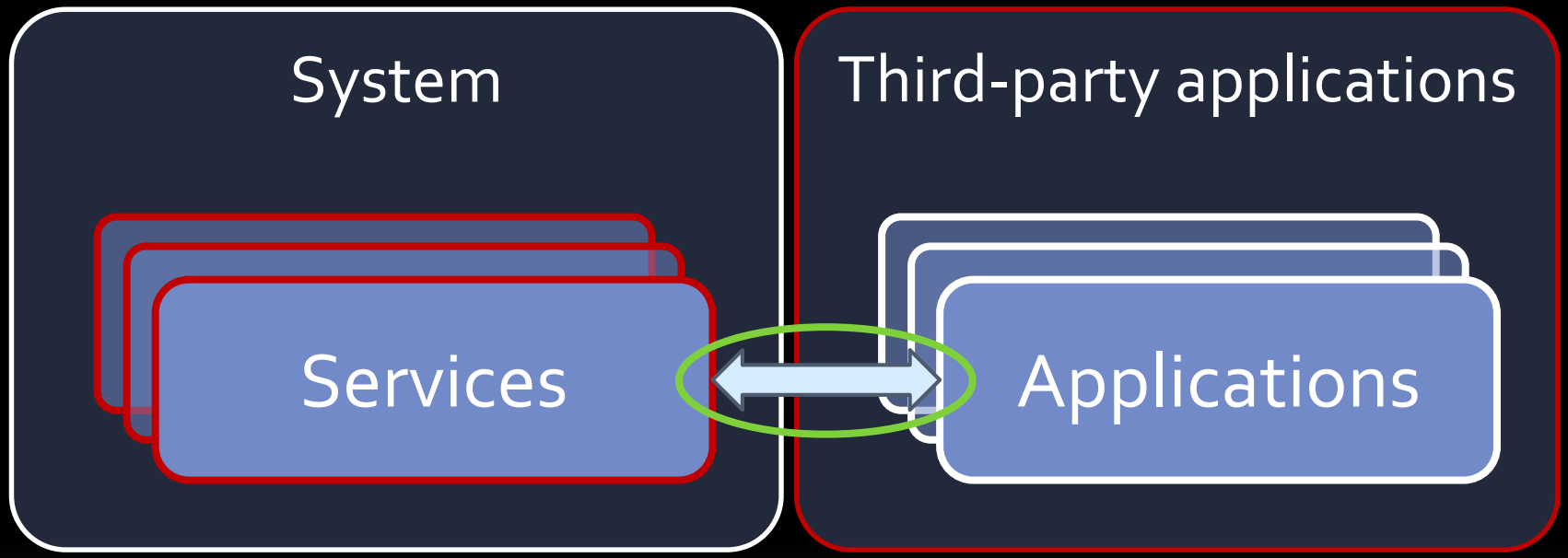


- Linux DAC domain (UID)
- Linux MAC domain (SELinux)

Protected APIs

- Goal: Protect system resources from unauthorized access

Protected APIs



- Linux DAC domain (UID)
- Linux MAC domain (SELinux)

Protected APIs (cont)

- Implementation in Android:
  - Protected APIs for ”risky” actions
  - Permission-based (mandatory) access control

Protected APIs(cont)

- What kinds of system calls on a smartphone would warrant protecting and why?

Examples of Protected APIs

"Risky"

- Changing device wallpaper, ringtone
- Making phone calls, sending SMS's
- Using camera, microphone, GPS
- Internet, wireless, Bluetooth access
- Reading/writing contacts, SMS log
- Rebooting device
- Factory reset

See also: http://developer.android.com/reference/android/Manifest.permission.html

Sensitive user data

- Subject to permissions checks:
  - Personal information (e.g. contacts)
  - Sensitive input devices (e.g.camera)
    - Location tracking can be manually disabled
  - Device metadata (e.g. logs,)

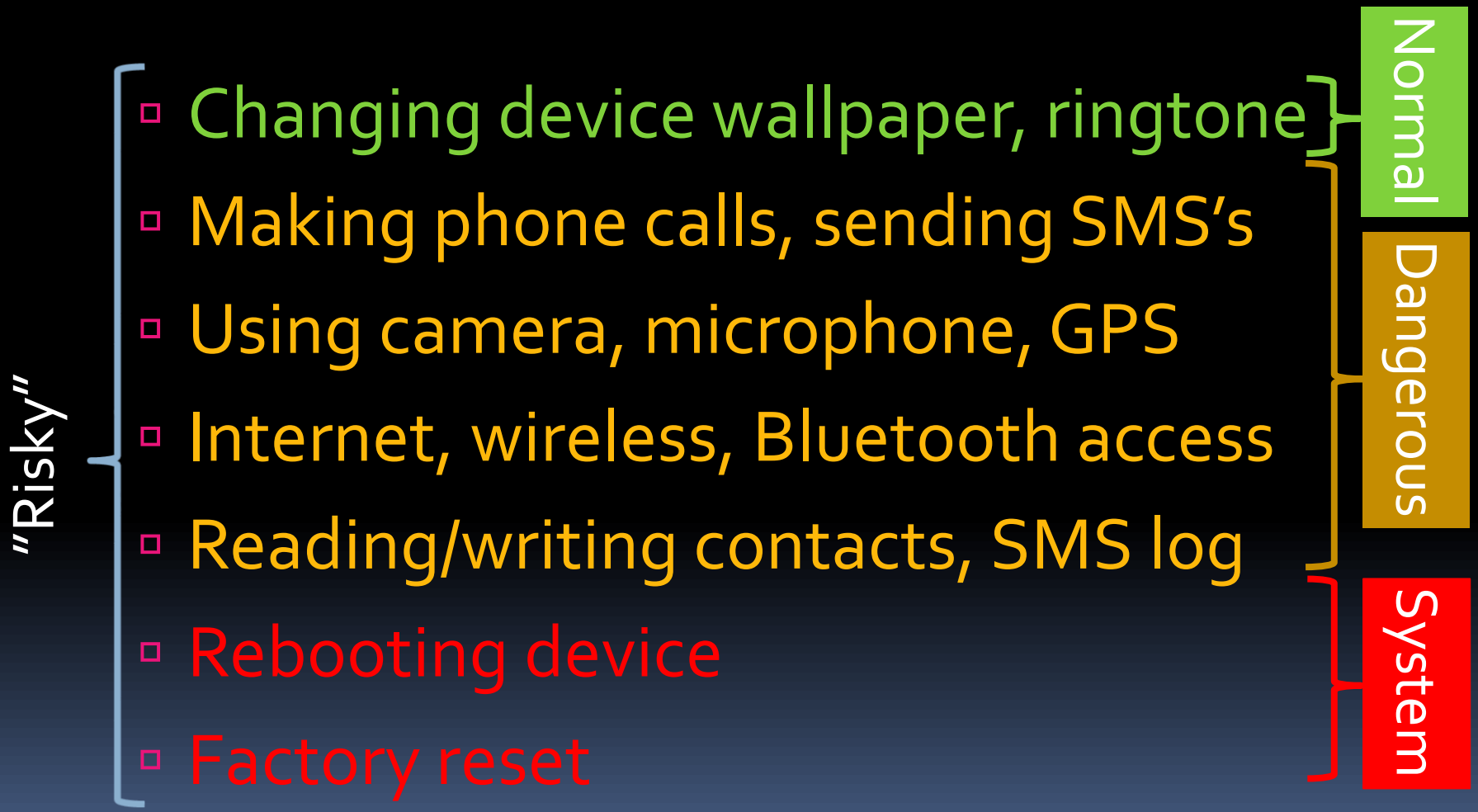
Access control & permissions

- Goal: Controls application access to protected APIs (and each other)
  - User agency vs. protecting system resources
  - Usability of security features

Android permissions

- 4 categories
  - Normal
  - Dangerous
  - System
  - Signature or System

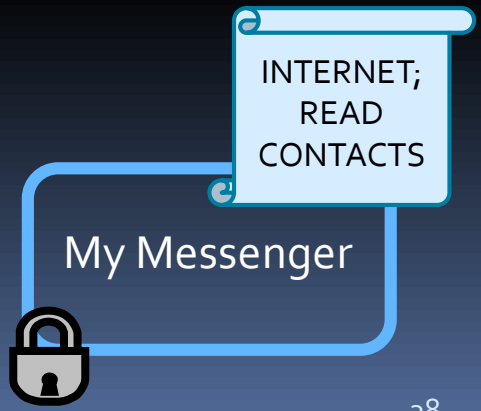
Examples of Protected APIs



See also: http://developer.android.com/reference/android/Manifest.permission.html

Permission assignment

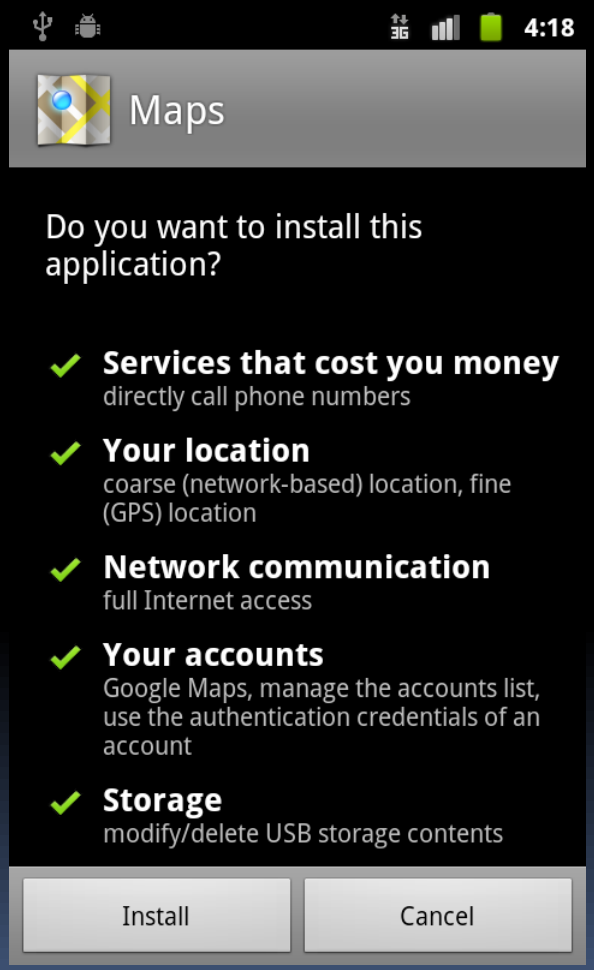
- Application declare all permissions in AndroidManifest.xml
- Permissions assigned to application UID
- Some permissions not user-grantable
  - Only available to pre-installed applications



Permission assignment

- **Normal** permission granted automatically
- **Signature** permissions granted if app signature matches the declarer of the signature
- **System** permission only assignable by OEM

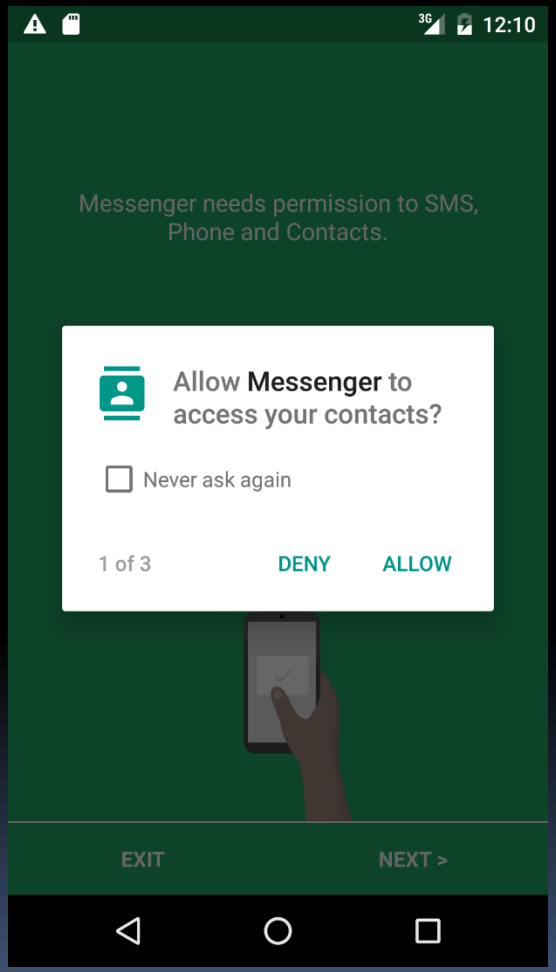
User approval (up to 5.1)



- **Dangerous** permissions require user approval at install time
- If not granted, application not installed at all!
- Granted for all users
  - Stored in `/data/system/packages.xml`

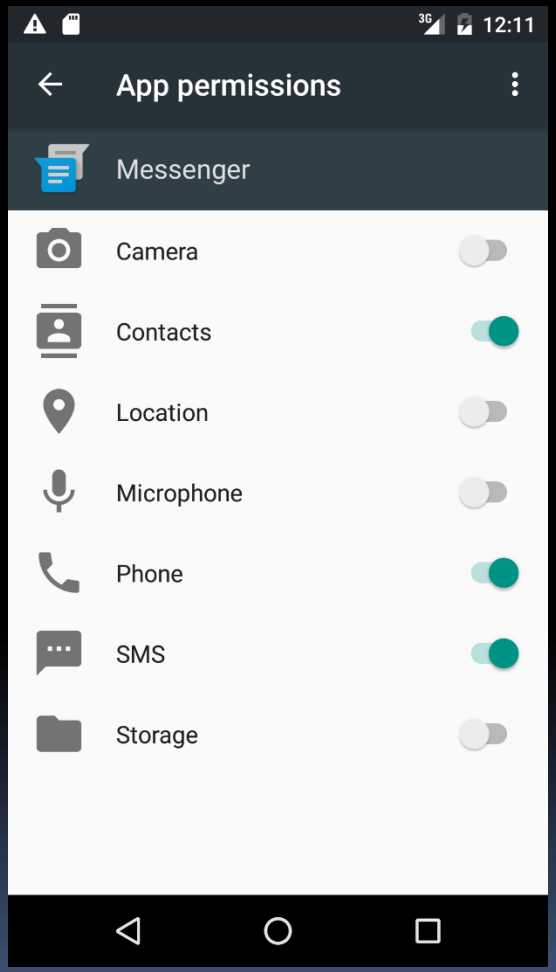
Image: http://source.android.com/devices/tech/security/

User approval (since 6.0)



- **Dangerous** permissions require user approval at runtime
- If not granted, application continues to run with limited capabilities
- Permission managed per application, per user
  - Stored in /data/system/users/<id>/runtime-permissions.xml

Permission revocation



- May be revoked later from application settings

Alternatives to obtaining permissions

- Delegate task to other application using *Intent*, e.g. invoke Camera app using `ACTION_IMAGE_CAPTURE` Intent
  - Caller does not need `CAMERA` permission
  - Caller cannot control the user experience, but does not have to provide UI for task
  - If no default app available, user is prompted to designate the handler

Android Open Source Project. Permissions Best Practices.

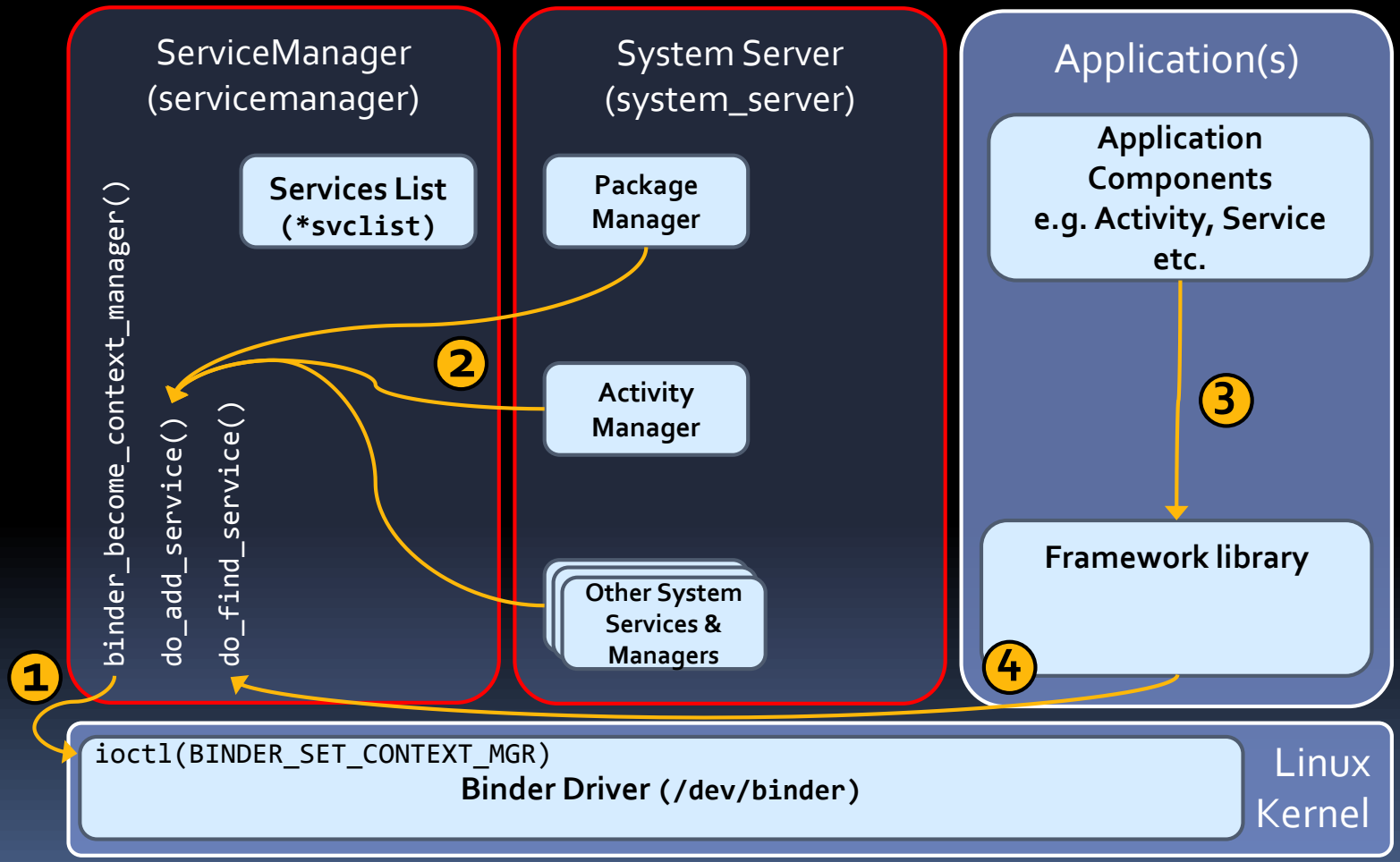
Intents

- Messaging object used for Inter-Component Communication (ICC)
  - Recall: activities, services, content providers, broadcast receivers
- Addressing
  - Explicit – fully qualified component name
  - Implicit – Intent filter declared in manifest
    - Provides a mechanism for late binding
- Pending Intents
  - Token-based access control delegation

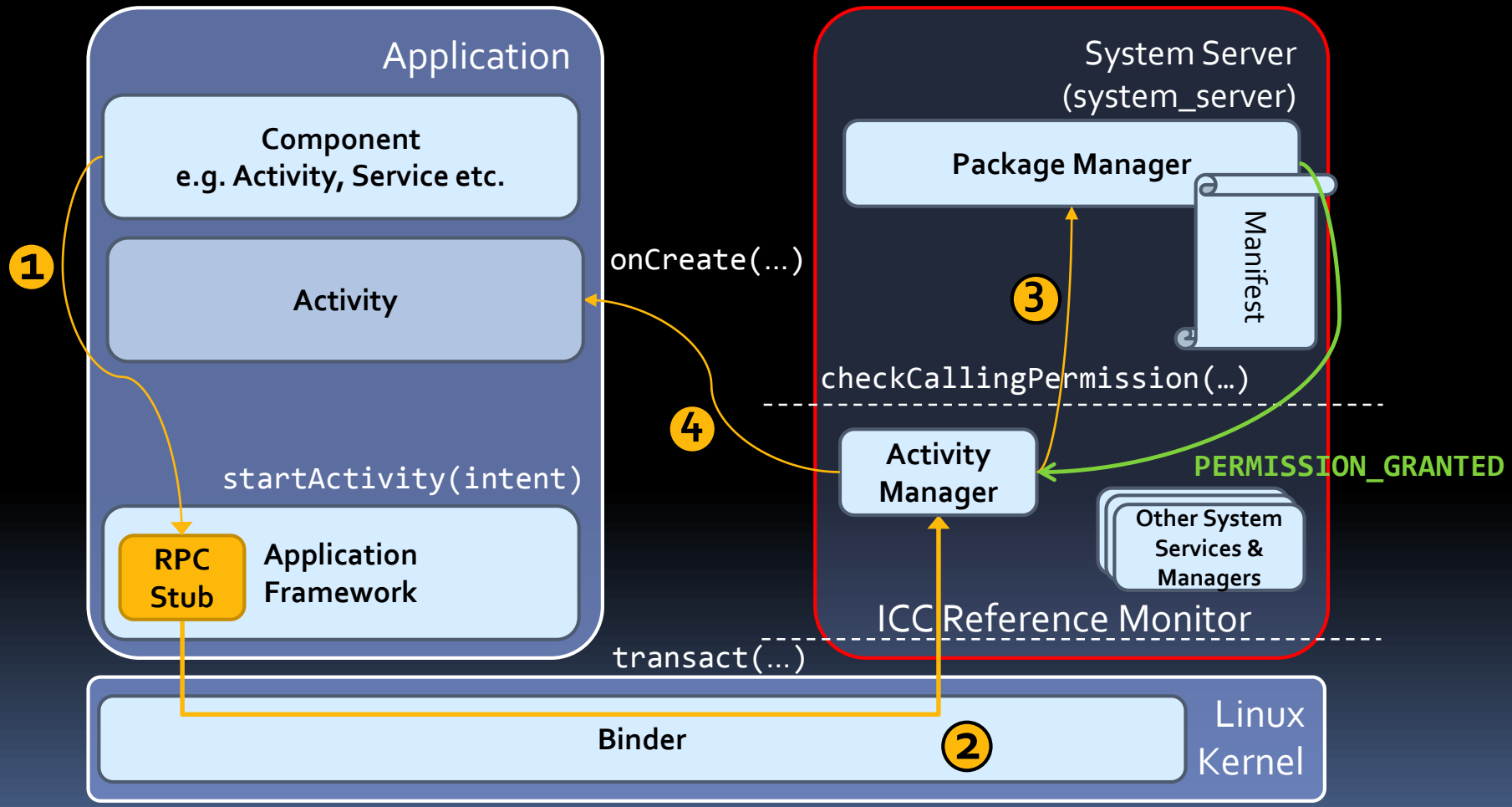
Binder

- IPC system for object-orientated operating system services (comp. CORBA/COM)
- Most underlying IPC based on Binder
  - Intents & content providers abstractions on top of Binder
  - Cf. local UNIX-domain sockets, signals, filesystem
  - Bionic libc doesn't support System V IPC
- Does not provide mediation by itself
  - Access mediated by system services

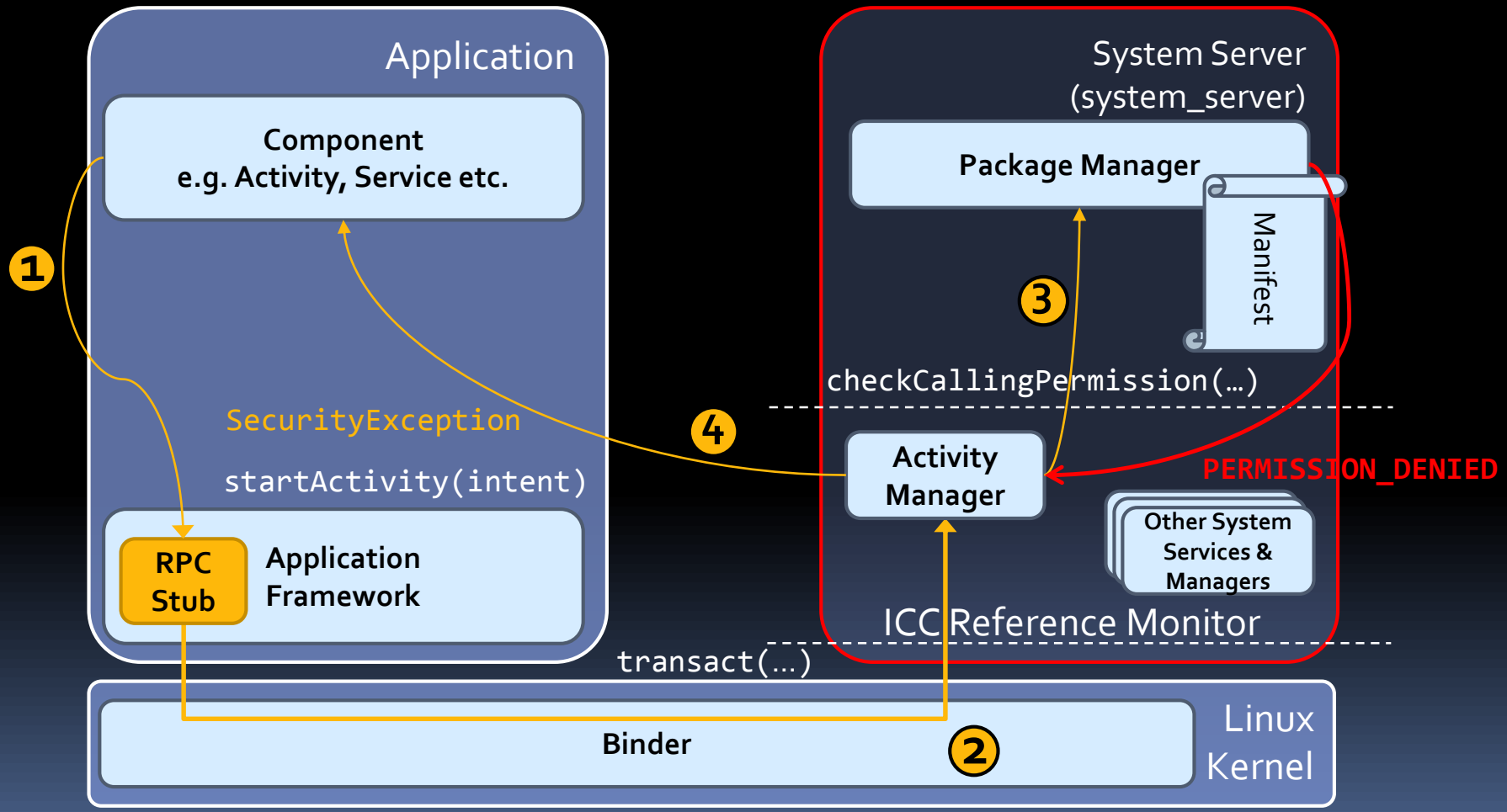
Binder Service Discovery



Starting an Activity



Starting an Activity



Authentication

- Keyguard
  - Pattern
  - PIN / Password
- Gatekeeper HAL (since 6.0)
  - Allows Keyguard to make use of native security features
- TrustAgentAPI (since 5.0)
  - Enables services that notify the system about whether they believe the environment of the device to be trusted

Android Open Source Project. Gatekeeper.
Elenkov. Password storage in Android M. 2015.
Elenkov. Dissecting Lollipop's SmartLock. 2014.

Authentication (cont.)

- Smart Lock Trust Agent (since 5.0)
  - Trusted Bluetooth device
  - Trusted NFC
  - Trusted place (via geofencing)
  - Facial recognition
  - On-body detection
- Fingerprint HAL (since 6.0)
  - Access to vendor-specific fingerprint hardware

Elenkov. Dissecting Lollipop's SmartLock. 2014.
Nexus Help: Set up your device for automatic unlock.
Android Developers. Creating and monitoring Geofences.

Hardware-based security features

- Goals:
  - Platform integrity
  - Secure storage

Hardware-based security features

- Implementation on Android:
  - Verified boot
  - Full-disk encryption
  - Keychain / Keystore

Keychain

- System credential store for private keys and certificate chains (since 4.0)
- KeyChain API is used for Wi-Fi and Virtual Private Network (VPN) certificates
- Hardware-backed keystore binds keys to device to make them non-exportable
  - `KeyInfo.isInsideSecureHardware()` (since 6.0) indicates if key is stored in hardware keystore

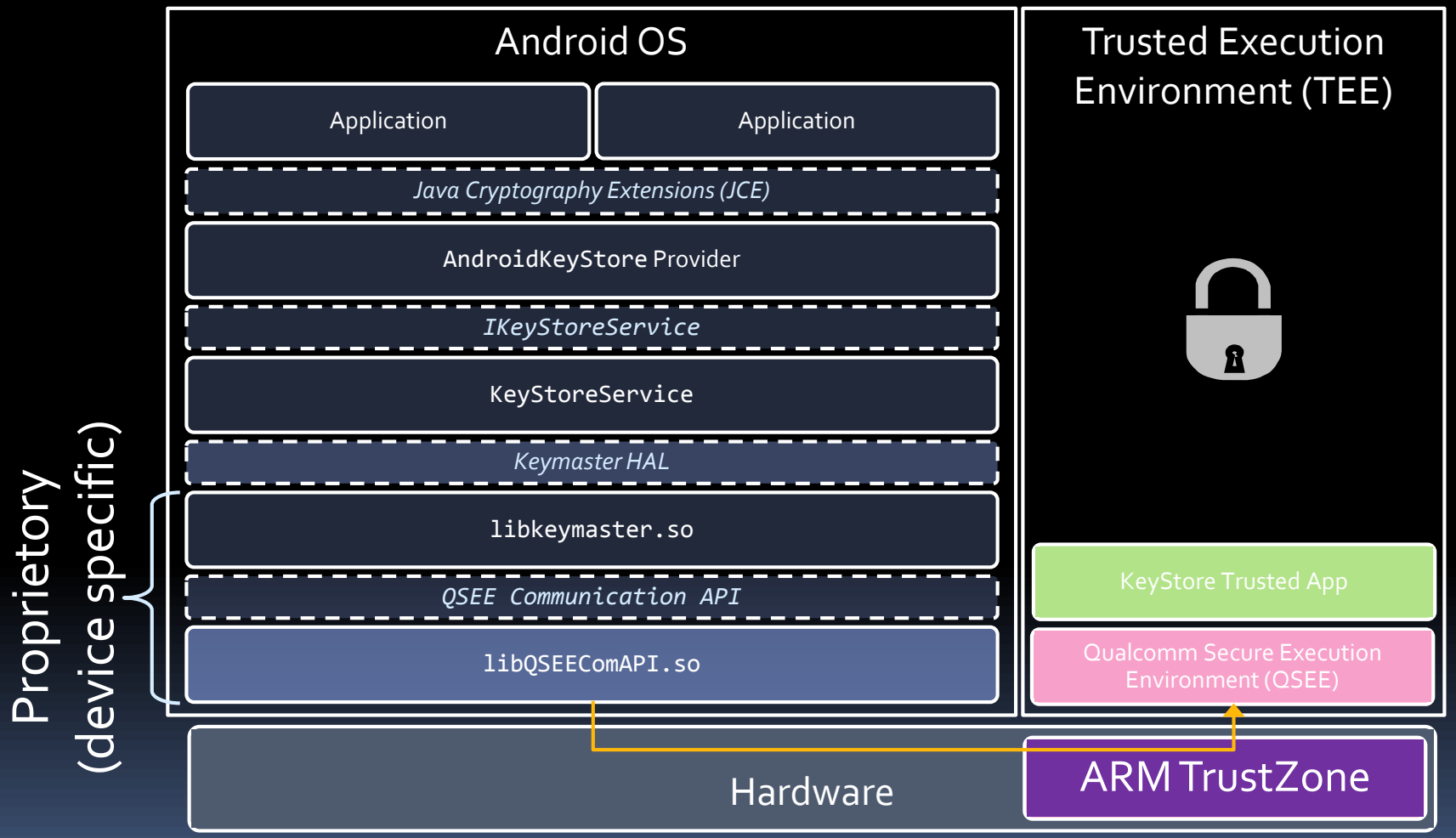
Google. Android for Work Security white paper. 2015

Keystore

- Keystore
  - For application-bound keys
  - Access via Java CE API
- KeymasterHAL
  - Access to hardware-backed keystore
  - Assymmetric key generation, signing and verification (since 4.1)
  - Binder IKeyStoreInterface (since 4.3)
  - Symmetric key support, access control, public key import and private / symmetric key import (since 6.0)

Android Open Source Project. Keymaster.
Elenkov. Keystore redesign in Android M.2015.
Elenkov. Credential storage enhancements in Android 4.3. 2013.

Keystore (cont.)



Elenkov. Keystore redesign in Android M.2015.
Elenkov. Credential storage enhancements in Android 4.3. 2013.

Full Disk Encryption

- Block-device encryption based on `dm-crypt`
- Encrypted on first boot (since 5.0)
- AES 128 CBC and ESSIV:SHA256
- DEK encrypted with AES 128
  - KEK derived from user PIN / password / pattern

  + hardware-bound key stored in TEE (since 5.0)
- Crypto acceleration through hardware AES (e.g. `dm-req-crypt`)

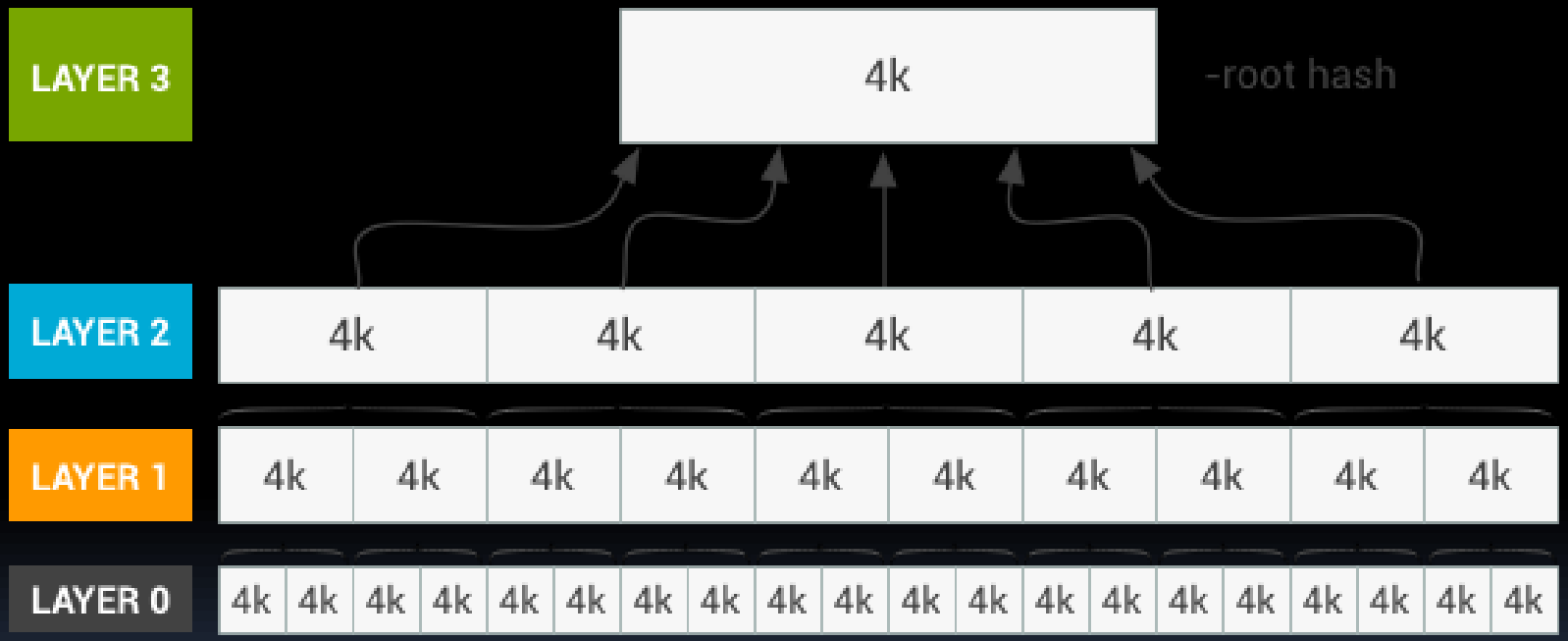
Android Open Source Project. Full Disk Encryption.
Elenkov. Hardware-accelerated disk encryption in Android M. 2015.

Verified Boot

- Based on `dm-verity` kernel feature
- Calculates SHA256 hash over every 4K block of the system partition block device
  - Hash values stored in hash tree
  - Tree collapsed into a single root hash
- Signature of the root hash verified with public key included on the boot partition
  - Must be verified externally by the OEM

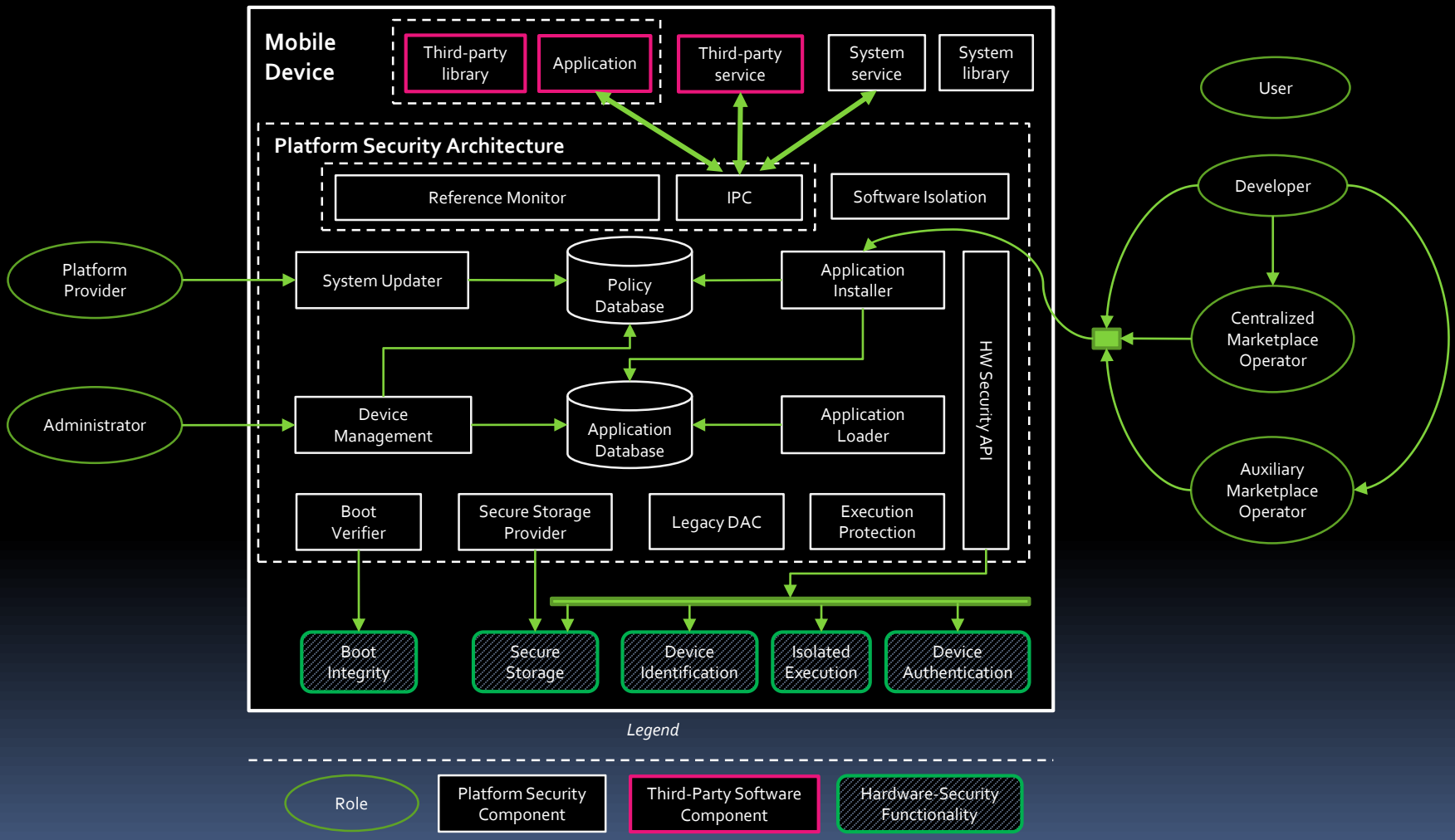
Android Open Source Project. Verified Boot.

Verified Boot Hash Tree



Android Open Source Project. Verified Boot.

Mobile Software platform security



Did you learn:

- Android as a software platform
  - Internals and surrounding ecosystem
- Security techniques in Android:
  - Application signing
  - Application isolation
  - Permission-based access control
  - Hardware-based security features

Contributors: Thomas Nyman, Sini Ruohomaa, N. Asokan

Plan for the course

- Lecture 1: Platform security basics
- Lecture 2: Case study – Android
- Lecture 3: Mobile software platform security in general
- Lecture 4: Hardware security enablers
- Lecture 5: Usability of platform security
- Invited lecture: SE Android policies
- Lecture 6: Summary and outlook