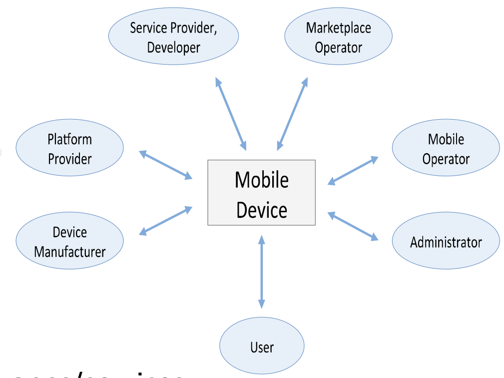
IS 2955 Special Topics: SAHI
Mobile Platform Security
Lecture 2.2

James Joshi
Professor,
School of Computing and Information
Sept 12, 2018

Mobile phone / Smartphone platforms security



Stakeholders

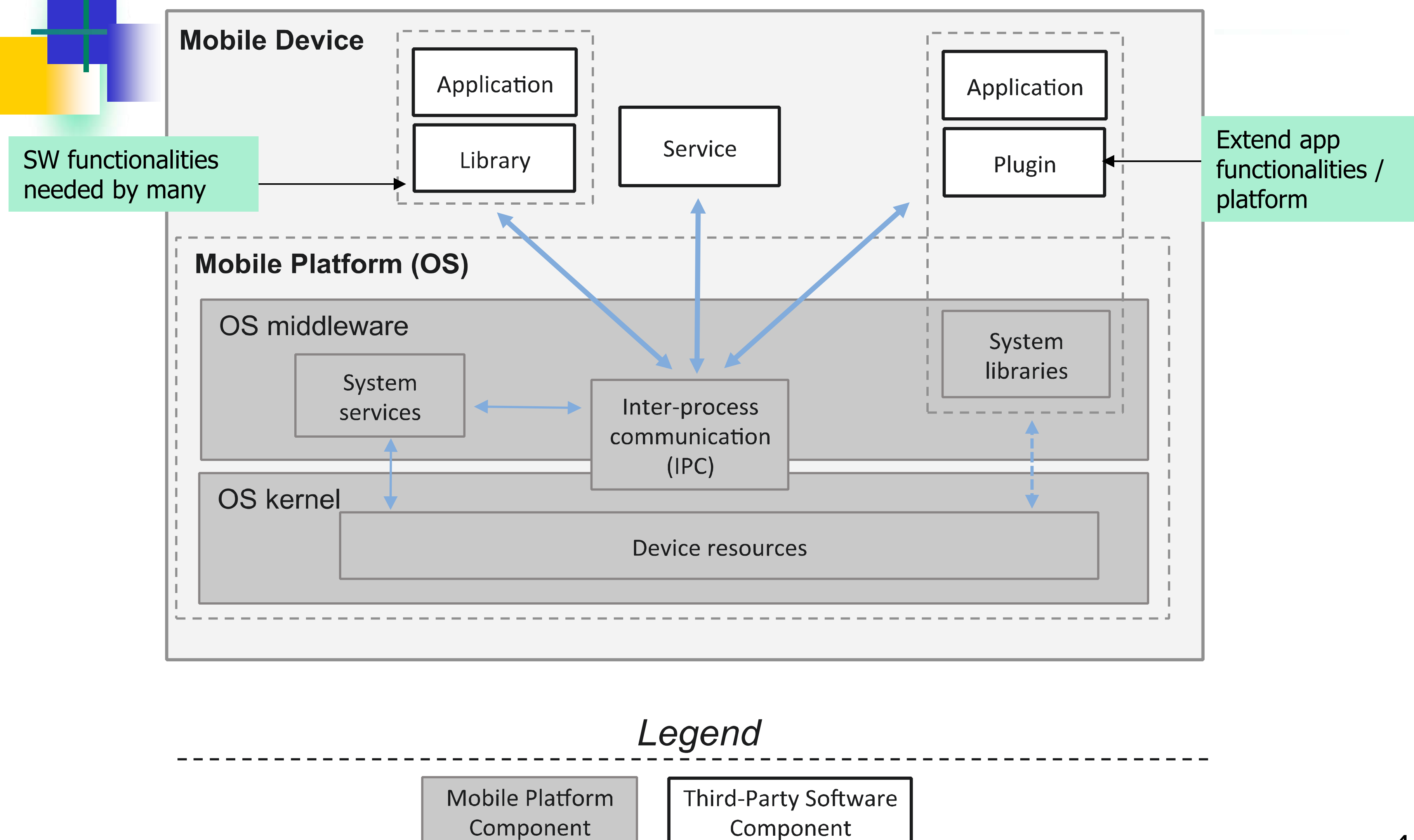
- Users –
  - need privacy of personal data (messages, profiles, contacts, location information)
  - Prevent misuse (e.g., unauthorized calls & SMS messages)
  - Protection against loss and theft (external / remote attackers)
- Manufacturers
  - Meet regulatory requirements or specifications -- device parameters protected (battery charging levels, wifi configuration, OS version, etc) from Users & external threats to users
- Mobile operators
  - Protect their business model – subscription control; control device functionality (e.g., tethering)
  - Adversary may be a device owner!!
- Service Providers and Developers
  - Primarily interested in the application data (may include copy-protected music)
  - Applications code needs to be protected from remote attacker
- Platform providers
  - OS and associated apps/services
  - PPs make app dev tools available, issue SW updates
  - Malicious app developer – primary adversary – exploit code vulnerabilities
  - Also device owners as adversary
- Marketplace operatorss
  - Distributes (App sores) – interested in protecting marketplace content
  - Key adversary – malicious developers who distribute malware infected SW / malware
- Administrators
  - Mobile devices may be owned by companies (for employees – for work+personal)
  - Confidential data needs to be protected
  - External attackers

Summary of stakeholders …

Table 2.1: Summary of mobile device stakeholders, their incentives and resources to protect, and adversarial models to consider

	Incentives	**Resources to protect**	**Primary adversary**	**Additional adversaries**
Users	preserve privacy, use device freely	private user data	remote attacker	attacker with temporary physical access
Manufac-turers	business model, regulatory requirements	device identifiers, configuration parameters, platform version	device owner	external attacker
Mobile operators	subscriber contract enforcement	usage of subsidized devices, mobile network resources	device owner	external attackers
Developers	mobile service protection	application data and code	remote attacker	device owner
Platform providers	business model	platform functionality	malicious developer	device owner
Marketplace providers	marketplace popularity	distributed applications	malicious developer	device owner
Administra-tors	company business model	company confidential data	remote attacker	attacker with temporary physical access

Mobile SW Architecture – a high level view



Platform Security Model

- Mobile Platforms – 2 SW components:
  - OS Kernel
  - OS Middleware
    - Set of libraries and services
- IPC Framework
  - For Communication between apps and services – uses API
  - Can be in kernel or middleware or both
  - Access to devices is mediated by IPC + services
    - e.g., **accessing GPS**: apps makes an IPC call to a system service – to get location API – it helps get device location by accessing GPS peripheral on the device via OS kernel;
    - direct access from apps to certain device resource may be allowed

A Mobile Platform Security Architecture Model – *from device manufacturers and platform providers*

Three basic functions

1. Software Isolation
   - Each app with its own execution and storage env
2. Access control model – IPC calls from Apps to services – permissions
   - AC Policy defined
3. Installed applications are cryptographically signed
   - Basis of permission assignment during app installation

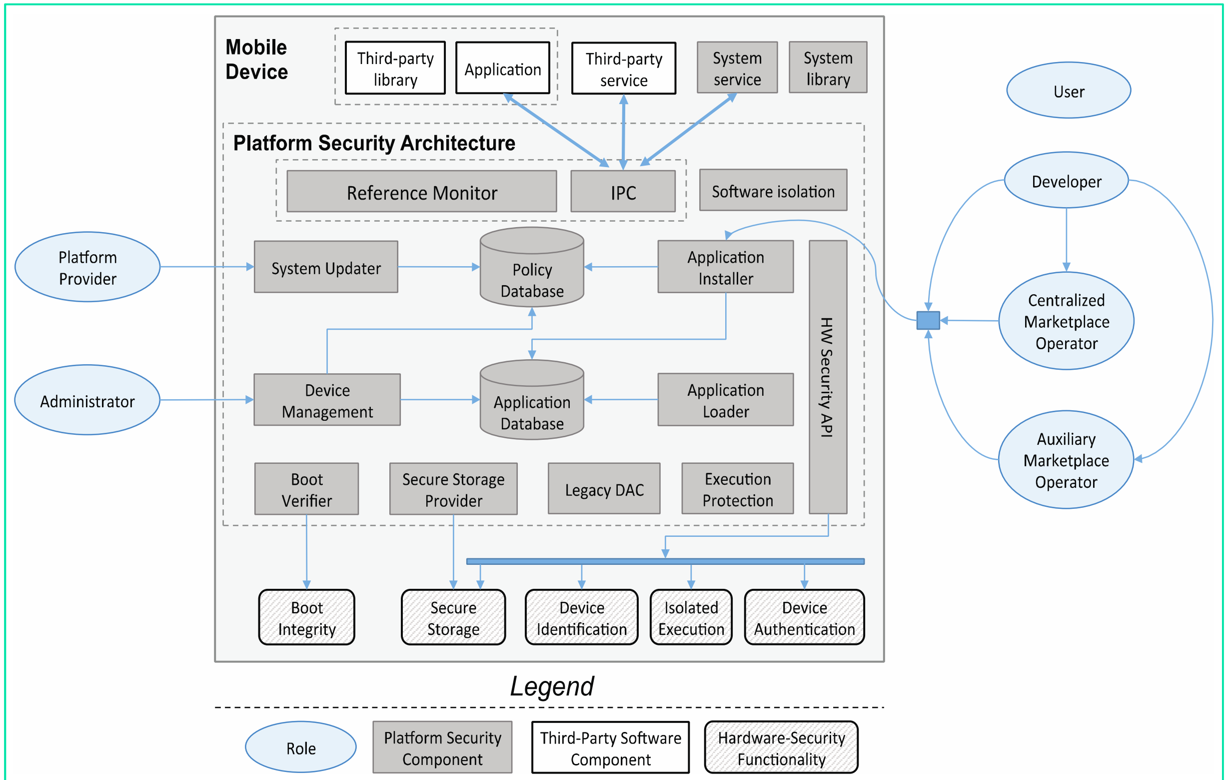


Figure 2.3: Mobile platform security architecture model.

Software Development/Deployment

- Distribution Model
  - Centralized marketplace or auxiliary marketplaces
  - Use application installer
  - Mobile platform may also allow direct application installation from developer -- *sideloading*
- Application signing – needed for installations
  - In centralized marketplace, CM provider does the signing – based on pub criteria
  - In auxiliary marketplace – AM provider signs (developer can sign – helps in *same origin policy* for update)
  - Can authenticate developers (can use external authentication; e.g., Credit Card) & issue developer identities
- Application identification
  - In centralized - signing authority may assign globally unique app ids
  - In auxiliary – ids are specific to that marketplace
  - Combination of signing key and marketplace-issued app id provides unique app identification
  - *Sideloading* – app ids must be picked by the developer (dev-issued)

Software Development/Deployment

- Permission request
  - For deployment of apps or TP service – developer defines the permissions that the app/service needs to access APIs that are protected with permissions
  - ***Manifest file***: configuration file in the service/app distribution package to request permissions – app installer uses this to assign the requested permissions during app installation
  - Permissions may be requested for libraries also
- Access control declaration
  - MP provider defines the permissions that are needed to use each service API call
  - TP service developers declare AC policies by defining permissions needed for each API call exposed by the service component – ***manifest file*** of the service.
- Access Control scope / granularity
  - Service/app developers may also declare AC policies for other types of resources – in addition to APIs; e.g, for data files created by the service
  - Fine-grained access policies may be needed – more permissions
    - E.g., separate permissions for each API call – better for principle of least privilege!

Application installation/update

- Permission Assignment
  - When an app is installed - ***application installer*** verifies the signature on the app & requested permissions from the *manifest file*.
  - App installer consults a *policy database* regarding the requested permissions and the signature.
  - *Policy database* contains
    - trust roots for signing authorities (typically, public keys of signing authorities) and
    - a list of permissions each authority is allowed to grant
  - may be solely based on application signing by trusted authorities or the installer may ask the user to authorize some of the requested permissions
  - *Application database*:
    - once verified- save the app executables, the set of assigned permissions and the application
- Permission presentation
  - coarse grouping may be used - when #permissions is large - based on data types (e.g., address book, emails, pictures, etc.)
- Application update - done through *app installer*
  - Checks if app distribution package is allowed to update the app specified in the **manifest file**
  - Verify that the update version is from the same developer

Runtime protection

- Runtime permissions
  - When an app/service is started, *app loader* uses permission database to associate the permissions to the process
  - *app loader* also links libraries to the process – once the app is loaded permissions remain constant.
  - Platform may allow apps/services to drop permissions, or gain more by loading a plugin
- Permission enforcement
  - Calls are processed by *reference monitor* (one or more)
  - When TP apps are allowed to make direct system calls – i.e., without using IPC calls – separate *RM*s for OS and IPC
  - *RM* may also *prompt* users at runtime
- Execution protection
  - Runtime software isolation and execution protection
    - Separate memory areas for processes (maybe randomized)

Runtime protection

- Application data protection
  - *Secure storage provider* enables isolated persistent storage areas for each application
    - Integrity protection (includes data freshness / replay protection)
    - Confidentiality protection
  - May use hardware-assisted secure storage functionality; fully software-based data protection may not be free of vulnerabilities (specially if the adversary has physical access to device)
- Hardware security APIs
  - SW based isolation mechanisms are vulnerable to implementation errors
  - Security-critical applications may thus require hardware-assisted isolated execution – hardware security architectures (e.g., ARM TrustZone):
    - Small pieces of security-sensitive code to be executed in isolation from the mobile OS
    - Hardware security API may provide an interface for isolated execution

Platform Management

- Platform boot integrity
  - All platform security components need to be protected –
  - They are stored in persistent storage
    - Attacker may bypass AC and other security mechanisms
    - E.g., tamper with app installer
  - Two approaches
    - *Hardware-assisted secure boot* –
      - uses platform verifier to check signatures over other platform security components
      - Does not prevent runtime modifications (use execution protection)
    - *Authenticated boot* -
      - DMs allow developers to create custom OS versions – but record measurements of the booted platform components to integrity protected hardware registers
      - Measurements can be used to enforce security decision during runtime !!

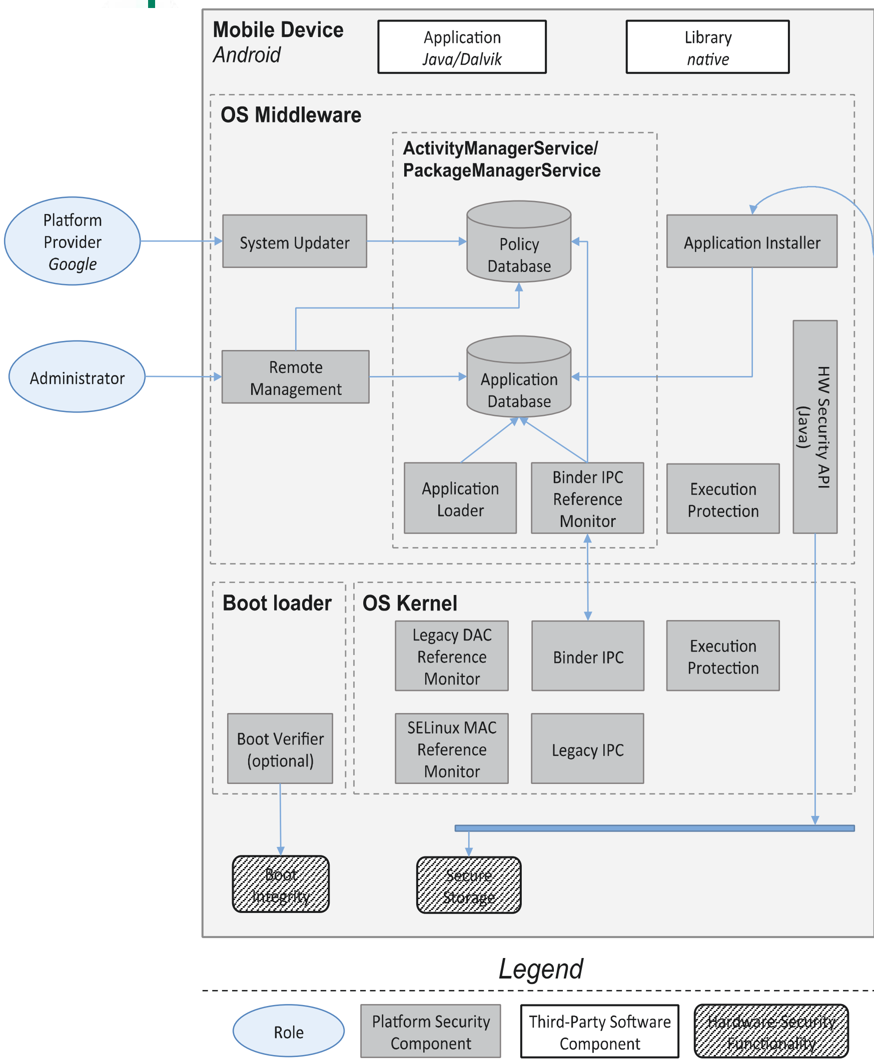
Platform Management

- Platform data integrity
  - Integrity of platform data is important – i.e., policy & app databases
  - the platform may support hardware-assisted secure storage (integrity protection), with possible replay protection mechanisms.
- Platform Updates
  - a *system updater* component authenticates system updates using trust roots and system update policies on the policy database.
  - In some platforms the system updater is part of the application installer implementation
- Device Management
  - Administrators can send device management commands
  - Device management component verifies commands using trust roots in policy databases
  - Commands for
    - Install new apps, remove apps, add or remove trust roots in PD

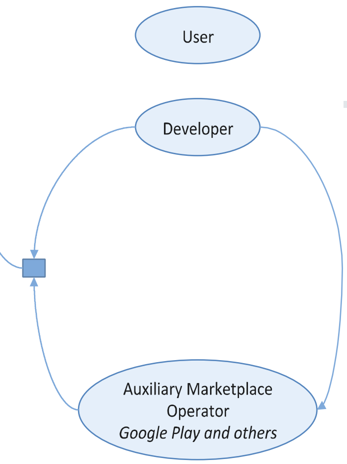
Mobile platforms

- Java ME, Symbian
- Android, iOS
- MeeGo, Windows Phone
- BlackBerry, Tizen
- Saifish OS, WebOS, FireFox OS
- …

Android

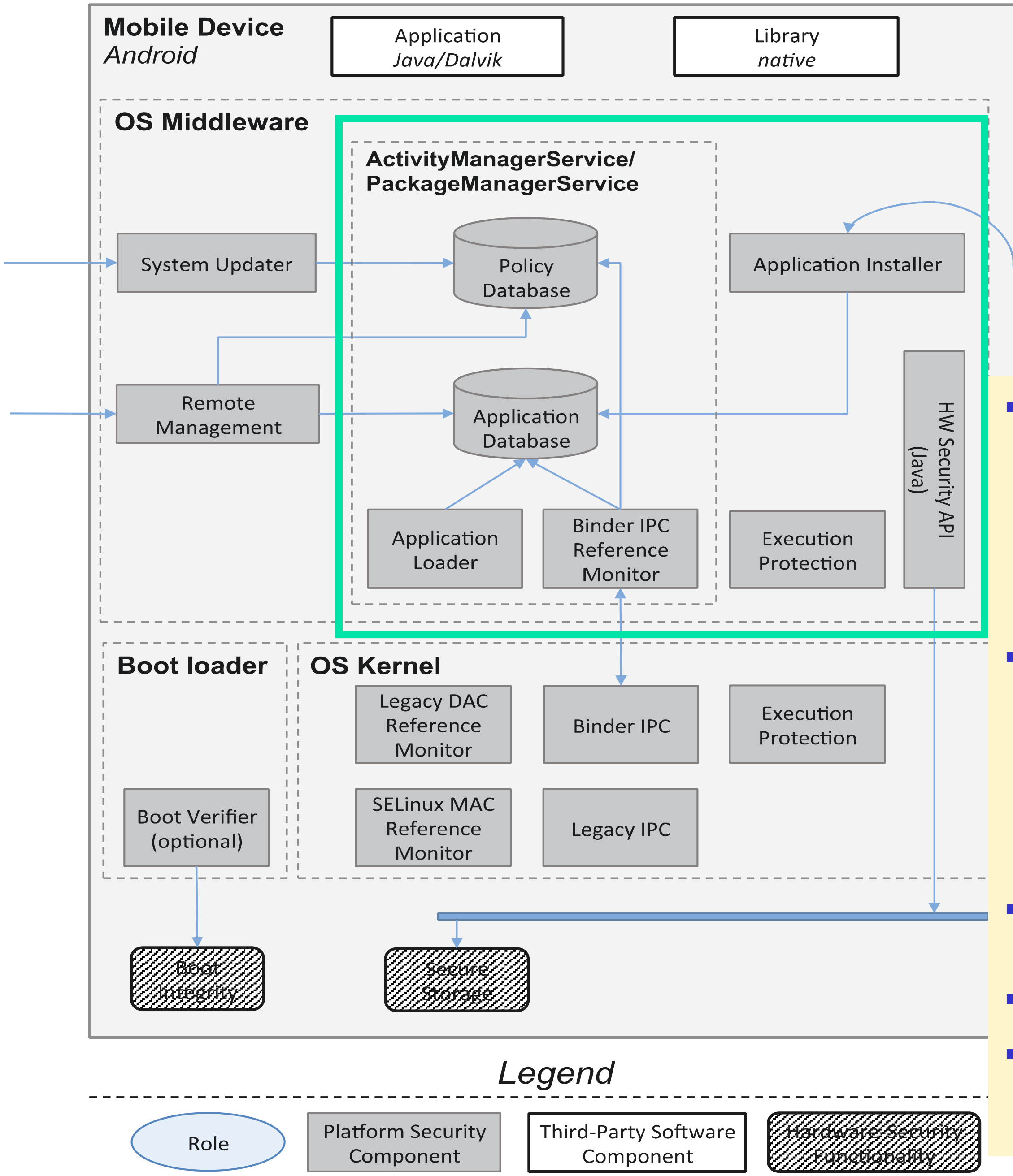


Open source smartphone platform from Google



- Based on modified Linux kernel
- Apps are sandboxed based on Linux DAC credentials
- TP apps cannot run with root ID
- Linux DAC acts as reference monitor – enforces separation of apps
- In each sandbox – an instance of register-based Dalvik/ART VM is executed
- App development is in Java mainly (native C/C++ libraries also deployed)

- Services perform non-interactive data processing,
- Content providers provide data sharing between apps
- Broadcast Receivers receive IPC messages
- Activities are software components with a user interface
- Android application components interact using IPC calls.
- Google Play – primary; but also have auxiliary/sideloading
- Android apps are signed by Developer

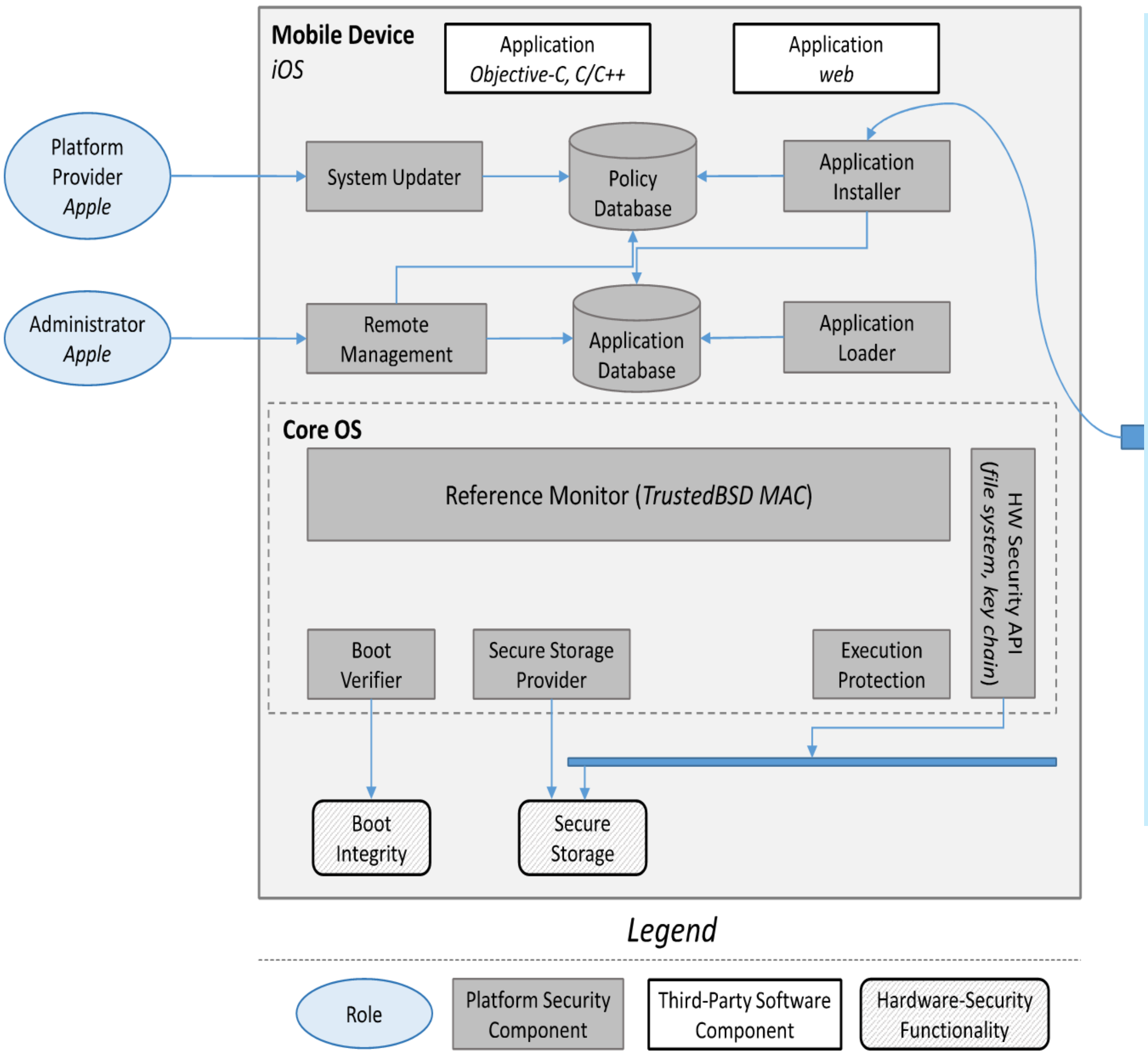


- Within sandbox, VM executes an Android System Server
  - A middleware component
  - SS sandbox has system privileges – can access protected device resources
  - TP apps issue IPC calls to SS components & pre-installed, privileged system apps that mediate system calls to OS
- IPC based on *Binder* – a re-implementation of the OpenBinder IPC framework
  - Core in the Linux kernel
  - Activity Management Service Component + Package Manager services in SS – act as primary RM to check permissions on Binder IPC
  - Legacy IPC is also supported
- Activity Manager service and other midlleware component act as application loader
- Mitigation for memory corruption attacks at run-time
- Ver 4.3 onwards support SELinux – adds MAC type enforcement (SEAndroid)

iOS Platform

- Mainly for iPhone, iPad, and iPod devices.
- TP app development is primarily done in *Objective-C*, although web applications running on top of the Webkit runtime are also supported
- App-specific libraries are allowed, but TP developers cannot deploy shared libraries or services.
- Distribution through centralized marketplace only, the *Apple Store*, that signs all applications,
- Access control enforcement is based on mandatory access control features of the TrustedBSD kernel
- All TP apps are assigned a single, *pre-defined sandboxing profile* that defines the assigned permissions for all applications; all apps are also assigned the same *user identifier*.
  - Run-time prompts for location info, contacts, reminders, calendar entries, mic, photos, etc.
- iOS 6 onwards, users can enable or disable access to private information for each application from the system settings.

iOS Platform Security Architecture



- Trusted BSD as RM
- Fine-grained AC rules based on system call arguments (e.g., file names)
- Supports enforcement of code signing
- App signatures are verified before app installation and execution
- Built-in apps have permissions for privileged tasks (*entitlements*)
- Dedicated apps for accessing security-sensitive system resources (messaging, cellular modem, calendar, etc.)
- Supports data/file encryption using hardware-resident, device-specific secret
- *Secure boot* is supported.

Comparison

- Software development security mechanisms
- Application Installation security mechanism
- Runtime protection mechanisms
- Platform management mechanisms

	Android	**iOS**
distribution model	multiple marketplaces, sideloading	centralized marketplace
application signing	developer signing	centralized signing
application identification	Linux user ID, package name	application identifier
permission request	applications, services	applications
access control scope	system APIs, application IPC	system APIs, file access
access control declaration	permissions, Linux AC, application identifiers	pre-defined profile
access control granularity		

	Android	**iOS**
permission assignment	user at installation	pre-defined profile
permission presentation	permission groups	permission names
application update	same-origin policy	central authority

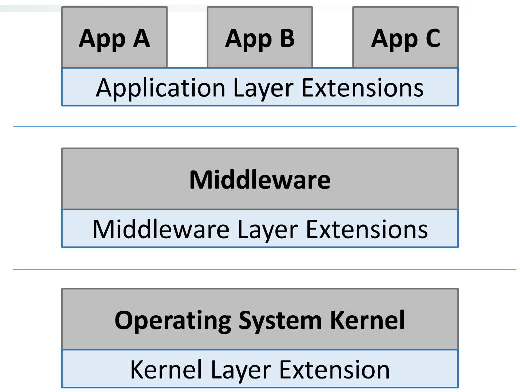
	Android	**iOS**
runtime permissions	constant (with exceptions)	increase by user approval
access control enforcement	reference monitor, callee, Linux DAC	reference monitor
execution protection	NX bit, ASLR	NX bit, ASLR, CSE
application data protection	dedicated directory and Linux access control	dedicated directory
hardware security APIs	secure stor (proprietar	

	Android	**iOS**
platform boot integrity	vendor-specific	secure boot
platform data integrity	Linux access control, UID-based sandboxing	dedicated directory, CSE
system updates	central signing	central signing
remote management	built-in features	built-in features

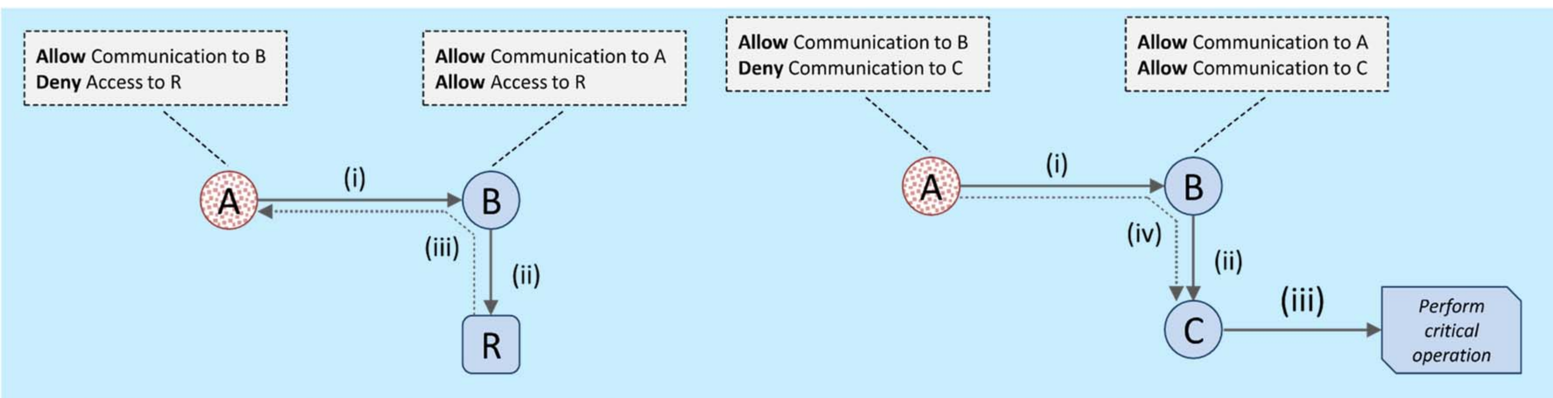
Android Based Attacks & Threats

- Privilege escalation attacks
  - Exploitation of *software* and *configuration* flaws to *elevate* the privileges of an app

Security extension target layers



Basic principles of application-level privilege escalation attacks



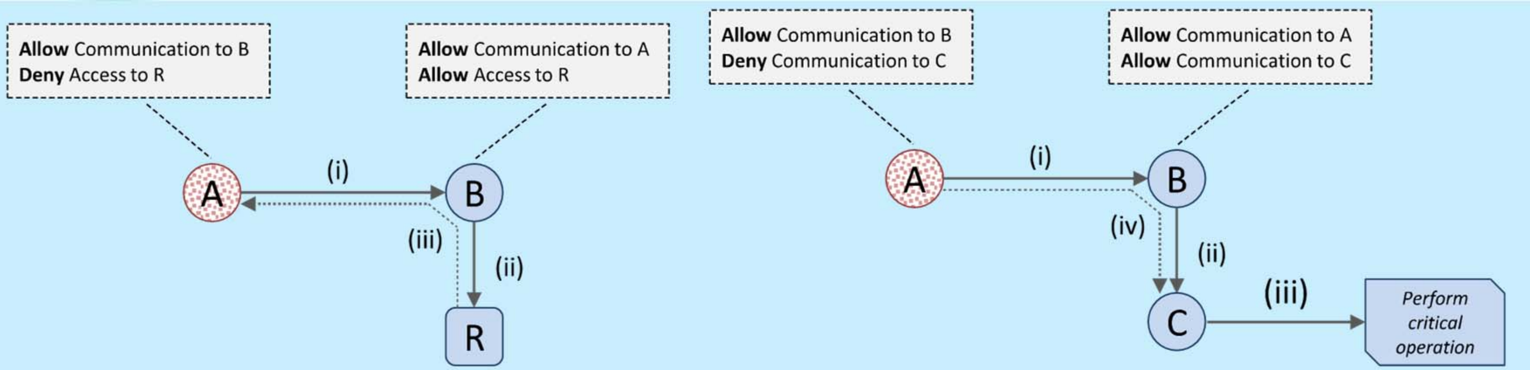
(a) Resource-sensitive privilege escalation

(b) Operational-sensitive privilege escalation

R can be a contact list; A exploits
B to receive it indirectly
A is malicious! B is vulnerable

Android Based Attacks & Threats

Basic principles of application-level privilege escalation attacks



(a) Resource-sensitive privilege escalation (b) Operational-sensitive privilege escalation

Confused deputy attack

- B is benign
- Does not enforce a permission check when A accesses its interfaces

B is simply acting as a deputy

Collusion attack

- B is malicious
- Merge their individual sets of permissions

High number of malware apps

- Open app ecosystem
- Through ad libs, repackaged apps, downloads, botnets
- ~520K new Android malware strains in first half of 2013 !!

Risky App Libraries

- App developers integrate ad libraries as part of app
- Hosting app and ad library share privileges – can be misused!

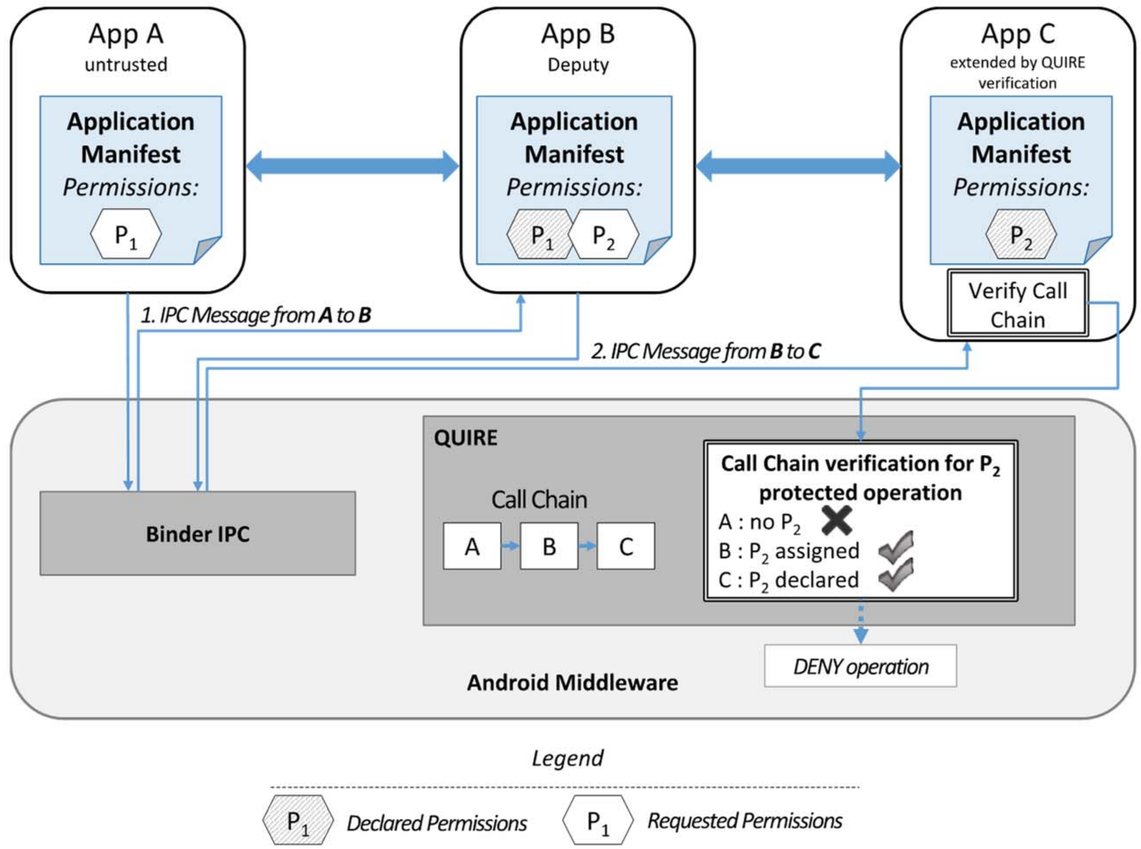
Risks

- accessing private user data (e.g., the user's call logs, phone number, browser bookmarks, or even the list of apps installed on the phone) ,
- deploying unsafe mechanisms
- directly fetch and run code from the Internet

Mitigation of *Confused Deputy* attack QUIRE

QUIRE: Lightweight provenance system for IPC

- Tracks and records the call chain of IPC calls
- Check originating app permissions
- Addresses vulnerable interfaces of trusted applications
- Can't stop address collusion attack – may forge the call chain



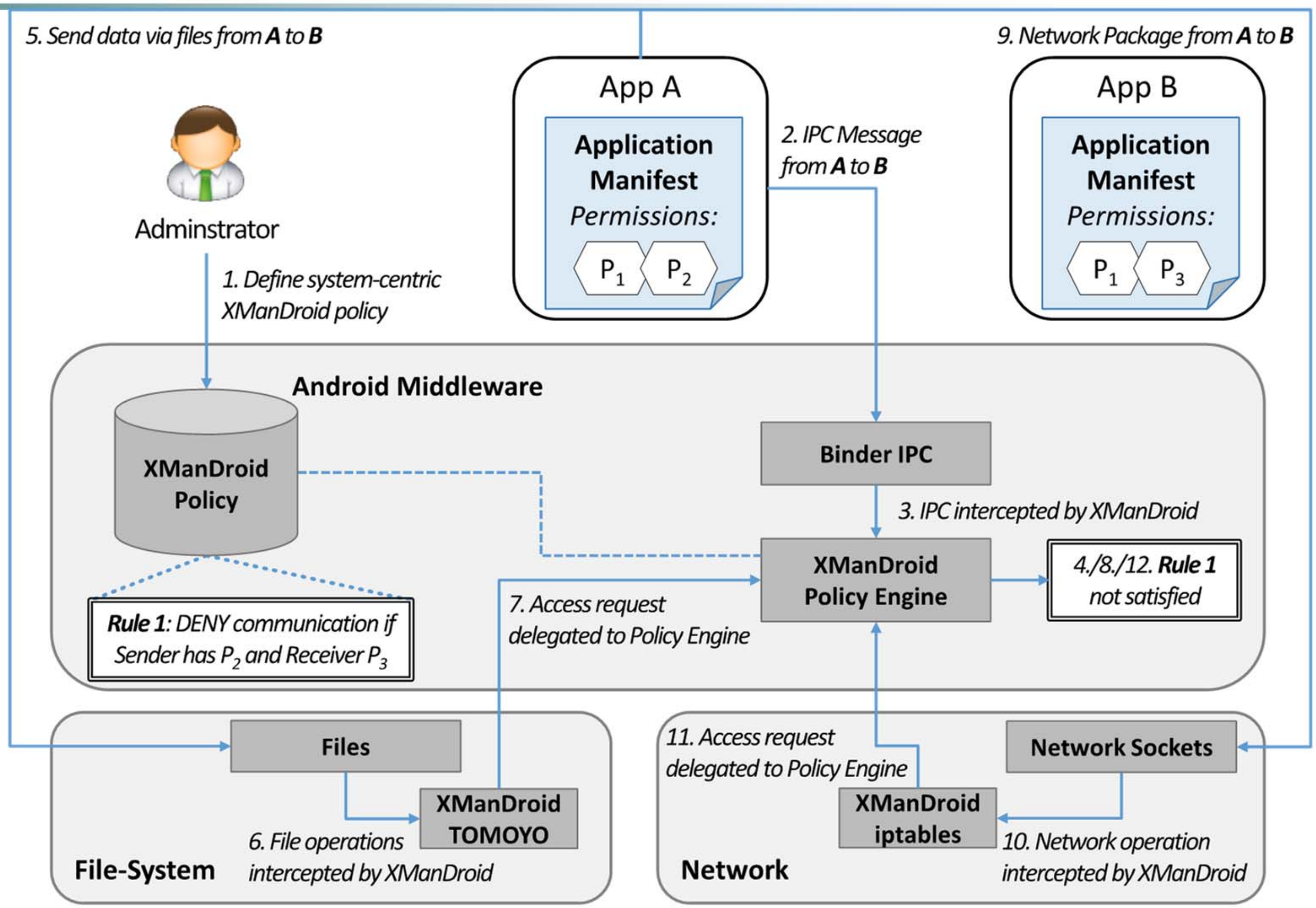
XManDroid

Addresses both confused deputy and some collusion attacks

Checks at runtime whether to allow a particular communication link

IPC, file access, network sockets

Needs to define policy



Summary

- Overview of platform security for Android and iOS based mobile platforms