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
Mobile Software platform security

Platform Security Architecture

- Reference Monitor
- IPC
- Software Isolation
- Policy Database
- Application Database
- Application Installer
- Application Loader
- Secure Storage Provider
- Legacy DAC
- Execution Protection
- Device Management

Legend

- Role
- Platform Security Component
- Third-Party Software Component
- Hardware-Security Functionality

- Boot Integrity
- Secure Storage
- Device Identification
- Isolated Execution
- Device Authentication

- Third-party library
- Application
- Third-party service
- System service
- System library

Mobile Device

- Platform Provider
- Administrator

Developer

Centralized Marketplace Operator

Auxiliary Marketplace Operator

System Updater

User
Mobile software platforms

- Which mobile platforms have you heard of?
Smartphone platforms

- Android: 84.4%
- iOS: 11.7%
- Windows Phone: 2.9%
- Blackberry: 0.5%
- Other: 0.6%

(IDC Oct 2014, shipments 2014Q3)
Android in a nutshell

- Linux-based (ARM & x86)
- Widely used for phones & tablets
  - Wearables, smart TVs, cameras (handheld) gaming consoles, etc.
- open-source software stack + closed source apps & 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.
## Comparison to desktop Linux

<table>
<thead>
<tr>
<th>OS</th>
<th>GNU/Linux</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>Linux</td>
<td>Linux</td>
</tr>
<tr>
<td>C library</td>
<td>GNU libc (a.k.a glibc)</td>
<td>Bionic libc</td>
</tr>
<tr>
<td>Application Runtime</td>
<td>SUS / POSIX-like and various more</td>
<td>Dalvik VM / ART</td>
</tr>
<tr>
<td>License(s)</td>
<td>GPL(v2/v3) / LGPL and various more</td>
<td>GPLv2 (kernel) BSD (bionic) Apache 2.0</td>
</tr>
</tbody>
</table>
Android software stack

Middleware

- Service Managers
- Content Providers
- Native Libraries
- Drivers
- Binder
- SELinux

Libraries

- Dalvik
- Core Libraries

Application Framework

- Applications
- Third-party Applications

Android Runtime

- System Applications

Linux Kernel

For an expanded image see e.g. [https://source.android.com/devices/tech/security/](https://source.android.com/devices/tech/security/)
Android software stack

For an expanded image see e.g. https://source.android.com/devices/tech/security/
Sample Application

For a more elaborate example, see Enck, Ongtang, McDaniel: Understanding Android Security, IEEE 2009 and package
Android application packages

APK
- META_INF
  - AndroidManifest.xml
- classes.dex
- lib
- res
- assets

- MANIFEST.MF
- CERT.RSA
- CERT.SF

- native code
- non-compiled resources
- application assets
Software distribution

- **Apps from multiple sources**
  - Google Play, auxiliary marketplaces, sideloading, pre-installed software

- **Marketplace services**
  - Discovery, installation, purchase
  - Community review, security scans, ...
Application signing

- Goal: same-origin policy for apps

Developer signs with private key

Signature verified with developer’s public key
Application signing

- Implementation in Android:
  - APK cryptographically signed by developer key
  - “Key A45B683C” does not imply “Jaana @ Rovio”
  - Developer signs with private key
  - Signature verified with developer’s public key
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)
- Kernel: mediation of Binder IPC
- Applications run in separate Dalvik virtual machine instances
Application Sandbox

Each app assigned a Unix UID

- UID owns:
  - Filesystem resources
  - Processes
  - Permissions (!)

- Apps from same developer may share UID sandbox
Application isolation

- Linux DAC domain (UID)

System

- Services

Third-party applications

- Applications
Rooting

- Rooting apps exploit vulnerabilities in privileged system daemons

- Note: boot-loader 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

- Implementation on Android:
  - Kernel-level MAC (SELinux) – Policies based on SELinux context
  - Middleware MAC (MMAC) – Policies based on package name and developer certificate

See also: Smalley, Craig: Security Enhanced (SE) Android paper (2013)
SE for Android

- Blocks many root exploits and misconfigurations
- Cannot protect against kernel exploits

See also: [Smalley, Craig: Security Enhanced (SE) Android paper (2013)](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4554863/)

27
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

- 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

Examples of Protected APIs

- Changing device wallpaper, ringtone (Low)
- Making phone calls, sending SMS’s (Low)
- Using camera, microphone, GPS (Med.)
- Internet, wireless, Bluetooth access (Med.)
- Reading/writing contacts, SMS log (High)
- Rebooting device (High)
- Factory reset (High)

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
Access control & permissions

- Implementation in Android:
  - Assign permissions at application installation time (with user consent)
  - Control at each API call / IPC message
Android permissions

- 4 categories
  - Normal
  - Dangerous
  - System
  - Signature or System
Examples of Protected APIs

- 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

Permission assignment

- Mandatory access control
- Apps declare permissions in manifest
- Permissions assigned to app UID
- Some permissions not user-grantable
  - Only available to preinstalled applications

INTERNET; READ CONTACTS
My Messenger
User Consent

- Normal permissions granted automatically
- Dangerous permissions require user confirmation

Services that cost you money
directly call phone numbers

Your location
coarse (network-based) location, fine (GPS) location

Network communication
full Internet access

Your accounts
Google Maps, manage the accounts list, use the authentication credentials of an account

Storage
modify/delete USB storage contents

User Consent

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

Do you want to install this application?

- **Services that cost you money**
  directly call phone numbers
- **Your location**
  coarse (network-based) location, fine (GPS) location
- **Network communication**
  full Internet access
- **Your accounts**
  Google Maps, manage the accounts list, use the authentication credentials of an account
- **Storage**
  modify/delete USB storage contents

Intents

- Messaging object used for Inter-Component Communication (ICC)
  - Recall: activities, services, broadcast receivers, content providers

- Addressing
  - explicit, implicit, pending
  - late binding via intent filters
Binder

- IPC system for object-orientated operating system services
- All underlying IPC based on Binder
  - Bionic libc doesn’t support System V IPC
  - Intents & content providers abstractions on top of Binder
- Does not provide mediation by itself
  - Access mediated by system services
Runtime access control

MyApp

startActivity(intent)

Reference Monitor

intent

Activity

Package Manager

checkCallingPermission(...)

PERMISSION_GRANTED

My App

Activity Manager
Runtime access control

MyApp

Package Manager

Activity Manager

Reference Monitor

startActivity(intent)

checkCallingPermission(...)

SecurityException

PERMISSION_DENIED
Runtime access control

MySender

SendBroadcast(intent, "ReceiveMyBroadcast")

myReceiver

pm.checkPermission("MyReceiver", "ReceiveMyBroadcast")

Permission Granted

Notification Manager

Package Manager

Reference Monitor

MyReceiver

SendBroadcast(intent, "ReceiveMyBroadcast")

pm.checkPermission("MyReceiver", "ReceiveMyBroadcast")

Permission Granted
Runtime access control

MySender

SendBroadcast(intent, "ReceiveMyBroadcast")

Package Manager

pm.checkPermission("MyReceiver", "ReceiveMyBroadcast")

PERMISSION_DENIED

Reference Monitor

MyReceiver
Runtime access control

Reference Monitor

Process

Domain

File

Type

SELinux policy

Linux kernel
Mobile Software platform security

Platform Security Architecture

Legend

Role
Platform Security Component
Third-Party Software Component
Hardware-Security Functionality
Did you learn:

- **Android as a software platform**
  - Internals and surrounding ecosystem

- **Security techniques in Android:**
  - Application signing
  - Application isolation
  - Permission-based access control

**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