You will be learning:

- What techniques are used in mobile software platform security?
- Is there a common general architecture?
- What is common/different among different systems?
  - Android, iOS, MeeGo (MSSF), Symbian
Mobile platform security

- Recall classes of basic security techniques:
  - Application isolation
  - Permission-based access control
  - Application signing
  - Hardware-based security features
Ecosystem stakeholders

- Developer
- Marketplace operator
- Mobile operator
- Administrator
- User
- Platform provider
- Device manufacturer

Think about threat models!

See also: Book section 2.1.
Modeling threats

Identify

- Assets and trust assumptions
- Potential adversaries
- Adversary capabilities/limitations
- Possible attack vectors

Cf. Software Security course
Ecosystem stakeholders

- Developer
- Marketplace operator
- Mobile operator
- Administrator
- User
- Device manufacturer
- Platform provider

See also: Book section 2.1.

Think about threat models!
Mobile platform architecture

Legend

Mobile Platform Component
Third-Party Software Component
Platform security architecture

Platform Security Architecture

- Reference Monitor
- IPC
- Software Isolation
- System Updater
- Policy Database
- Application Installer
- Application Loader
- Application Database
- Legacy DAC
- Execution Protection
- Secure Storage Provider
- Secure Storage
- Device Identification
- Isolated Execution
- Device Authentication
- Boot Verifier
- Boot Integrity
- Secure Storage
- Device Identification
- Isolated Execution
- Device Authentication

Legend

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

User

Developer

Centralized Marketplace Operator

Auxiliary Marketplace Operator

Platform Provider

Administrator
Step 1a: Developer publishes an application

Developer requests permissions for his application.

In some platforms the application can be directly “sideloaded” to the mobile device.

Some platforms support auxiliary marketplaces.

Legend:
- Role

Developer submits the application to a centralized marketplace.

Centralized Marketplace Operator

Auxiliary Marketplace Operator
Step 1b: Marketplace signs the application

Marketplace provider checks the application (and requested permissions) and signs the app package.

In some platforms the developer signs the app package.
Step 2: Application installation

Mobile device receives an application installation package from a marketplace (or developer).

Installer may prompt the user to accept some of the requested permissions.

Application installer component needs integrity protection.

Permission and policy databases need integrity protection.

Legend:
- User
- Developer
- Centralized Marketplace Operator
- Auxiliary Marketplace Operator

Installer consults local policy database about requested permissions.

Installer checks application signature and requested permissions.

Installer stores assigned application permissions.

Application installer component needs integrity protection.

Permission and policy databases need integrity protection.

Mobile device receives an application installation package from a marketplace (or developer).
Step 3a: Application loading

Loader attaches permissions to the started process

Loader reads permissions from the permission database

Integrity of installed application binaries

Loader component needs integrity protection

Third-Party Software Component

Platform Security Component

Legend

Application Installer

Policy Database

Application Loader

Application Database

Secure Storage Provider

Secure Storage

Boot Verifier

Boot Integrity

User

Developer

Centralized Marketplace Operator
Step 3b: Application execution

Reference monitor checks permissions to control access to system resources.

Applications and platform need to communicate with each other and HW.

Some applications need secure state (e.g., DRM).

Some applications need secrecy for their persistent storage.

OS/HW isolate applications from one another at runtime.

Some applications need device identification and authentication (e.g., provisioning).
Step 4: System updates

System updater verifies received update using policy database

System updater rewrites parts of system software

Platform providers issue (signed) system updates

Platform Provider

Administrator

Administrators may update device policies and applications

System updates need secure state to prevent rollbacks to previous system version

System updates may need device identification

Legend

Platform Security Component

Third-Party Software Component

Device Management

Application Database

Policy Database

System Updater

Application Installer

Application Loader

Boot Verifier

Secure Storage Provider

Legacy DAC

Execution Protection

Boot Integrity

Secure Storage

Device Identification

Isolated Execution

Device Authentication

System Updater Provider

Centralized Marketplace Operator

Auxiliary Marketplace Operator

Developer

User
Recap: main techniques

Platform Security Architecture

1. Permission request
2. Application signing
3. Permission assignment
4. Permission-based access control
5. Application isolation
6. API to system functionality (e.g. secure storage)

Legend
- Role
- Platform Security Component
- Third-Party Software Component
- Hardware-Security Functionality
Platform security architecture

Legend
- Role
- Platform Security Component
- Third-Party Software Component
- Hardware-Security Functionality
Why Generalize?

Mobile platforms revisited

- Android ~2007
- Java ME ~2001
  - “feature phones”: 3 billion devices!
  - Not in smartphone platforms
- Symbian ~2004
  - First “smartphone” OS
Mobile platforms revisited

- iOS ~2007
  - iP* devices; BSD-based
- MeeGo ~2010
  - Linux-based
  - MSSF (security architecture)
- Windows Phone ~2010

...
Symbian

▪ First widely deployed smartphone OS
  □ EPOC OS for Psion devices (1990s)

▪ Microkernel architecture:
  □ OS components as user space services
  □ Accessed via inter-process communication (IPC)
Symbian Platform Security

- Introduced in ~2004
- Apps distributed via Nokia Store
  - Sideload support
- Permissions are called ‘capabilities’, fixed set (21)
  - 4 Groups: User, System, Restricted, Manufacturer
Symbian Platform Security

Applications identified by:

- UID from protected range, based on trusted code signature
- Or UID picked by developer from unprotected range
- Optionally, vendor ID (VID), based on trusted code signature
Apple iOS

- Native application development in Objective C
  - Web applications on Webkit
- Based on Darwin + TrustedBSD kernel extension
  - TrustedBSD implements Mandatory Access Control
  - Darwin also used in Mac OS X
iOS Platform Security

- Apps distributed via iTunes App Store
- One centralized signature authority
  - Apple software vs. third party software
- Runtime protection
  - All third-party software sandboxed with same profile
  - Permissions: “entitlements” (post iOS 6)
  - Contextual permission prompts: e.g. location
MeeGo

- Linux-based open source OS
  - Intel, Nokia, Linux Foundation
  - Evolved from Maemo and Moblin

- Application development in Qt/C++

- Partially buried, but lives on
  - Linux Foundation shifted to HTML5-based Tizen
  - MeeGo -> Mer -> Jolla’s Sailfish OS
MeeGo Platform Security

- Mobile Simplified Security Framework (MSSF)
  - Permissions: “resource tokens”
  - Enforced via “Smack”
  - Apps identified by signatures from “software sources”
  - Policy specifies privileges grantable by software sources
Model for platform security

Four processes to protect:

1. Software deployment
2. Application installation
3. Runtime operation
4. Platform management
1. Software deployment

Developing and publishing

- Design choices:
  - Distribution: centralized vs. decentralized
  - App signing: certified vs. self-signed
1. Software deployment

- Design choices:
  - App identification: global vs. local
  - ...

## Software deployment

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distribution model</strong></td>
<td>Multiple marketplaces, sideloading</td>
<td>Centralized marketplace</td>
<td>Multiple marketplaces, sideloading</td>
<td>Centralized marketplace, limited sideloading</td>
</tr>
<tr>
<td><strong>Application signing</strong></td>
<td>Developer signature</td>
<td>Centralized signature</td>
<td>Marketplace and developer signature</td>
<td>Centralized or developer signing: affects set of permissions</td>
</tr>
<tr>
<td><strong>Application identifier</strong></td>
<td>Package ID, local Linux UID for permissions</td>
<td>Application ID</td>
<td>3-part ID: Marketplace - package - application</td>
<td>Application ID, vendor ID</td>
</tr>
</tbody>
</table>
2. App installation

Acquiring/installing a new app

- Design choices:
  - Permission assignment: user vs. authority?
  - Permission granularity?
  - Application updates: same origin vs. centrally authorized?
App permissions

- Ask user?
  - Install or use time?
- Automatic granting?
- Revocation?
- What about libraries?

Symbian

iOS

Android
### App permissions

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Granularity</strong></td>
<td>Fine-grained</td>
<td>Pre-defined profiles (iOS6 entitlements)</td>
<td>Fine-grained</td>
<td>Coarse-grained</td>
</tr>
<tr>
<td><strong>Assignment</strong></td>
<td>Ask user or app signature</td>
<td>Fixed profile for all apps</td>
<td>Marketplace-specific rights profiles</td>
<td>Ask user or centralized signature</td>
</tr>
<tr>
<td><strong>Ask user: presentation</strong></td>
<td>by group (11), install time &amp; runtime (6.0)</td>
<td>by name, runtime</td>
<td>never ask</td>
<td>by name (21), install time</td>
</tr>
</tbody>
</table>

Both Android (> 6.0) and iOS allow revocation of granted permissions
# App permissions

## Runtime permission changes

<table>
<thead>
<tr>
<th>Changes in process permissions at runtime</th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre 6.0: Constant (except URI permissions) &gt; 6.0: User can change rights</td>
<td>User can change rights</td>
<td>Rights can increase by plugin loading, <strong>decrease by request</strong></td>
<td>Constant (library loading can fail)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permissions of libraries</th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td>App’s permissions</td>
<td>App’s permissions</td>
<td>Union of app and library permissions</td>
<td>App’s permissions (library perms must be a superset)</td>
<td></td>
</tr>
</tbody>
</table>
App updates

- Who can update an app?
  - Same-origin: same dev. key
  - Trusted marketplace(s)
  - Allow anyone

<table>
<thead>
<tr>
<th>Updates allowed if...</th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Same-origin: must match old</td>
<td>Centrally signed</td>
<td>Marketplace’s trust level high</td>
<td>Protected? Same-origin;</td>
</tr>
<tr>
<td></td>
<td>version’s developer key</td>
<td></td>
<td>enough</td>
<td>Unprotected? Anyone</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. Runtime operation

- Design choices:
  - Permission enforcement: where?
  - App data protection: how to secure storage?
## Runtime operation

- **Access control enforcement: where is “reference monitor”?**

<table>
<thead>
<tr>
<th>Where is access control enforced?</th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UID/GID-based in kernel + IPC access control in Binder + application code</td>
<td>Centralized</td>
<td>D-Bus framework + socket IPC in kernel + application code</td>
<td>Reference monitor for IPC calls + application code</td>
</tr>
</tbody>
</table>
Protecting data & code

- Applications: isolation for data access
- Platform: executables (see later)

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iOS</th>
<th>MSSF</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application data integrity</td>
<td>Own directory and Linux access control</td>
<td>Access to own directory only</td>
<td>Permission-based policies</td>
<td>Own directory</td>
</tr>
</tbody>
</table>
4. Platform management

Bootup, platform integrity, updates

- Design choices:
  - Boot integrity: secure vs. authenticated?
Secure boot vs. authenticated boot

Secure boot

Authenticated boot
Boot integrity

- **Secure boot**
  - Only authorized images can be booted
- **Authenticated boot**
  - Access levels depend on booted image

<table>
<thead>
<tr>
<th>Platform boot integrity</th>
<th>Android</th>
<th>iOS</th>
<th>MeeGo</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vendor-specific</td>
<td>Secure boot</td>
<td>Secure boot, authenticated boot</td>
<td>Secure boot</td>
</tr>
</tbody>
</table>
Platform data integrity

<table>
<thead>
<tr>
<th>Platform data integrity</th>
<th>Android</th>
<th>iOS</th>
<th>MeeGo</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linux A/C, SELinux, UID-based sandboxing</td>
<td>Dedicated directory, code signing enforcement</td>
<td>Linux A/C, Smack, IMA, EVM</td>
<td>Dedicated directory</td>
</tr>
</tbody>
</table>

- IMA: Integrity measurement architecture
- EVM: Extended validation module

(see “An Overview of The Linux Integrity Subsystem”)

The big picture

Recurring common themes

- Permission-based security architectures
  - VAX/VMS privileges (~1970’s): adapted for applications
  - Code signing (mid 1990’s): adapted for application installation

- Application/process isolation
The big picture

Different choices in the design space lead to different architectures

Open issues remain: can you think of some?

- More in Lecture 6
Did you learn:

- Generalization of mobile software platforms
  - Key security techniques and general architecture
- Comparison of four systems
  - Android, iOS, MeeGo (MSSF), Symbian

Contributors: Kari Kostiainen, N. Asokan, Sini Ruohomaa
Plan for the course

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