Towards Releasing Your Mobile App

Mobile Application Development

LTAT.06.021

Jakob Mass
You built the app and want to show it to the world!

**Before release, we should**..

- .. ensure high quality
  - Software quality checklist
  - Security checklist

- .. prepare files, resources needed to release
  - Light-weight, cryptographically signed

- .. manage app store requirements
  - Dev. license, policy alignment, promo materials

**After release**: monitoring & feedback
Ensuring app quality
App quality - checklists

- Systematic, documented approach
  - Set of criteria + test cases
  - Ensuring user-experience, performance

- Existing **checklists** for this purpose:
  - E.g. security checklists
  - Store policy (later)

- Principle:
  - Adopt test cases to your app, perform the checks
  - Document gaps found
  - Analyse & fix
Finding checklists and guidelines

We will look at examples from

- Android Core App Quality checklist
- OWASP Security checklist

Some more resources at:

- Design guidelines also exist
  - Material design
  - Apple design guidelines
- ....
Android core app quality checklist

- Covers:
  - Audio, Media, Sharing, Background Services
  - Stability, Performance, SDK, Battery
  - Permissions, Data & Files, Identity, App Components, Networking, WebViews, Execution, Cryptography

- Example:
  
<table>
<thead>
<tr>
<th>PS-T2</th>
<th>SP-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The app targets the latest Android SDK needed to align with Google Play requirements by setting the targetSdk value.</td>
<td></td>
</tr>
</tbody>
</table>

- Each criteria has associated test procedures:

<table>
<thead>
<tr>
<th>SP-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review the Android manifest file and build configuration to ensure that the application is built against the latest available SDK (targetSdk and compileSdk).</td>
</tr>
</tbody>
</table>

### Example: VX-U1

| UI and Graphics | VX-U1   | CR-5                                                                 | The app supports both landscape and portrait orientations (if possible) and folding / unfolding. Orientations expose largely the same features and actions and preserve functional parity. Minor changes in content or views are acceptable. |
|-----------------|---------|----------------------------------------------------------------------|
|                 | VX-U2   | CR-5                                                                 | The app uses the whole screen in both orientations and does not letterbox to account for orientation changes, including folding and unfolding. Minor letterboxing to compensate for small variations in screen geometry is acceptable. |
|                 | VX-U3   | CR-5                                                                 | The app correctly handles rapid transitions between display orientations and device folding / unfolding without rendering problems or losing state. |

https://developer.android.com/docs/quality-guidelines/core-app-quality
### Stability

<table>
<thead>
<tr>
<th>Area</th>
<th>ID</th>
<th>Tests</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability</td>
<td>PS-S1</td>
<td>CR-all</td>
<td>The app does not crash or block the UI thread causing ANR (Android Not Responding) errors. Utilize Google Play's pre-launch report to identify potential stability issues. After deployment, pay attention to the Android Vitals page in the Google Play developer console.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SD-1</td>
<td></td>
</tr>
</tbody>
</table>

### Performance

<table>
<thead>
<tr>
<th>Area</th>
<th>ID</th>
<th>Tests</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>PS-P1</td>
<td>CR-all</td>
<td>The app loads quickly or provides onscreen feedback to the user (a progress indicator or similar cue) if the app takes longer than two seconds to load.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SD-1</td>
<td></td>
</tr>
</tbody>
</table>

---

Security - OWASP

● OWASP - Open Web Application Security Project
  ○ Community-based security resources
  ○ Articles, methodologies, guidelines, documentation

● OWASP Mobile Application Security (MAS)
  ○ https://mas.owasp.org/
  ○ OWASP MASVS - Security Verification Standard
  ○ OWASP MASTG - Security Testing Guide (iOS + Android checklists)
OWASP MASVS

- [https://mas.owasp.org/MASVS/](https://mas.owasp.org/MASVS/)
- For software architects, developers, testers
- Distinguishes 2 security levels
  - **Level 1 (L1)** - general, for all apps “Standard Security”
  - **Level 2 (L2)** - for apps handling sensitive data “Defense-in-depth”
- Distinguishes
  - Reverse engineering resiliency req.-s (R)
MASVS levels

- **L1**
  - **Basic** reqs - code quality, data handling, interaction in mobile env.
  - For all mobile apps

- **L2**
  - **Advanced** security, part of app architecture and design
  - Threat model
  - For apps with highly sensitive data (e.g. banking, health)

- **R**
  - **State-of-the-art** security
  - Resilience against defined attacks:
    - Tampering, modding, reverse-engineering
  - Leverages Hardware security features or verifiable software protection techniques
  - For apps dealing with preserving intellectual property, apps which need tamper-proofing

https://mas.owasp.org/MASVS/Intro/0x03-Using_the_MASVS/#verification-levels-in-detail
OWASP MAS Checklist

- [https://mas.owasp.org/MAS_checklist/](https://mas.owasp.org/MAS_checklist/)
- Requirements and test cases
- Distinguish L1/L2/R
- For both Android, iOS

<table>
<thead>
<tr>
<th>ID</th>
<th>MAGVS-ID</th>
<th>Detailed Verification Requirement</th>
<th>L1</th>
<th>L2</th>
<th>R</th>
<th>Common</th>
<th>Android</th>
<th>iOS</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>MSTG-STORAGE-1</td>
<td>System credential storage facilities need to be used to store sensitive data, such as PI, user credentials or cryptographic keys.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>2.2</td>
<td>MSTG-STORAGE-2</td>
<td>No sensitive data should be stored outside of the app container or system credential storage facilities.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>2.3</td>
<td>MSTG-STORAGE-3</td>
<td>No sensitive data is written to application logs.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fail</td>
</tr>
<tr>
<td>2.4</td>
<td>MSTG-STORAGE-4</td>
<td>No sensitive data is shared with third parties unless it is a necessary part of the architecture.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N/R</td>
</tr>
<tr>
<td>2.5</td>
<td>MSTG-STORAGE-5</td>
<td>The keyboard cache is disabled on text inputs that process sensitive data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>2.6</td>
<td>MSTG-STORAGE-6</td>
<td>No sensitive data is exposed via IPC mechanisms.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fail</td>
</tr>
<tr>
<td>2.7</td>
<td>MSTG-STORAGE-7</td>
<td>No sensitive data, such as passwords or pins, is exposed through the user interface.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fail</td>
</tr>
<tr>
<td>2.8</td>
<td>MSTG-STORAGE-8</td>
<td>No sensitive data is included in backups generated by the mobile operating system.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>MSTG-CODE-4</td>
<td>Debugging code and developer assistance code (e.g. test code, backdoors, hidden settings) have been removed. The app does not log verbose errors or debugging messages.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>MSTG-CODE-5</td>
<td>All third party components used by the mobile app, such as libraries and frameworks, are identified, and checked for known vulnerabilities.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>MSTG-CODE-6</td>
<td>The app catches and handles possible exceptions.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Profiling

- Measure performance
  - CPU, Memory, network, energy
  - Java heap
- DEMO with Weather?

https://developer.android.com/studio/profile
Building for release
Builds

- So far our builds used the "debug" configuration
  - app-debug.apk
  - Extra logging, debugging enabled
- For release, should create "release" build
  - Optimized
- In Android studio, switch to "release" build using "Build Variants"
  - DEMO

Release-ready build

- Enable **minify & Proguard (R8)**
  - Check Project-level build.gradle for relevant configuration

```java
android {
  buildTypes {
    release {
      minifyEnabled true
      shrinkResources true
      proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
  }
}
```

https://developer.android.com/studio/publish/preparing#publishing-configure
Signing process

- Your built app should also be **signed**
  - Verifiable mark from the developer - identifies author
  - Ensures app has not been modified since initial distribution

- When app is installed, Android Package Manager checks that it’s signed with certificate

- Android uses public/private certificates to sign Android apps (.apk files).
  - Certificate:
    - Info about developer + key

Signing Process

Goal - associate app with devs public key (asymmetric crypto signing)

- Dev: Calculate hash of APK file, encrypt it with private key
- 3rd party: use devs public key to decrypt the encrypted hash and verify that it matches the actual hash
- Android - “KeyStore” - container of certs & keys
  - Make sure you store it safely, secretly

https://developer.android.com/studio/publish/app-signing
Managing keys

● Lose the key -> can’t update app anymore
● Leak the key -> security risk
  ○ Someone creates new malware version of your app
  ○
● Previously, you could directly publish signed apk files.
  ○ Since August 2021, the signing is \textit{partially delegated} to Play Store with \textbf{Bundles}
  ○ Avoids above risks

Modern apps are built as **AAB** before release

- **AAB** - archive of resources and compiled bytecode
- **Google Play** generate and signs APK based on AAB
  - Goal: optimized APKs - only include necessary resources
- **2 keys are involved: Upload + Signing**
  - Signing key not managed by you
Signing Demo

Generate Signed Bundle or APK

Module: My_Finished_App.app
Key store path: /home/jakob/Android/keystore/mystore.jks
Key store password: ********
Key alias: key0
Key password:

- Remember passwords
- Export encrypted key for enrolling published apps in Google Play App Signing

Choose Key

- Use an existing key: key0
- Create a new key
- Alias:
- Password: [masked]
- Confirm: [masked]
- Validity (years): 25

Certificate
- First and Last Name:
- Organizational Unit:
- Organization:
- City or Locality:
- State or Province:
- Country Code (XX):

[Next] [Cancel] [Help]
Test your release build!

- Don’t forget to test the release build
- Emulator - try different form factors (foldable, tablet, wearable)
- Hardware
- Device test labs
  - Firebase, TalTech testlab
Publishing
App Store

- Need a Developer Account
  - Google Play ($25)
  - Apple - $99 / annum

- Create (test) release
  - AAB
  - Package name uniquely identifies

- Describe app content

- Provide other materials
  - EULA, screenshots, descriptions, banner

https://play.google.com/about/developer-content-policy/
Crash reports

- Play Console provides “Android Vitals” service
  - ANR rate: % of daily active users who experienced any type of ANR.
  - User-perceived ANR rate: % of active users who experienced 1+ user-perceived ANR.
  - Multiple ANR rate: % of daily active users who experienced at 2+ ANRs.

- **User-perceived ANR** rate is a core vital meaning that it affects the discoverability of your app on Google Play.
  - important because the ANRs it counts always occur when the user is engaged with the app

https://developer.android.com/topic/performance/vitals/anr#android-vitals
## Real-time ANRs

### Last 7 days

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Reports</th>
<th>Impacted users</th>
<th>Last reported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input dispatching timed out (Waiting because the touched window's input channel... in com.google.firebaseMessaging.UnityPlayerActivity</td>
<td>8</td>
<td>8</td>
<td>Today, 00:11</td>
</tr>
<tr>
<td>Input dispatching timed out (Waiting to send non-key event because the touched... in com.google.firebaseMessaging.UnityPlayerActivity</td>
<td>7</td>
<td>5</td>
<td>Yesterday, 00:16</td>
</tr>
<tr>
<td>Broadcast of Intent { act=com.google.android.c2dm.intent.RECEIVE pkg=0x10000... in com.google.firebase.idFirebaseinstancesReceiver</td>
<td>5</td>
<td>5</td>
<td>Yesterday, 19:12</td>
</tr>
<tr>
<td>Input dispatching timed out (Waiting because no window has focus but there is a... in com.google.firebaseMessaging.UnityPlayerActivity</td>
<td>5</td>
<td>5</td>
<td>Yesterday, 18:57</td>
</tr>
</tbody>
</table>

### By app version

- **400092**: 8 (100.0%)

### By Android version

- **Android 9**: 8 (100.0%)

### By device

- **Honor 8A (HWJATM)**: 5 (62.5%)
- **Galaxy A10 (a10)**: 1 (12.5%)
- **Huawei Y5 2019 (HWAW8N-M)**: 1 (12.5%)
- **Moto G6 (play) (ajriter)**: 1 (12.5%)

---

*Note: The above information is extracted from a screenshot of a dashboard showing real-time ANRs (App Necessity Reports) for a mobile application. The dashboard is displaying data for the last 7 days, categorized by cluster, app version, Android version, and device.*
How it works
1. You digitally sign each release using your upload key before publishing it to a track in the Play Console.
2. Google Play uses the upload certificate to verify your identity and then re-signs your release using the app signing key for distribution.
3. Each Android device checks that the release’s app signing certificate matches the certificate of the installed app before updating it.

Learn more about how App Signing by Google Play works.

App signing certificate
This is the public certificate for the app signing key that Google Play uses to sign your app before distributing it to Android devices. The app signing key itself is inaccessible and kept on a secure Google server. Use the certificate below to register your app signing key with your API providers.

- MD5 certificate fingerprint:

- SHA-1 certificate fingerprint:

- SHA-256 certificate fingerprint:

Upgrade your app signing key for new installs
You can upgrade your app signing key for new installs; for example, if you want to move to a cryptographically stronger key. Google Play will use the new key to sign all new installs and their updates. Your legacy key will continue to be used for updates for users who installed the app signed with that key. You can request a new key once per app for the lifetime of the app. Review these important considerations before requesting an upgrade.

REQUEST KEY UPGRADE

Upload certificate
Summary

- Test with checklists and guidelines
  - OWASP
  - Using different devices
  - Test release build!

- Prepare signed “release” build
  - Minified, Compressed
  - AAB format

- Play Store
  - Dev. license, policy alignment, promo materials
  - AAB signed and final apks generated

For iOS, process is similar (but more stringent for Store)
Final weeks of the course

- **Next week - Guest lecture about industry experience** *(Kelian Kaio)*
- **Projects (more on next slide)**
  - Lab sessions - project consultation
  - Projects Final presentations on **12.12**
    - OR 19th? We can vote - until 30.11 - Find doodle on Slack
  - Report deadline - **19.12**
- **Quiz #2 - 15.12**
  - Android testing
  - General concepts of App Architecture
  - Web, Cloud Services for mobile apps
  - iOS, Flutter, React Native
  - Release of mobile apps
Project - presentations

Each team has max 13 minutes (including Q&A)

● Pitch video (1-3 minutes)
  ○ a kind of advert for your product/solution
  ○ should demonstrate the features from the user(s) and story perspective
  ○ technical details are not important - a working prototype must still be be part of it
    ■ can mock/fake some aspects

● Project experience (~7 minutes)
  ○ Technical comments, role assignment
  ○ What went well, what went poorly
  ○ Future improvement ideas

● Discussion & Questions (3 minutes)
Project reports

● Technical Readme
  ○ How to compile / run this app? What HW?
  ○ Technical dependencies - need some API key / DB access? Link to create account etc

● The concept story (idea) of your app - the features, target audience.

● Brief report about project experience
  ○ Which functional feature blocks?
  ○ What other technologies, libraries you used. Comment on choice.
  ○ Who worked on what? What went well/south ?
  ○ What would you add/do if you had more time? Most challenging problem?

● OWASP report
  ○ Choose 2 OWASP-MASTG L1 requirements
  ○ (More on next slide)
OWASP MASTG task

- Choose two L1 requirements which have a link to a Test Case
  - Write down the MASVS-ID
    - Provide links to test case
  - Explain how it applies to your project
  - Explain how you tested the requirement
    - Activities, relevant files, ...
  - Discuss - did you fulfil the requirement?
    - If not - what needs to be done to fulfil it? (no need to implement)
That’s all!

See you during project consultations
MAD Skills

https://www.youtube.com/watch?v=PB-hZVTScUg&list=PLWz5rJ2EKKc91i2QT8qfrfKgLNIJiG1z7