Gen™

Android lecture 6

Release process

2023

Lukáš Prokop
Simona Kurňavová

norton Avast LifeLock by norton Avira AVG REPUTATION DEFENDER by norton CCleaner

Application release

Gather materials for release

- **Cryptographic keys**
  - Apps in store are digitally signed, by developer certificate
  - Identify app developer
  - Self-signed certificate is enough
  - Needs to end after 22 October 2033 (>25 years recommended)
  - Protect your private key and password, it is not possible to update application if it is lost
- **End-user license agreement (EULA)**
  - User should know if you gather some data
  - Not required but recommended
  - GDPR

Gather material for release

- **Application icon**
  - Visible in launcher, settings or other applications
  - High-res assets for google play store listing
- **Misc. materials**
  - Promo and marketing materials
  - Promotional text and graphic for store listing
- **Changelog**

Before release

- **Delete unused parts (sources, resources, assets)**
- **Package name**
  - https://play.google.com/store/apps/details?id=com.avast.android.mobilesecurity
- **Turn off debug features**
  - Delete Log.* calls
  - Remove android:debuggable flag from AndroidManifest
  - Remove Debug or Trace calls
  - If you using WebView ensure that debug is disabled, otherwise is possible to inject js code
- **Clean up your directories, checks if libraries doesn’t include some unnecessary files to your *.apk (*.proto, java manifests, …)**

Configure application for release

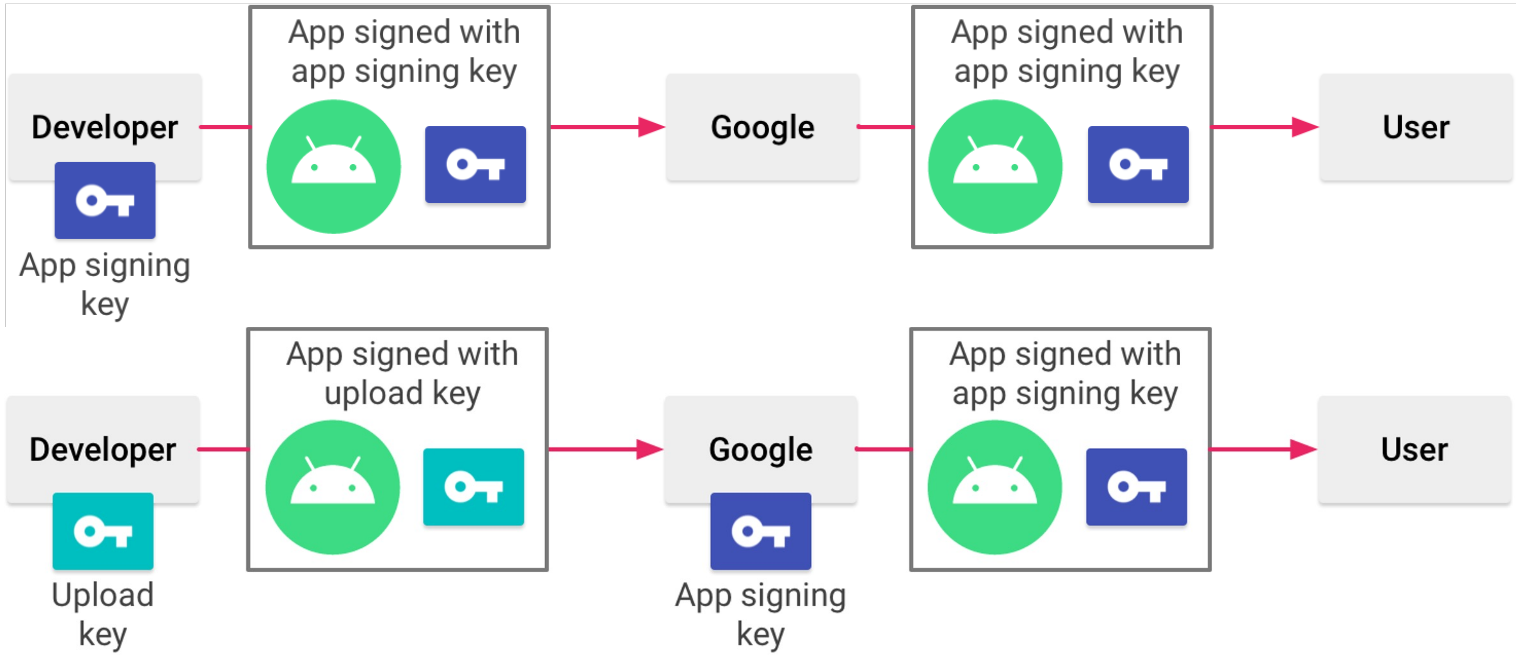
- **Review**
  - permissions - remove unnecessary
  - App icon and label
  - Version code and version name
  - Used URLs test vs. production backend
- **Check compatibility**
  - Support of multiple screens
  - Tablet mode

Build application for release

- **Signing**
  - Manually
    - Using Keytool and Jarsigner from JDK
  - Configure sign options in gradle
  - Android studio
  - Sign scheme v2 https://source.android.com/security/apksigning/v2
  - Sign scheme v3 https://source.android.com/security/apksigning/v3
  - Sign scheme v4 https://source.android.com/security/apksigning/v4
- **Obfuscation**
  - Use proguard to obfuscate your code, it is really easy to decompile application and find how it works, endpoints
- **Consider**
  - Who has access to your sign key
  - Signing server

Signing

- **Sign vs upload key**


Developer
App signing key
App signed with app signing key
Google
App signed with app signing key
User
Developer
Upload key
App signed with upload key
Google
App signing key
App signed with app signing key
User

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

Prepare external servers and resources

- Ensure that backend is running
- Check if app is switched to prod environment

Testing your application for release

- Regression test
- Test new features
- If it is possible test it on multiple devices, android versions
- Test multiple languages, including RTL
- Check lint for several issues

Publishing

- **Google play store**
  - Registration cost 25 USD
  - Reporting about installs
  - Crashes
    - If user sends it
  - Cloud test lab
    - Run monkey tests on multiple devices before releasing
  - Alpha/Beta groups
  - Distribution specific domain - internal apps
  - Staged rollout
  - API - import crash reports to bug tracker, upload new APK, ...

Publishing

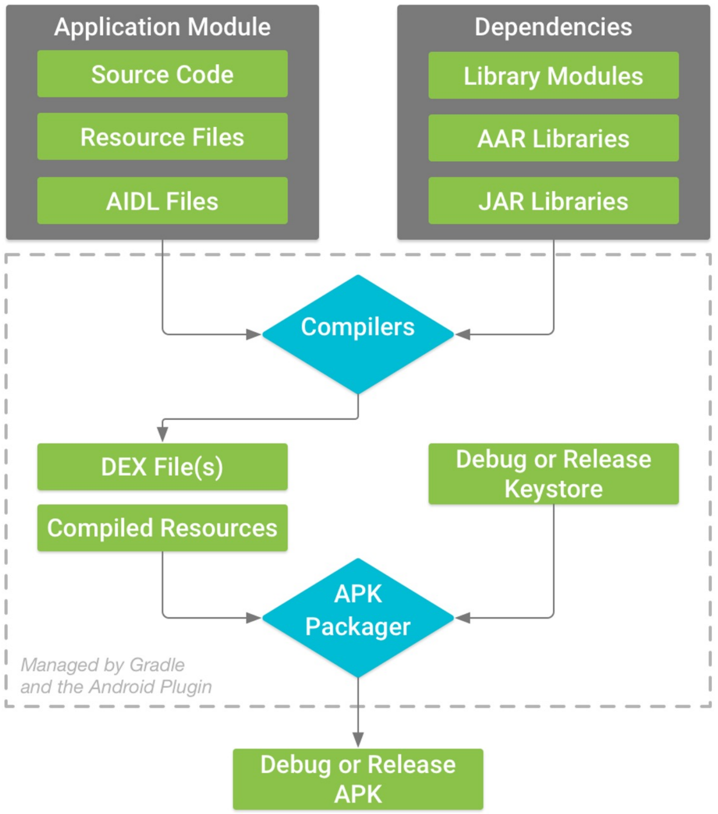
- **Email, Web page**
  - Untrusted sources - security risk
  - Manual Updates
- **3rd party distribution**
  - Amazon
  - F-droid

After publish

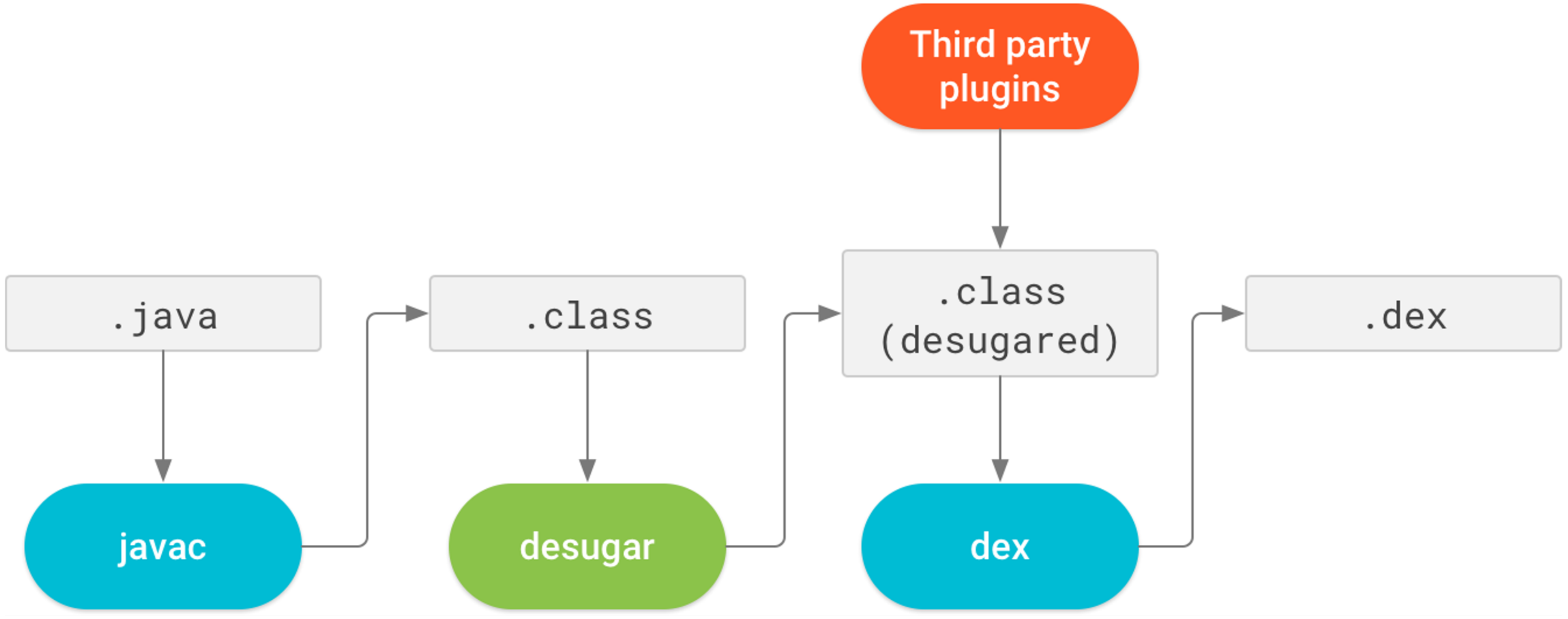
- Monitor new crashes
- New permission slows down spreading between users on API < 23 (Marshmallow 6.0)
- Not good idea publish app before weekend or vacation

Build process and APK structure

Build process



Support java 8 features (d8)



Building android apps limitations

- **Dex limit 64k methods**
  - Shrink unused methods and classes (libraries)
    - You need to specify what is entry point, build dependency graph. Classes which are not part of graph are removed, unused methods as well
    - When you work on library, provide proguard rules together with library
  - Sometimes is better copy some classes from library into application
    - for example guava library
- **Google doesn’t allow code side load**

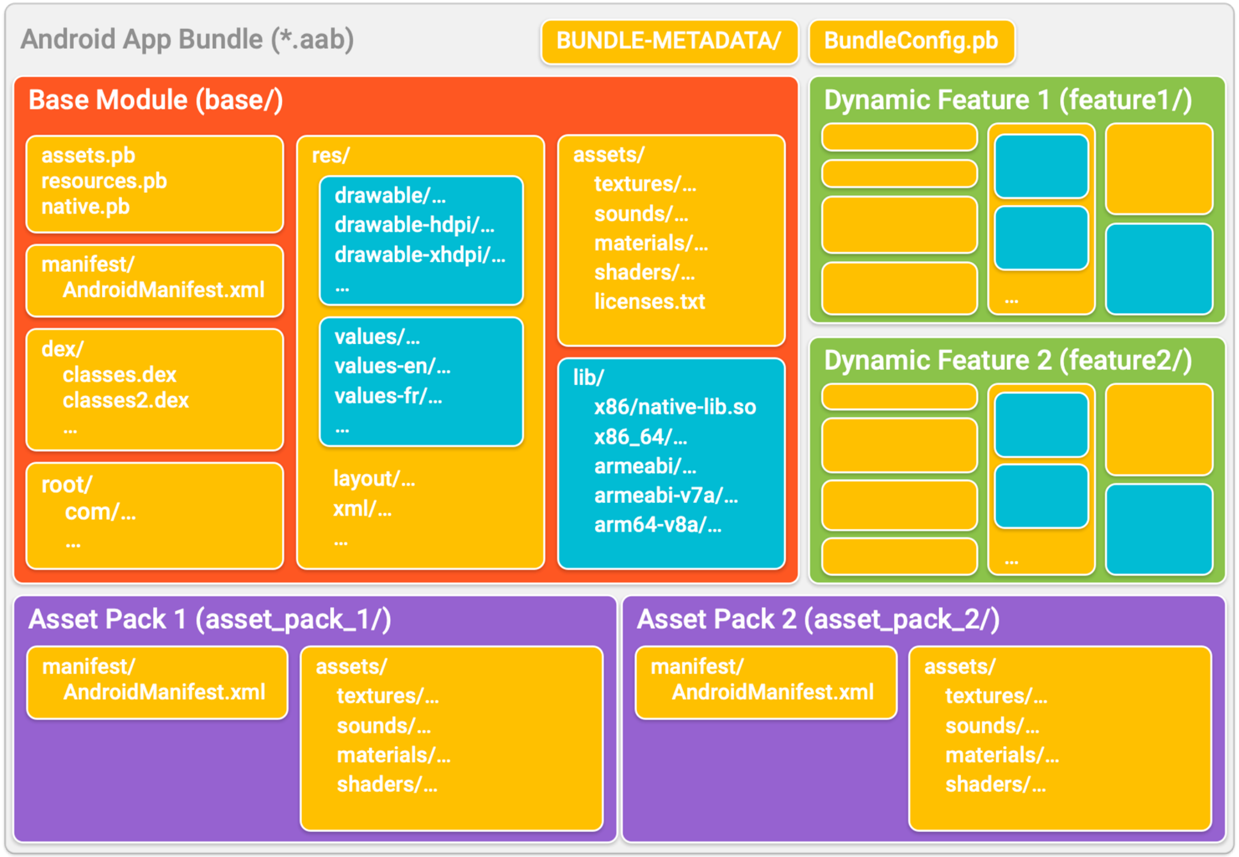
Multidex

- Native support since API-21 for older version support library
- Try to avoid using multidex, it slows down application start
- Splits classes to multiple dex files
- on API>=21 dex files are converted to single .oat file (ART runtime)
- Main dex file loaded when app is started
- Loading of additional dex files is performed during initialization
- Dex files are in app folder
- https://www.blackhat.com/docs/ldn-15/materials/london-15-Welton-Abusing-Android-Apps-And-Gaining-Remote-Code-Execution.pdf

Apk structure

```
.
├── AndroidManifest.xml- binary XML form of Android manifest
├── classes.dex     - classes compiled to dex
├── META-INF
│   ├── CERT.RSA       - Application certificate
│   ├── CERT.SF        - SHA-1 digest of corresponding lines in the MANIFEST.mf
│   └── MANIFEST.MF    - Manifest file list of files with their SHA-1 hash
├── assets             - assets files
├── res                  - resources
│   ├── drawable-hdpi-v4
│   ├── drawable-mdpi-v4
│   ├── drawable-xhdpi-v4
│   ├── drawable-xxhdpi-v4
│   ├── drawable-xxxhdpi-v4
│   ├── layout
│   ├── menu
│   └── xml
└── resources.arsc       - resources compiled to binary form
```

AppBundle



Gen

https://developer.android.com/guide/app-bundle

Proguard/R8

- Shrink - smaller code, faster build
- Optimize - faster code, removing static conditions
- Obfuscate - make it harder to read

Decompile apk

- **Unzip the apk**
- **Dex2Jar to convert classes.dex to jar archive**
  - https://github.com/pxb1988/dex2jar
- **jd-gui to view decompiled classes**
  - http://jd.benow.ca/
- **BytecodeViewer**
  - https://bytecodeviewer.com/
- **Android studio apk analyzer**
  - Easy to check resources
  - Compare multiple apk

Android and SW development - best practices

- Keep strings, dimensions, colors, … in resources
- Create libs for parts used in multiple projects (simplify maintenance, speed-up builds)
- Use git
- Do code reviews
- Write tests

Build flavors/Build variants

- Whitelabel apps
- Debug variant of the app

Thank you

Lukáš Prokop
Simona Kurňavová

Lukas.Prokop@gendigital.com
Simona.Kurnavova@gendigital.com

Gen™