Security in Android

Seminararbeit

Hochschule für angewandte Wissenschaften
Würzburg-Schweinfurt

Kristoffer Schneider

25. Mai 2015

Inhaltsverzeichnis

1 Vorwort **4**

2 Das Android Betriebssystem **5**

3 Grundlegender Aufbau einer Android App **6**

4 Sicherheitsaspekte der Android-Architektur **7**

4.1 Basis Rechtesystem 7

4.2 Sandboxing und Permissions 7

4.2.1 Permissions im Detail 7

Abbildungsverzeichnis

1 Die Architektur von Android ASI-P-2 5

1 Vorwort

Das Smartphone hat in den letzten Jahren sich in den Alltag der meisten Menschen eingefügt (Statistik?!?!). Es dient als Alltagshelfer, Notizbuch, Terminkalender, Kommunikationsmittel und Zeitvertreib, und ist dabei fast überall dabei.
Viele vergessen dabei, dass Smartphones mittlerweile die Leistung eines kleinen Computers haben und somit auch die selben Gefahren wie am PC Zuhause vorhanden sind. Kaum einer hat auf seinem PC kein Anti-Viren System installiert. Aber wer hat eines auf seinem Smartphone oder Tablet? Dabei hat man gerade auf diesen Geräten zum Teil hoch sensible Daten gespeichert.
Daher wollen wir im Folgenden auf sicherheitstechnische Aspekte des Android und iOS Betriebssystems eingehen und aufzeigen welche Hilfsmittel beide für Entwickler und Nutzer bereitstellen.

2 Das Android Betriebssystem

Das Unternehmen Android wurde 2003 von Andy Rubin gegründet und wurde 2005 von Google aufgekauft. Seitdem kümmert sich Google und das Android Open Source Project (AOSP) um die Weiterentwicklung des Systems. Aktuell ist Android, mit 55.6% Marktanteil[1], das vorherrschende Betriebssystem für mobile Endgeräte in den USA.

Basis für das Betriebssystem ist ein modifizierter Linux-Kernel und eine Java Virtual Machine (JVM). Bis einschliesslich Version 4.4 wurde hierfür die Dalivk Runtime und für alle neueren Versionen die Android Runtime (ART) verwendet. Jede App läuft in einer eigenen Instanz der entsprechenden Runtime und damit in einer Sandbox.
Oberhalb der JVM sind die meisten Komponenten in Java implementiert.

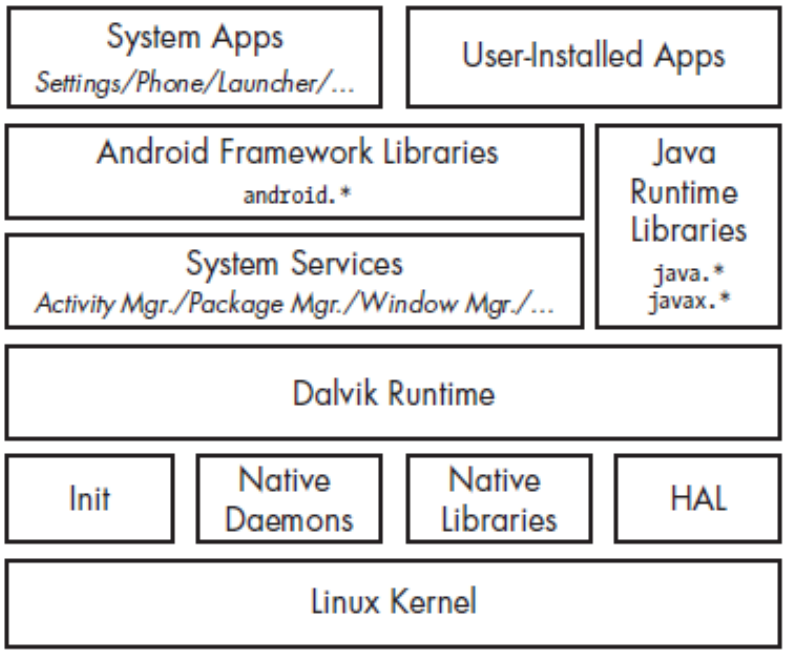

Abbildung 1: Die Architektur von Android ASI-P-2

[1] Kantar Worldpanel: Smartphone OS sales market share, http://www.kantarworldpanel.com/global/smartphone-os-market-share/, 17.5.2015

3 Grundlegender Aufbau einer Android App

> Android apps are written in the Java programming language. The Android SDK tools compile your code - along with any data and resource files - into an APK: an *Android package*, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.
> (Android: Application Fundamentals,
> http://developer.android.com/guide/components/fundamentals.html,
> 20.5.2015)

Eine App besteht im Kern aus zwei Teilen. Den eigentlichen Programmkomponenten und einer Manifest Datei (AndroidManifest.xml).

Als Programmkomponenten können unter anderem vorkommen:

- Activities - stellen die Benutzeroberfläche dar
- Services - kann im Hintergrund laufen, auch wenn die App minimiert ist
- Content Provider - stellt Daten für die eigene und evtl für andere Apps zur Verfügung
- Broadcast Receiver - um Systemweite Benachrichtigungen zu empfangen (z.B dass ein Download beendet wurde)

In der Manifest-Datei werden Eigenschaften der App definiert. Darunter zählen beispielsweisse:

- Name der App
- Ziel SDK-Versionen
- Versionsnummer
- optional eine UserId (siehe 4.1)
- Permissions (siehe 4.2)

Des weiteren muss jede App signiert werden. Das hierfür benötigte Zertifikat kann sich jeder Entwickler selbst generieren und muss nicht durch einen Certification Authority (CA) beglaubigt werden. Dabei wird angeraten, dass ein Entwickler für all seine Apps dasselbe Zertifikat nutzt. Mithilfe der dadurch gegebenen Signatur wird eine *Same-Origin-Policy* erschaffen, die bei jedem Update sicherstellt, dass dieses wirklich vom Entwickler der Applikation stammt und nicht durch einen dritten eingebracht wurde

4 Sicherheitsaspekte der Android-Architektur

Bereits durch die Architektur des Betriebssystems, insbesondere durch die restriktive Rechtevergabe und das Sandboxing, wird versucht ein möglichst sicheres System bereitzustellen.

4.1 Basis Rechtesystem

Von Linux wurde auch das Basis-Rechtesystem übernommen. Hierbei bekommt jede App eine eindeutige User-ID (UID) zugewiesen, welche im Normalfall zur Installationszeit zugeteilt wird. Jeder Nutzer, und somit auch jede App, arbeitet grundsätzlich erst einmal nur innerhalb der ihm zugewiesenen virtuellen Maschine und dem damit verbundenen Dateisystem.

Da es dennoch in vielen Fällen nötig ist Daten zwischen verschiedenen Apps auszutauschen, gibt es mehrer Möglichkeiten dies zu tun. Die üblichen Wege wären Intends oder SharedPreferences. Zusätzlich gibt es noch die Möglichkeit mehreren Apps dieselbe UID zuweisen zu lassen. Dies ist allerdings nur möglich wenn die entsprechenden Applikationen mit dem selben Zertifikat signiert wurden und in deren Manifest Datei eine gemeinsame UID festgelegt wurde. Durch dieses Rechtesystem wird versucht sicherzustellen, dass kein Nutzerprogramm als *root* ausgeführt wird.

4.2 Sandboxing und Permissions

Wie bereits erwähnt, laufen die Applikationen jeweils in ihrer eigenen Sandbox. Grundsätzlich ist die App damit in ihrer Ausführung auf ihren Bereich beschränkt und kann nicht mit anderen Prozessen und Daten ausserhalb interagieren. Dennoch ist es in den meisten Fällen sinnvoll mit Systemservices und Nutzerdaten zu interagieren die nicht in der eigenen Sandbox verfügbar sind.

4.2.1 Permissions im Detail

Um nun die bestehenden Zugriffsrechte erweitern zu können, müssen die entsprechenden Rechte (Permissions) in der Manifest Datei deklariert und angefordert werden. Zu Installationszeit werden diese Permissions dem Nutzer angezeigt und dieser wird gefragt ob er den Rechtswünschen der App zustimmt oder nicht. Dabei gilt das *Alles-Oder-Nichts-Prinzip*, d.h. entweder bekommt die Anwendung alle Rechte oder keine - was eine nicht Installation zur Folge hat. Des weiteren können die Berechtigungen nach der Installation nicht mehr angepasst werden.
Oben genannte Berechtigungen sind beispielsweiße für Zugriffe auf externe Speichermedien oder auch die Kamera nötig. Dabei ist allerdings zu beachten, dass die Permissions zum Teil sehr grob definiert sind. Wodurch für den Nutzer nicht unbedingt erkenntlich ist, welche Informationen eine App warum abgreift und ob die App wirklich gebraucht des Rechts macht.

Um die Problematik dessen zu verdeutlichen, gehe ich nun exemplarisch auf besonders problematische Permissions eingehen.

RECORD_AUDIO Permission:

> Allows an application to record audio [2]

Dabei ist für den Nutzer nicht sichtbar wann eine Aufnahme läuft, ausser die Applikation stellt dafür einen Hinweis bereit - wobei hier die Frage ist ob dieser auch wirklich verlässlich ist. Stellt die App einen Service bereit, kann eine solcher Mitschnitt auch im Hintergrund geschehen. Die einzige Chance AUDIO_RECORD zur Laufzeit zu unterbinden ist es den Service bzw. die App über den Anwendungsmanager zu beenden.

[2] https://developer.android.com/reference/android/Manifest.permission.html#RECORD_AUDIO