Dr. Mario Kubek

Kurs 01864

Mobile Security

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Kurseinheit 1

Einführung in Mobile Security

Diese Kurseinheit führt in die allgemeinen Sicherheitskonzepte und -mechanismen mobiler Endgeräte wie Smartphones und Tablets sowie der auf ihnen laufenden Betriebssysteme und Applikationen ein. Zuvor wird zu ihrer Motivation auf die Bedrohungen, denen solche Geräte und ihre Nutzer ausgesetzt sind, eingegangen.

Lernziele

Die Lernziele dieser Kurseinheit sind:

- die Ursachen mannigfaltiger Bedrohungen für mobile Endgeräte und die bei ihrer Nutzung beteiligten Parteien kennenzulernen,
- die allgemeinen Sicherheitsmechanismen mobiler Plattformen nennen und erklären zu können,
- die Kernaspekte der Sicherheitsarchitekturen der populären Betriebssysteme Android und iOS verstanden zu haben,
- konkrete Bedrohungsszenarien durch schadhafte Applikationen benennen und unterscheiden zu können,
- eine Laborumgebung für die Analyse mobiler Applikationen einrichten zu können.

1.1 Einleitung

In den letzten Jahren haben sich Smartphones sehr stark verbreitet. Es handelt sich hierbei um eine Kombination aus Mobiltelefon und Computer. Man kann damit mobil telefonieren, aber auch Programme (engl. **application**) installieren und laufen lassen. Diese Programme werden kurz **App** genannt. Mit Hilfe dieser Programme lässt sich komfortabler kommunizieren. Eine kleine Kontaktdatenbank enthält die Namen, Telefonnummern und weitere Kontaktdaten von Familie, Freunden, Kollegen, usw. Statt umständlich lange Telefonnummern eintippen zu müssen, kann man jetzt bequem einen Eintrag der Kontaktliste auswählen und diesen direkt anrufen.

Über das Mobilfunknetz oder ein WLAN kann ein Smartphone ständig mit dem Internet verbunden sein. Es kann Webanwendungen benutzen und Daten aus dem Internet laden, aber natürlich auch Daten ins Internet senden. Weiterhin sind moderne Smartphones auch mit einem GPS-Empfänger ausgestattet. Mit Hilfe dieses Empfängers ist der jeweilige Standort des Smartphones bekannt.

Weiterhin können Smartphones auch beliebige andere Programme ausführen. Hierzu gehören Spiele oder auch Anwendungen, die vom aktuellen Standort des Smartphones abhängen[1], ebenso wie Büroanwendungen. Asokan et al. [1] charakterisieren Smartphones anhand dieser drei Eigenschaften:

1. Sie können ihre Funktionalität erweitern, indem neue Software-Komponenten installiert werden.

2. Sie können auf das Internet zugreifen und man kann aus dem Internet auf das Smartphone zugreifen.

3. Auf einem Smartphone liegen private und vertrauliche Daten.

Smartphones werden nicht nur privat benutzt, sondern auf ihnen werden immer mehr dienstliche Anwendungen installiert. Daraus ergeben sich eine Reihe von Sicherheitsproblemen. Sie werden in Abschnitt 1.2 besprochen. Danach werden in Abschnitt 1.3 einige allgemeine Sicherheitstechniken vorgestellt, um den Bedrohungen begegnen zu können. Den Abschluss dieses Kursteils bildet Abschnitt 1.4. Dort werden zu den aktuell verbreitetsten Betriebssystemen von Smartphones die dort implementierten Sicherheitsmechanismen vorgestellt.

1.2 Allgemeine Bedrohungen

Die erste und nächstliegende Bedrohung bei so kleinen und handlichen Geräten besteht im Verlust des Smartphones. Normalerweise führt man sein Smartphone immer mit und somit gibt es auch viele Gelegenheiten, bei denen das Smartphone vergessen und liegen gelassen werden kann. Aber auch bei Dieben sind Smartphones begehrt und so werden sie häufig gestohlen.

Ein unredlicher Finder oder ein Dieb soll das Gerät aber nicht benutzen und damit dann auf Kosten des legitimen Eigentümers kommunizieren können. Um das zu

[1] Beispielsweise Restaurants in der Nähe finden.

erschweren wird der PIN-Mechanismus benutzt, der immer schon bei der Mobilfunktechnik eingesetzt wurde. Der Dieb soll aber natürlich auch keine Gelegenheit bekommen, die (vertraulichen) Daten auf dem Gerät auslesen zu können.

Häufig ist der Speicherplatz auf Smartphones knapp, so dass alle Systeme auch die Speicherung von Benutzerdaten im Internet ermöglichen. Diese Dienste nennt man auch **Cloud Services**. Da das Smartphone ständig im Netz ist, kann es die Benutzerdaten auch jederzeit aus der Cloud des Benutzers laden und dann verarbeiten. Damit kann man beispielsweise seinen Kalender einfach zwischen PC und Smartphone synchronisieren. Aber auch Dokumente und Fotos werden gerne in der Cloud gespeichert.

Natürlich möchte man nicht, dass andere diese Daten mitlesen können. Die Daten müssen also bei der Übertragung verschlüsselt werden und die Cloud muss einen guten und sicheren Mechanismus zur Benutzerauthentisierung einsetzen. Hier kommen heute überwiegend passwortbasierte Verfahren zum Einsatz. Ist so ein Passwort auf dem Smartphone gespeichert – vielen Benutzern ist es zu unbequem, das Passwort immer wieder einzugeben – dann kann ein Dieb möglicherweise mit dem Smartphone alle Daten des Benutzers mitlesen. Sind in der Cloud jetzt noch die Benutzerpasswörter bei den vielen Internetdiensten (Online-Händler, Banking, eBay, Foren, usw.) gespeichert, so kennt der Dieb alle Informationen, um im Internet an die Stelle desjenigen zu treten, der das Smartphone verloren hat. Damit könnte der Dieb das Konto des Opfers plündern, seinen Ruf ruinieren und andere schlimme Dinge tun.

Spionierende Anwendungen: Durch das GPS-Modul im Smartphone können die dort installierten Programme den Aufenthaltsort des Besitzers verfolgen. Voraussetzung ist, dass dem jeweiligen Programm der Zugriff auf das GPS-Modul erlaubt ist. Für alle **location based services** ist das auch erforderlich. Was ein Programm dann mit diesen Daten macht, bleibt häufig unklar. Gerade kostenlos vertriebene Programme sind oft so programmiert, dass sie diese Daten irgendwie kommerziell verwerten. Das Recht hierzu bekommen sie vom Benutzer bei der Installation des Programms erteilt. In den Lizenzbedingungen des Programms bekommt der Benutzer das Recht, das Programm installieren und benutzen zu dürfen und im Gegenzug gewähren die Benutzer dem Programmautor oft das Recht, die gewonnenen Daten zu verwerten.

Neben diesen Aufenthaltsdaten gibt es viele weitere Daten, die kommerziell interessant sind. Wie oben schon erwähnt, gibt es die persönliche Kontaktliste des Benutzers. Sie enthält die Namen und Kontaktdaten von Familie, Freunden, Kollegen, usw. Mit diesen Daten kann man Beziehungsnetze erstellen, aus denen hervorgeht, wer wen kennt. Diese Daten lassen sich gut für Marketingzwecke einsetzen. Wenn viele Ihrer Kontakte als Tennisspieler bekannt sind, dann sind Sie vermutlich auch empfänglicher für Werbung für Tenniszubehör als der Durchschnittsmensch.

Spezielle Programme wie z. B. *WhatsApp*, die einfache Kommunikationsmöglichkeiten ähnlich zur SMS anbieten, können bessere und bequemere Dienste anbieten, wenn der Benutzer ihnen den Zugriff auf die Kontaktliste gewährt. Diese Kontaktinformationen werden dann auf den Servern des Programmanbieters (also *WhatsApp*) gespeichert. Auch wenn Sie selbst kein *WhatsApp*-Benutzer sind, so können Ihre Kontaktdaten dort auf den Servern gespeichert sein. Dazu muss

nur einer Ihrer Freunde, Familienmitglieder, Kollegen, usw. Ihre Daten in seiner Kontaktliste stehen haben und Benutzer von *WhatsApp* sein. Stehen die Server des Programmanbieters dann nicht in der EU und unterliegen den europäischen Vorschriften zum Datenschutz, so kann man nicht wissen, wer alles Zugriff auf diese Daten hat.

Manipulierte Anwendungen: Neue Programme werden auf einem Smartphone normalerweise installiert, indem sie aus dem Internet geladen und dann installiert werden. Die früher üblichen Installationsmedien wie DVDs lassen sich bei Smartphones nicht mehr einsetzen. Woran kann ein Benutzer aber erkennen, dass ein Programm beim Herunterladen nicht unbemerkt manipuliert wurde?

Für die verbreiteten Systeme **Andoid** und **iOS** gibt es sogenannte **App Stores**. Dort kann im Prinzip jeder Entwickler seine Programme anbieten. Die Verteilung des Programms und die Abrechnung des Kaufpreises übernimmt der Betreiber des App Stores. Entwickler müssen sich dort zwar registrieren, aber man kann als Käufer nicht sicher davon ausgehen, welche Person oder Firma denn ein Programm dort eingestellt hat.

Man muss also davon ausgehen, dass Programme aus den App Stores möglicherweise von Hackern erstellt wurden und neben ihrer angepriesenen Funktion auch weitere unbekannte Funktionen enthalten. Man nennt solche Programme **trojanisches Pferd**. Damit ein trojanisches Pferd die anderen laufenden Programme nicht beeinträchtigen kann, bieten alle modernen Betriebssysteme für mobile Geräte spezielle Abschottungsmechanismen zwischen den Programmen an.

Drahtlose Datenübertragung: Mobile Endgeräte verbinden sich i. d. R. drahtlos mit dem Internet. Dabei wird entweder WLAN oder ein Mobilfunknetz (GSM, UMTS, LTE, ...) benutzt. Durch Funk übertragene Daten lassen sich einfacher abhören oder manipulieren als bei kabelgebundener Übertragung. Ihr eigenes WLAN zu Hause können Sie durch den Einsatz von WPA2 einfach vor Abhören schützen. Aber über die vielen anderen, freien WLANs haben Sie keine Kontrolle. Häufig benutzen diese WLANs keine oder nur schwache Verschlüsselung. Sie müssen also davon ausgehen, dass andere im WLAN Ihren Datenverkehr mitlesen können.

Ein Smartphone ist ständig im Netz, man kann nicht am herausgezogenen Kabel erkennen, dass es gerade offline ist. Auch fehlt eine dedizierte Firewall zwischen dem Smartphone und dem Internet. Zu Hause haben Sie i. d. R. eine Firewall, die die Kommunikation Ihrer Heimrechner mit Rechnern im Internet kontrolliert. Ein Smartphone kann ständig „angefunkt“ werden und muss sich selbst darum kümmern, dass nur die vom Benutzer erlaubte Kommunikation stattfinden kann.

Finanzielle Schäden: In Telefonnetzen werden seit längerem sogenannte **Mehrwert-Dienste** angeboten. Sie sind unter speziellen Rufnummern erreichbar und kosten den Anrufer mehr als ein „normaler“ Anruf. Diese Rufnummern werden von den heute üblichen Flatrates i. d. R. **nicht** abgedeckt. Ein Programm auf dem Smartphone, das im Hintergrund ständig solche Nummern anruft, kann den Besitzer des Smartphones sehr teuer zu stehen kommen.

Beim Internet-Banking bieten einige Banken das **MTAN**-Verfahren an. Möchte der Benutzer eine Überweisung tätigen, so erzeugt der Server der Bank eine neue Transaktionsnummer (TAN) und sendet diese per SMS an die hinterlegte Mobiltelefonnummer des Kontoinhabers. Der Inhaber tippt diese Nummer dann ab und autorisiert damit die Überweisung. Ein Angreifer muss also (1) den PC des Benutzers erfolgreich angreifen, um an an die PIN zu kommen und (2) irgendwie an die SMS mit der TAN kommen.

Hierzu kann man ein Programm schreiben, das, wenn es auf dem Smartphone des Kontoinhabers installiert ist, die SMS der Bank im Hintergrund abfängt und an einen Anschluss des Angreifers umleitet. Sollte der Benutzer seine PIN auch auf dem Smartphone gespeichert haben, dann fällt Schritt (1) für den Angreifer weg und die Anwendung auf dem Smartphone kann auch die PIN ausspionieren und dem Angreifer verraten.

Benutzer eines Smartphones haben i. d. R. ihre Bezahlinformationen oder ein Guthaben *(pre paid)* beim App Store hinterlegt und können so einfach und bequem neue Programme kaufen, direkt herunterladen und installieren. Diese Funktion kann auch von Programmen ausgenutzt werden. Beispielsweise kann ein Benutzer nach dem Kauf eines Programms aus diesem Programm heraus weitere Komponenten „nachkaufen“. Spieleprogrammierer können so neue Levels erstellen und Spieler können diese dann aus dem Spiel heraus nachkaufen. Man nennt das auch **In-App-Käufe**.

Eine bösartige App könnte ihre Benutzer nun durch In-App-Käufe dazu bringen, ihr Guthaben/Geld für nutzlose Dienste auszugeben. Neben dem Programmierer der bösartigen App profitieren auch die Betreiber der App Stores hiervon, sie bekommen ja eine Provision bei allen Käufen. Für Benutzer muss es also immer eindeutig und klar erkennbar sein, ob ein Klick auf einen Button in einer App einen kostenpflichtigen Einkauf auslöst oder nicht.

Beteiligte Parteien: Beim Einsatz von Smartphones gibt es verschiedene beteiligte Parteien (engl. stakeholder) mit unterschiedlichen Interessen und Sicherheitsanforderungen [1].

Benutzer: Da ist als erstes der Benutzer des Smartphones. Sein Hauptinteresse liegt darin, dass seine Daten auf dem Smartphone sicher sind und dass das Gerät keinen „Unsinn“ macht. Außerdem möchte der Benutzer die völlige Freiheit mit dem Gerät haben, also beliebige Software installieren können. Das ist primär durch externe Angreifer gefährdet, der entweder direkten Zugriff auf das Gerät erlangt oder über das Netz das Smartphone angreift. Aber auch die Hersteller wollen manchmal verhindern, dass Benutzer bestimmte Änderungen am Smartphone vornehmen.

Hardware-Hersteller: Jedes Smartphone hat einen Hersteller. Er ist dafür verantwortlich, dass beispielsweise die elektromagnetische Strahlung im vorgesehenen Rahmen bleibt. Außerdem muss er sicher stellen, dass die weltweit eindeutige Gerätenummer **International Mobile Equipment Identifier (IMEI)** beispielsweise von einem Dieb nicht nachträglich verändert werden kann. Diese Daten müssen also in einem speziell geschützten Speicherbereich des

Geräts abgelegt sein. Die Schutzziele des Herstellers sind primär durch den Besitzer des Geräts (der legitime oder ein Dieb) gefährdet.

Netzbetreiber: Die Betreiber von Mobilfunknetzen haben auch oft spezielle Interessen an Smartphones, die beispielsweise subventioniert an Kunden abgegeben werden. Solche Geräte sollen nur im Netz dieses Betreibers eingesetzt werden. Auch könnte das Geschäftsmodell des Betreibers dazu führen, dass bestimmte Programme, wie beispielsweise Voice over IP Anwendungen, *nicht* vom Benutzer installiert werden sollen. Statt dessen soll der Benutzer die kostenpflichtigen Dienste des Netzbetreibers benutzen.

Software-Entwickler und Dienstanbieter: Sie möchten gerne beliebige Anwendungen installieren können und den kompletten Funktionsumfang des Smartphones in der Anwendung einsetzen können. Allerdings möchten sie nicht, dass die Benutzer des Smartphones beispielsweise die Anwendung beliebig kopieren können oder auf die Daten der Anwendung / des Dienstes zugreifen. Hierbei könnte es sich um kopiergeschütztes Material (Musik oder Filme) handeln. Der Benutzer soll diese Daten nur auf dem Gerät einsetzen können, für das er diese Daten gekauft hat.

Plattform-Anbieter: Zum Smartphone gehört auch der Anbieter des Betriebssystems. Er bietet meist auch die Entwicklungsumgebung für Anwendungsentwickler und einige „Basis-Anwendungen“ des Systems an. Der Plattform-Anbieter legt auch die Regeln fest, wie zusätzliche Software auf das Smartphone kommen soll. Das führt zum nächsten Beteiligten.

Marktplatzbetreiber: Programme für Smartphones werden i. d. R. über einen Marktplatz vertrieben. Er wird oft auch **App Store** genannt. Sein Betreiber hat ein Geschäftsmodell und verdient am Verkauf der Software für die Smartphones. An diesen Erlösen werden die Software-Entwickler normalerweise beteiligt. Unlautere Software-Entwickler oder unlautere Smartphonebesitzer können die Interessen der Marktplatzbetreiber gefährden.

Administratoren: Immer wenn Smartphones von Arbeitgebern gestellt werden und von den Angestellten auch privat benutzt werden, haben die Administratoren des Arbeitgebers Sorgen. Sie möchten gerne sicherstellen, dass vertrauliche Firmendaten nicht durch die private Nutzung des Smartphones gefährdet werden. Neben den ohnehin vorhandenen Gefährdungen führen privat installierte Apps zusätzliche (und meist größere) Gefährdungen hinzu. Unvorsichtige Smartphonebenutzer sind hier die größte Gefährdung.

1.3 Allgemeine Sicherheitsmechanismen

Schichtenarchitektur: Für alle Smartphones gibt es eine gemeinsame in Schichten angeordnete Systemstruktur, siehe Abbildung 1.1. Die Basis der Struktur ist die Hardware des Smartphones. Sie muss nicht nur die Programme ausführen, sondern auch bestimmte Sicherheitsfunktionen bereitstellen. Darüber befindet sich ein Betriebssystem, dessen Kern sich u. a. um die Steuerung der Hardware kümmern muss.

Weiterhin bietet das Betriebssystem auch die Schnittstelle zu den Anwendungspro-

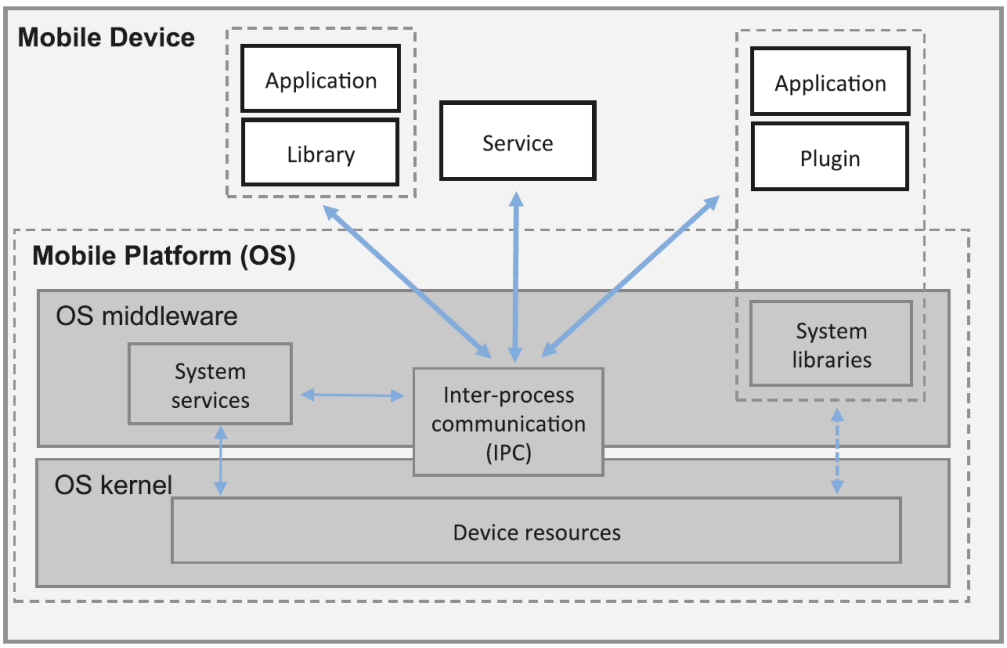

Abbildung 1.1: Architekturübersicht von mobilen Plattformen aus [1]

grammen. Sie besteht aus den System-Diensten und den System-Bibliotheken. Darauf bauen die Anwendungsprogramme auf. Anwendungsprogramme können „klassische“ Programme mit einer Benutzerschnittstelle (engl. user interface) sein oder ein Dienst (engl. service) ohne eigene Benutzerschnittstelle.

Das Betriebssystem muss auch eine Reihe von Sicherheitsfunktionen anbieten. Abbildung 1.2 zeigt, wie man sich eine nächste Detaillierungsstufe vorstellen kann. Die **Platform Security Architecture** ist eine Menge von Software-Komponenten, die einerseits die zugrunde liegende Hardware ansprechen und andererseits von den darüber liegenden Anwendungen und Diensten benutzt werden. Die Platform-Security-Architecture-Komponenten sind in den grau hinterlegten Rechtecken dargestellt. Sie werden vom Hersteller des Smartphones typischerweise mitgeliefert. Rechtecke mit weißem Hintergrund stellen die von Dritten bereitgestellten Programme oder Dienste dar. In Ovalen sind die o. g. Beteiligten am Gesamtsystem dargestellt. Sie benutzen bestimmte Komponenten der Security Architecture.

Grundprinzipien: Ausgehend von der Architektur aus Abbildung 1.2 erfüllt jede Plattform dieselben drei Prinzipien [1]:

1. *Isolierung der Programme:* Jedes Programm (engl. application) läuft in seiner eigenen Laufzeitumgebung und hat seine eigene persistente Speicherumgebung.

   Man nennt das oft auch Sandkasten (engl. sandbox). Ein Programm agiert nur in seiner Sandbox und kann nicht mit anderen Programmen, bzw. deren Sandbox, interagieren. Eine spezielle Komponente der Security Architecture, die in Abbildung 1.2 IPC (Inter-Process Communication) genannt wird, ist

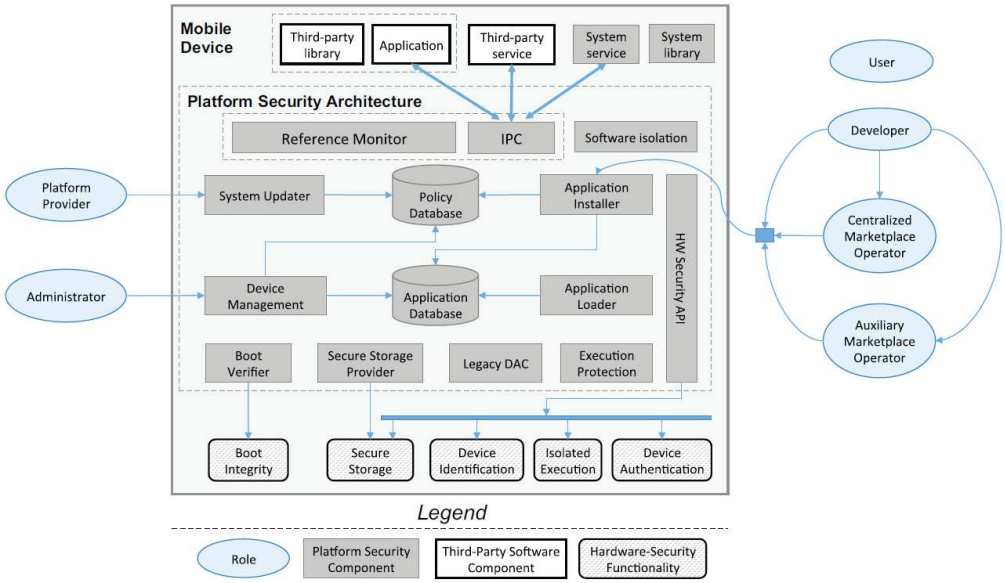

Abbildung 1.2: Modell einer Mobile Platform Security Architecture aus [1]

die einzige Möglichkeit, über die verschiedene Programme miteinander interagieren.

2. *Zugriffskontrollmodell:* Damit der Benutzer die Kontrolle behält, welche Programme auf Daten welcher anderen Programme zugreifen können, arbeitet die IPC-Komponente sehr eng mit einem sogenannten **Reference Monitor** zusammen. Nur wenn entsprechende Berechtigungen eingetragen sind, wird die Kommunikation über die IPC-Komponente erlaubt.

3. *Digital signierte Programme:* Bei der Zugriffskontrolle müssen Subjekte eindeutig authentisiert werden. Dazu werden Programme digital signiert. Dadurch wird nicht nur sichergestellt, wer das Programm geschrieben hat, sondern auch, dass das Programm unverändert vorliegt.

Software-Verteilung: Die Softwareverteilung erfolgt i. d. R. über das Internet. Auf einem Marktplatz werden Programme angeboten und die Benutzer können die Programme über den Marktplatz erwerben und herunterladen. Im Prinzip kann *jeder* Anbieter auf diesem Marktplatz sein. Ein Benutzer kann nicht ohne weiteres erkennen, ob es sich um einen seriösen Anbieter oder einen Hacker handelt.

Einige Marktplatzbetreiber versuchen den Zugang zu ihrem Marktplatz zu kontrollieren. So müssen sich Programmierer beispielsweise namentlich registrieren. Trotzdem bleibt das Problem, dass niemand weiß ob der Entwickler „Meier, Kurt aus Saarbrücken“ zu den seriösen Anbietern oder den Hackern gehört. Also werden meist auch die Programme selbst überprüft. Aber auch das ist sehr schwierig, denn man kann nicht automatisch herausfinden, ob ein Programm etwas „böses“ macht oder nicht. Das vergleichsweise viel einfachere Halteproblem ist ja schon nicht entscheidbar. Daher müssen Menschen die Programme prüfen, bevor sie auf dem Marktplatz angeboten werden.

Hat man ein passendes Programm im Marktplatz gefunden und möchte es installieren, dann ergibt sich das nächste Problem. Der Benutzer muss kontrollieren können, welche Berechtigungen das Programm denn bekommen soll. Letztlich konfiguriert der Benutzer hierbei den Reference Monitor. Das Programm muss dem Benutzer mitteilen, welche Berechtigungen es gerne haben möchte und der Benutzer muss hierüber entscheiden.

Die konkreten, evtl. sehr detaillierten Berechtigungen überfordern häufig den normalen Benutzer. Das System muss also eine vereinfachte Abstraktion des Berechtigungsmodells anbieten, so dass der Benutzer sich nicht mit den vielen Details beschäftigen muss. Das *MS Windows* Dateisystem bietet auch eine solche Abstraktion. Hinter dem „abstrakten“ Recht *Vollzugriff* verbergen sich die vielen einzelnen Rechte wie Lesen, Schreiben, Ändern, Rechte vergeben, usw.

Wichtig ist hierbei, dass dem Benutzer nur Abstraktionen angeboten werden die (1) für den Benutzer verständlich sind und (2) hinter denen sich tatsächlich auch nur die detaillierten Berechtigungen verbergen, die man normalerweise dort auch erwarten würde. Das abstrakte Recht *Kontaktdaten lesen dürfen* darf daher nicht auch den Zugriff auf das GPS-Modul erlauben oder Zugriff auf andere vertrauliche Daten ermöglichen.

Außerdem ist es wichtig, dass die Benutzerentscheidungen so gespeichert werden, dass sie nicht nachträglich von der Anwendung verändert werden können. Also darf nur ein spezielles Installationsprogramm diese Informationen erheben und speichern. Trotzdem sollte der Benutzer seine Entscheidungen auch nach der Installation eines Programms noch einmal ändern können.

Laufzeitumgebungen: Das Betriebssystem auf dem Smartphone sollte eine Benutzerverwaltung realisieren. Sie ist die Basis für die Implementierung des Reference Monitors. Alle Zugriffe von Programmen auf kritische Ressourcen müssen vom Reference Monitor vorab überprüft werden. Hat der Benutzer bei der Installation des Programms die Berechtigung erteilt, dann darf das Programm den Zugriff ausführen. Fehlt die Berechtigung, dann wird dem Benutzer häufig eine Dialogbox angezeigt. Dort steht, dass das Programm xy auf die Ressource z zugreifen will und die Berechtigung fehlt. Das wird mit der Frage an den Benutzer verbunden, ob er die Berechtigung jetzt erteilen mag.

Weiterhin müssen Laufzeitumgebungen verhindern, dass Anwendungen das Berechtigungssystem umgehen oder auf den Adressraum einer anderen Anwendung zugreifen kann. Die Konzepte hierfür sind bekannt und sie werden bei Betriebssystemen schon seit Jahren angewendet. Das Konzept der virtuellen Maschinen trennt nicht nur einzelne Prozesse, sondern komplette virtuelle Computer voneinander. Sie werden daher auch in den Betriebssystemen von Smartphones oft eingesetzt.

Persistent gespeicherte Daten, z. B. auf Speicherkarten, müssen verschlüsselt sein, damit die Vertraulichkeit der Daten auch nach Verlust der Speicherkarte gewahrt bleibt. Es ist besser, wenn sich nicht jedes Programm selbst um die Verschlüsselung der Daten kümmern muss. Statt dessen sollen die Programme so wie immer geschrieben werden. Die Laufzeitumgebung muss nun transparent für den Programmierer sicherstellen, dass alle Systemaufrufe zur Speicherung von Daten erst die erforderliche Verschlüsselung durchführen und danach die Daten auf das Speicher-

medium schreiben. In den Systemaufrufen zum Lesen muss dann natürlich eine entsprechende Entschlüsselung vorgesehen werden.

Da Smartphones oft in „unsicheren“ WLANs eingesetzt werden müssen die Netzverbindungen *immer* verschlüsselt werden. Die Smartphone-Betriebssysteme unterstützen daher alle die Verschlüsselungsprotokolle SSL bzw. TLS. Surft man mit dem Smartphone im Internet, kann man damit vertrauliche und authentische Verbindungen zu Web-Servern realisieren. Auch der Abruf von E-Mails von einem Server kann hiermit geschützt werden. Auch **Virtual Private Network (VPN)**-Software gehört zu einem Smartphone-Betriebssystem.

Management: Dieser Problembereich ist mehr organisatorischer als technischer Natur. Statt eines Firmen-Smartphones und eines privaten Smartphones möchten die meisten Menschen nur ein Smartphone mit sich herum tragen. Darauf sollen dann die geschäftlichen und die privaten Anwendungen oder Daten gemeinsam gespeichert und verwaltet werden. Sinnvollerweise möchte man eine Trennung dieser beiden Bereiche vornehmen. Auch sollen für diese beiden Bereich möglicherweise unterschiedliche Security-Policies gelten. Während Firmen daran interessiert sind, dass alle Systemaktualisierungen rechtzeitig installiert werden, möchte ein „normaler“ Anwender vielleicht lieber noch weiter mit einer älteren Version des Systems arbeiten.

1.4 Konkrete Systeme

In diesem Abschnitt werden zwei populäre Betriebssysteme für Smartphones etwas genauer betrachtet. Neben ihrer Funktionsweise soll insbesondere darauf eingegangen werden, wie die Systeme aufgebaut und wie die Sicherheitsmechanismen dort umgesetzt sind.

Android: Die Firma *Google* hat auf der Basis des Betriebssystems *Linux* das Smartphone-Betriebssysteme Android entwickelt. Wie *Linux* ist auch Android im Quelltext frei verfügbar (engl. open source) und verschiedene Hersteller vom Smartphones passen Android für ihre Geräte an und erweitern es auch falls erforderlich.

Der *Google Play Store* ist der zentrale Verteilmechanismus für Android-Programme. Neben diesen Programmen werden dort auch andere digitale Produkte vertrieben, z. B. Nachrichten, Musik, Filme oder Bücher. *Google* behält sich vor, bestimmte Programme/Inhalte im *Play Store nicht* zuzulassen. Außerdem werden alle Programme automatisch von einem Antivirus-Programm geprüft, bevor sie im *Play Store* akzeptiert werden. Leider garantiert das *nicht*, dass „schädliche“ Programme nicht in den *Play Store* gelangen.

Benutzer müssen die Programme für ein Android-Smartphone allerdings nicht über den *Google Play Store* erwerben und installieren. Man kann auch Programme direkt von der Website eines Entwicklers laden und installieren. Dazu muss der Entwickler ein sogenannten **Android Package** erstellen. Das ist eine digital signierte Archivdatei, deren Name auf **.apk** endet. Auch können andere Betreiber von Shop-Systemen für Android auftreten.

Bevor ein Programm geladen und installiert wird bzw. wenn eine bestimmte, ggf. gefährliche Funktion genutzt werden soll, werden die vom Programm angeforderten Berechtigungen angezeigt. Der Benutzer muss nun entscheiden, ob diese Berechtigungen erteilt werden sollen (dann kann das Programm installiert bzw. die Funktion ausgeführt werden) oder nicht (dann wird das Programm nicht installiert bzw. die Funktion nicht ausgeführt). Eine Manifestdatei **AndroidManifest.xml** aus dem Android Package beschreibt (1) die Komponenten, aus denen die App besteht, (2) welche Hardwareeigenschaften zur Ausführung erforderlich sind (z. B. eine Kamera für Apps, die Fotos machen können) sowie (3) die Berechtigungen, die für die Ausführung benötigt werden.

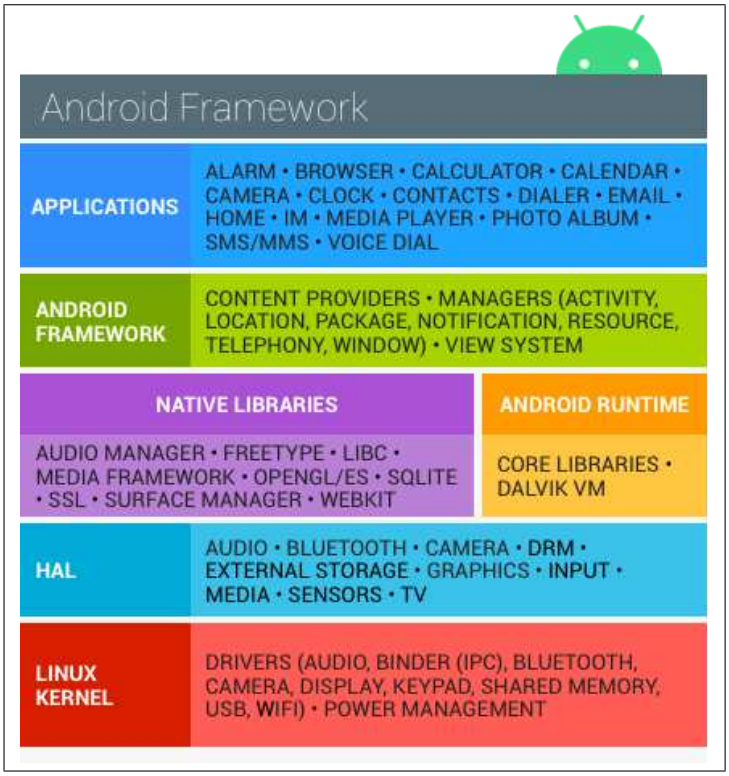

Abbildung 1.3: Übersicht über den Softwarestack von Android
Quelle: `http://source.android.com/devices/tech/security/index.html`

Abbildung 1.3 zeigt den Softwarestack von Android. Die Basis ist ein *Linux*-Kernel. Er sorgt für die Ansteuerung der Hardware und bietet bereits die im Betriebssystem angebotenen Sicherheitstechniken. Darüber liegt ein Hardware Abstraction Layer (HAL). Auf einer dritten Ebene befinden sich die Bibliotheken (engl. libraries) sowie die Laufzeitumgebung.

Programme für Android werden i. d. R. in der Programmiersprache Java (neuerdings auch in Kotlin) entwickelt. Ausgeführt werden sie dann in einer Java Virtual Machine (JVM). Jede App wird in einer eigenen JVM gestartet, die ihrerseits ein eigener Prozess im zugrundeliegenden *Linux*-System ist. Jede App bekommt eine eigene UNIX *user id*, unter der sie läuft. Die Dateizugriffsberechtigungen des Linux-Basissystems basieren auf ihnen. Dadurch wird sichergestellt, dass eine App nicht auf die Dateien einer anderen App zugreifen kann. Die Laufzeitumgebung

hieß früher Dalvik VM (siehe Abbildung 1.3), aktuell heisst sie **Android Runtime (ART)**. Sicherheitskritische Funktionen aus den Bibliotheken sind so implementiert, dass sie zunächst die *user id* der aufrufenden App nehmen und die zu dieser id vorliegenden Berechtigungen überprüfen. Nur wenn die passende Berechtigung gewährt und eingetragen wurde, wird die eigentliche Bibliotheksfunktion ausgeführt.

Die Abschottung zwischen den Anwendungen wird vom *Linux*-Kernel implementiert. Da jede Virtual Machine und damit jede Anwendung unter einer eigenen Benutzerkennung (engl. user id) in einem eigenen Prozess laufen, können über passende Zugriffsrechte (nur der Benutzer selbst darf lesen, schreiben, ausführen und die Gruppe/der Rest haben keine Berechtigungen) die betriebssystemeigenen Discretionary Access Controls (DAC) verhindern, dass eine Anwendung mit einer anderen Anwendung interagiert.

Seit Android 4.4 ist die Sicherheitsfunktion "**Verified Boot**" verfügbar, welche die Integrität der Gerätesoftware bei jedem Start überprüft. Wenn das Android-Gerät startet, wird der Bootprozess durch das Kernel-Feature *device-mapper-verity*, kurz *dm-verity* verifiziert. Hierbei werden die durch den Gerätehersteller signierten Hashwerte von zum Betriebssystem gehörenden Speicherblöcken mittels SHA-256 berechnet und mit zuvor gespeicherten Hashwerten verglichen. Ihre Unversehrtheit wird durch die Verifikation der Boot-Partition mittels eines unveränderbaren, in der Gerätehardware gespeicherten Schlüssels sichergestellt. Auf diese Weise sollen Manipulationen durch Schadsoftware und Rootkits aufgedeckt werden. Im Falle einer festgestellten Manipulation der Boot-Partition wird das Gerät mit einer entsprechenden Fehlermeldung ausgeschaltet. Ist eine andere Partition betroffen, wird der Nutzer möglicherweise darauf hingewiesen, dass das Gerät fehlerhaft ist, ihm darum nicht vertraut werden kann und es nicht ordnungsgemäß funktionieren könnte. Ein Start des Systems ist dann aber teilweise noch möglich. Somit stellt "Verified Boot" eine zuverlässige Basis für die übrigen Sicherheitsfunktionen dar. Ihr Einsatz ist seit Android 7.0 obligatorisch, um den Start kompromittierter Geräte zu verhindern.

Weitere Informationen zu Android und zur Sicherheit von Android Apps finden Sie in den eingangs genannten neueren Fachbüchern, bei Gunasekera [2] oder im Internet unter:

```
https://developer.android.com/index.html
      http://source.android.com/
```

iOS: Die Firma *Apple* hat auf das Basis von *OS X* für das eigene Smartphone *iPhone* das Betriebssystem iOS entwickelt. Es ist keine freie Software und läuft nur auf den Smartphones und Tablets der Firma *Apple*.

Programme bzw. Apps können *nur* über den *App Store* von *Apple* auf ein *iOS-Gerät* geladen und installiert werden. Voraussetzungen sind, dass die Entwickler bei *Apple* registriert sind und dass die Programme digital signiert sind. Auch *Apple* behält sich vor, Programme vor der Aufnahme in den App Store zu überprüfen. Auch dieses Verfahren kann *nicht* garantieren, dass „schädliche" Programme nicht in den App Store kommen.

In Abbildung 1.4 findet sich eine Übersicht über die Sicherheitsarchitektur in *Apple* iPhones. Sie ist hierarchisch aufgebaut und basiert auf einer Hardware, die bereits bestimmte Sicherheitseigenschaften aufweist. Im Boot-ROM befindet sich

neben dem Code zum Laden des Betriebssystems auch der öffentliche Schlüssel der Apple Wurzelzertifizierungsstelle (engl. root CA). Mit diesem Schlüssel kann man die digitalen Signaturen unter den anderen Apple-Zertifikaten prüfen und letztlich damit auch die digitalen Signaturen unter dem Betriebssystem-Code. Man kann das mit der *chain of trust* in einem **Trusted Platform Module, TPM** vergleichen.

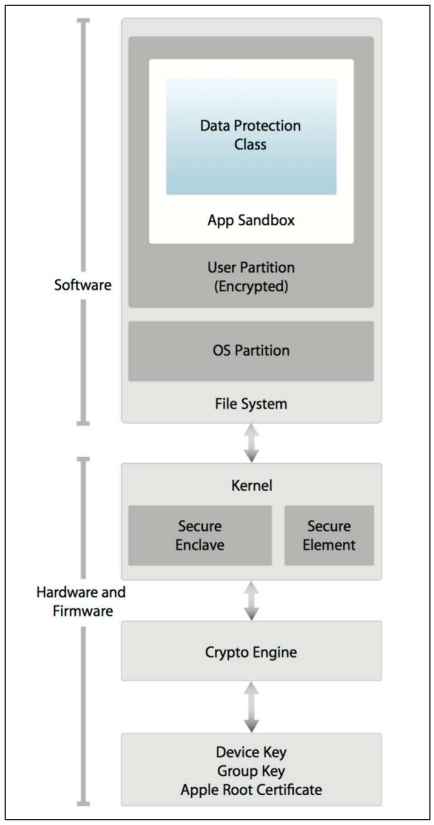

Abbildung 1.4: Übersicht über die Sicherheitsarchitektur von *Apple* iOS aus [3]

Nur Betriebssystem- oder Programm-Code mit einer gültigen digitalen Signatur wird auf dem Gerät ausgeführt. In der Hardware des Geräts sind auch Komponenten zur Unterstützung von Verschlüsselungsverfahren vorgesehen (*Crypto Engine* in Abbildung 1.4). Neben einer Hardware-Implementierung des Advanced Encryption Standard (AES) ist dort auch ein Zufallszahlengenerator implementiert.

In dieser Hardware wurde auch eine eindeutige Geräte-ID bei der Herstellung „eingebrannt“. Sie wird als Schlüssel für die Verschlüsselung der RAM-Inhalte benutzt. Außerdem werden mit diesem Schlüssel die weiteren Schlüssel geschützt. Das System erstellt weitere Schlüssel beispielsweise für jede Datei, die gespeichert werden soll. Die Datei wird dann symmetrisch mit dem weiteren Schlüssel verschlüsselt, während der weitere Schlüssel mit einem übergeordneten Schlüssel verschlüsselt wird.

Programme für iOS werden i. d. R. in einer der Programmiersprachen Objective-C oder Swift geschrieben. Von der Firma *Apple* gibt es eine vollständige Entwicklungsumgebung, genannt *XCode*. Sie enthält auch alle Bibliotheken und Hilfsprogramme, die man bei der Entwicklung benötigt.

Die Abschottung zwischen den Programmen erfolgt durch die *Mandato-*

ry Access Control (MAC) Mechanismen des zugrundeliegenden Betriebssystems *FreeBSD*. Auf ihm basiert *OS X* und darauf basiert iOS. Jedes Programm bekommt bei der Installation ein *home directory* zugewiesen, dessen Name zufällig erzeugt wird. Der Zugriff auf Daten anderer Programme ist nur über Systemfunktionen von iOS möglich. Während bei Android jedes Programm unter einer eigenen Benutzerkennung läuft, laufen „normale“ Programme unter iOS unter der Benutzerkennung *mobile*. Dieser Benutzer hat nur sehr eingeschränkte Berechtigungen. Trotzdem können die Dateisystemberechtigungen nicht verhindern, dass ein Programm den Namen des *home directory* eines anderen Programms errät und dort Dateien öffnet.

Das Lesen der Dateiinhalte wird statt dessen durch Verschlüsselung verhindert. Das System kann mit Hilfe des eingebauten Hardware-Zufallszahlengenerators sichere symmetrische Schlüssel erstellen. Mit diesem Schlüssel wird der Dateiinhalt verschlüsselt. Dieser Schlüssel wird dann so sicher gespeichert, dass nur dieses Programm auf ihn zugreifen kann. Das bedeutet, dass auch dieser Schlüssel wiederum verschlüsselt wird. Letztlich braucht man den Geräteschlüssel (engl. Device Key) (vergleiche Abbildung 1.4), um an den Schlüssel für die Datei zu kommen. Soll auch ein anderes Programm diese Datei lesen dürfen, dann muss man diesem Programm im Prinzip nur Zugriff auf den o. g. Schlüssel ermöglichen. Dieser Mechanismus verhindert auch, dass Angreifer die Flash-Speicherbausteine ausbauen und in ein anderes Gerät wieder einbauen. Das andere Gerät hat einen anderen Geräteschlüssel (der auch nicht veränderbar ist) und kann daher den Dateischlüssel nicht entschlüsseln und damit die Datei lesen. Das verhindert allerdings auch, dass man SD-Speicherkarten unter iOS benutzen kann, um Daten von einem Gerät auf ein anderes Gerät zu übertragen.[2]

Dieses Konzept erlaubt auch ein sehr einfaches, schnelles und sicheres Löschen von Dateien. Statt die ganze Datei mit Nullen zu überschreiben und dann den Verzeichniseintrag dieser Datei zu entfernen, reicht es jetzt, den Schlüssel zu löschen, mit dem die Datei verschlüsselt wurde. Danach stehen die Daten der Datei zwar noch im Flash-Speicher, sie sehen aber aus wie Zufallszahlen. Zusätzlich zum Löschen des Schlüssels muss natürlich auch in den Verzeichniseinträgen die Datei entfernt werden. Tatsächlich ist es so, dass der Zugriff auf die Schlüssel wiederum durch Verschlüsselung realisiert ist. Die Schlüssel, mit denen Dateien verschlüsselt sind, werden wiederum mit Schlüsseln verschlüsselt. Durch Löschen der „Schlüssel-Verschlüsselungs-Schlüssel“ entzieht man einem Programm die Möglichkeit, die „Datei-Verschlüsselungs-Schlüssel“ zu lesen und dann die Dateien zu entschlüsseln. Details hierzu finden Sie in dem aktuellen Platform Security Guide [4] von *Apple*.

Weitere Informationen zu iOS finden Sie in den eingangs genannten neueren Fachbüchern, bei Zdziarski [5] oder im Internet bei *Apple* unter:

`https://developer.apple.com/`

Fahren Sie bitte nun mit dem Studium der ersten beiden Kapitel des o. g. Basistextes fort, in denen insbesondere der Aufbau mobiler Applikationen unter sicherheitsbezogenen Gesichtspunkten sowie die Grundlagen ihrer technischen Analyse erläutert werden.

[2] Ganz abgesehen davon, dass die Geräte von Apple gar keinen Einschub für diese Karten haben.

Literaturverzeichnis

[1] N. Asokan, Lucas Davi, Alexandra Dmitrienko, Stephan Heuser, Kari Kostiainen, Elena Reshetova und Ahmad-Reza Sadeghi. Mobile Platform Security. Bd. 4. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool, 2013.

[2] Sheran Gunasekera. Android Apps Security. Berkeley, CA, USA: Apress, 2012. ISBN: 1430240628, 9781430240624.

[3] Apple. iOS Security – iOS 8.3 or later. Techn. Ber. Apple, April 2015. https://www.apple.com/business/docs/iOS_Security_Guide.pdf.

[4] Apple. Apple Platform Security. Techn. Ber. Apple, Fall 2019. https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf.

[5] Jonathan Zdziarski. Hacking and Securing iOS Applications: Stealing Data, Hijacking Software, and How to Prevent It. O'Reilly Media, Inc., 2012. ISBN: 1449318746, 9781449318741.