Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Christoph Behr

Untersuchung der Implementierungsmöglichkeiten sicherheitstechnischer Funktionen mobiler Geräte und deren Betriebssysteme

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Christoph Behr

Untersuchung der Implementierungsmöglichkeiten sicherheitstechnischer Funktionen mobiler Geräte und deren Betriebssysteme

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung im Studiengang Informations- und Elektrotechnik am Department Informations- und Elektrotechnik der Fakultät Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. R. Fitz
Zweitprüfer: Prof. Dr. H. Neumann

Abgegeben am: 26. Februar 2015

Christoph Behr

Thema der Bachelorthesis

Untersuchung der Implementierungsmöglichkeiten sicherheitstechnischer Funktionen mobiler Geräte und deren Betriebssysteme

Stichworte

Smartphone, Tablet, Betriebssystem, Android, Blackberry OS, Apple, iOS, Windows Phone, Sicherheitsfunktion, Malware, Schadsoftware, Rechte, Berechtigungsgruppen, Play Store, Mobilgerät

Kurzzusammenfassung

Diese Abschlussarbeit befasst sich mit den Sicherheitsfunktionen mobiler Betriebssysteme. Es wurde festgestellt, welche Funktion diese Systeme bereits integriert haben, um die Daten der Nutzer von Mobilgeräten zu sichern und vor Fremdzugriff zu schützen. Zudem wurde untersucht, wie Schadsoftware auf den mobilen Geräten installiert wird. Durch das nicht korrekte Deuten bei der Vergabe von Berechtigungen, speziell bei *Android*-Systemen, wurde eine Anwendung entwickelt, die Nutzer dabei unterstützt, diese Rechte besser bewerten zu können, wobei die Notwendigkeit der Berechtigungsgruppen im Rahmen eines Ampel-Systems dargestellt wird.

Christoph Behr

Title of the paper

Investigation and implementation options of safety-related functions from mobile devices and their operating systems

Keywords

Smartphone, tablet, operating system, android, blackberry os, apple, iOS, windows phone, security function, malware, malicious software, rights, permission groups, play store, mobile device

Abstract

In this paper, mobile operating systems is dealt with the security features. It was found which functions the systems have included to protect the data from smartphone or tablet users and unauthorized access. It was investigated how developed malware was installed on mobile devices. With indicated incorrect from rights, especially for *Android*-systems, it was developed a program to assist the user, to assess the rights of installed apps. The need of permission groups for an app is presented in the context of a traffic light system.

Danksagung

Hiermit möchte ich mich bei Herrn Prof. Dr.-Ing. Robert Fitz für seine freundliche Betreuung bedanken. Ebenfalls bedanken möchte ich mich bei Frau Prof. Dr. Heike Neumann, die sich bereit erklärt hat, bei dieser Arbeit das Zweitgutachten zu übernehmen.
Des Weiteren möchte ich mich bei meinen Kollegen der Alfred Kuhse GmbH für die Unterstützung bei diesem Thema bedanken.
Ebenfalls geht ein Dank an die Personen, die sich die Mühe gemacht haben, diese Arbeit Korrektur zu lesen und mir somit eine große Hilfe bei der Fertigstellung waren.
Insbesondere danke ich meinen Eltern, die mich während meines Studiums in allen Lagen unterstützt haben.

Christoph Behr, Februar 2015

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Mit der zunehmenden Verbreitung von Smartphones und Tablets spielen die Betriebssysteme und deren Sicherheit eine entscheidende Rolle. Heutzutage werden immer mehr empfindliche Daten auf mobilen Geräten gespeichert, auf die in der Regel nur die Nutzer des Gerätes Zugriff haben sollten. Es ist interessant, herauszufinden, welche Sicherheitsfunktionen die jeweiligen Betriebssysteme beinhalten und wie sie die Daten von Nutzern schützen. Ein weiterer Punkt ist die Ermittlung, wie die Schadsoftware-Entwickler es schaffen, Schadsoftware auf den mobilen Systemen zu installieren. Es ist zu prüfen, welchen Einfluss die Nutzer bei der Installation von Applikationen[1] haben und ob eine Möglichkeit besteht, den Anwender besser auf eine ungewollte Installation von Schadsoftware hinzuweisen.

1.2 Aufgabenstellung der Arbeit

Im Rahmen dieser Arbeit werden unterschiedliche Betriebssysteme von mobilen Geräten, wie Smartphones oder Tablets und deren Sicherheitsfunktionen untersucht. Es wird ermittelt, welche mobilen Betriebssysteme auf dem Markt am weitesten verbreitet sind. Diese Systeme werden auf ihren Aufbau und deren Funktionsweise überprüft. Auf Grundlage dieser Erkenntnisse wird dargestellt, wie auf den jeweiligen Systemen Programme ausgeführt werden. Dadurch, dass in dieser Arbeit die Sicherheitsfunktionen der mobilen Betriebssysteme untersucht werden, muss ebenfalls geprüft werden, wie sich bereits vorhandene Schadsoftware gegen die einzelnen Systeme richtet und wie diese auf den Geräten installiert werden. Mit den gewonnen Erkenntnissen wird abschließend eine Applikation entwickelt, die die Nutzer bei der Erkennung von Schadsoftware unterstützt.

1.3 Aufbau der Arbeit

Zu Beginn dieser Arbeit wird in Kapitel „Verteilung der Betriebssysteme auf dem Markt“ ab Seite 9 ein Überblick über die vertretenen mobilen Betriebssysteme auf dem Markt dargestellt. Als Referenz wird der Zeitraum zwischen 2010 und 2014 betrachtet und untersucht, wie sich die jeweiligen mobilen Betriebssysteme auf dem Markt während dieser Zeit entwickelt haben. Im Anschluss an die Darstellung der unterschiedlichen Marktentwicklungen, werden diejenigen mobilen Betriebssysteme detaillierter betrachtet, die die größte Marktpräsenz aufweisen. Diese werden hinsichtlich in Aufbau und Funktionsweise in Kapitel „Übersicht der mobilen Betriebssysteme“ ab Seite 14 näher betrachtet. Es wird weiterhin dargestellt, wie Anwendungen auf den jeweiligen Systemen ausgeführt werden. Darüber hinaus wird in Kapitel „Vergleich der Betriebssysteme“ ab Seite 39 erläutert, welche Voraussetzungen App-Entwickler haben müssen, um Anwendungen für die untersuchten Systeme zu entwickeln und was von den Herstellern unternommen wird, damit die entwickelten Programme auf den Geräten installiert werden können bzw. zum Download in den jeweiligen App-Stores[2] zur Verfügung gestellt werden. Es werden die bereits integrierten Sicherheitsfunktionen der unterschiedlichen Betriebssysteme aufgezeigt und wie

[1] Applikationen [deutsch: Anwendung], kurz App, werden Computerprogramme genannt, welche bestimmte Aufgaben mit Hilfe eines Rechnersystems durchführen. Dieses kann z.B. ein Bildverarbeitungsprogramm sein.

[2] Bei einem App-Store handelt es sich um einen Service von Betriebssystem-Herstellern, über die kostenfreie oder kostenpflichtige Anwendungen auf den mobilen Geräten installiert werden können.

sie die Daten von Nutzern schützen sollen. Um einen besseren Eindruck zu vermitteln, wie Schadsoftware auf den Geräten installiert wird, wird in Kapitel „Schadsoftware“ ab Seite 42 untersucht, welche Angriffsmöglichkeiten es gegen die betrachteten Betriebssysteme gibt, welche Angriffe es bereits in der Vergangenheit gegen die Systeme gegeben hat und wie diese ausgeführt wurden. Zum Abschluss dieser Arbeit wird eine eigens entwickelte Applikation in Kapitel „Entwicklung einer eigenen *Android*-Anwendung“ ab Seite 52 vorgestellt, die die Nutzer von *Android*-Geräten dabei unterstützt, Berechtigungsgruppen, die eine zu installierende Anwendung fordert, in Form eines Ampel-Systems besser bewerten zu können.

2 Erläuterung von mobilen Betriebssystemen

Da in Mobilgeräten[3], womit in dieser Arbeit Smartphones und Tablets gemeint werden, mehrere unterschiedliche Module miteinander vernetzt sind, sind sie als ein System zu betrachten. Die einzelnen Einheiten bzw. Module haben zum Teil eine eigene Firmware[4], wodurch diese unabhängig vom Rest des Systems ausgeführt werden können.[2]
Um die integrierte Hardware von mobilen Geräten nutzen zu können, benötigen diese ein Betriebssystem, die eine Steuerungssoftware für die Prozessoren darstellt, welche in der Hardware verbaut sind. Durch das Betriebssystem erhalten die Prozessoren Zugriff auf Arbeitsspeicher, die Tastatur und das Mikrofon zur Eingabe von Text und Sprache, das Display und den Lautsprecher zur Ausgabe, sowie die Elektronik-Komponenten, die zum Senden und Empfangen von Daten per Mobilfunk dienen. Über das Betriebssystem werden die Funktionen der mobilen Geräte zum Verwalten von Adressen oder Terminen, das Multitasking[5] und die Software von Drittanbietern gesteuert. Es regelt ebenfalls, wie diese gegen Fremdzugriff gesperrt und wie sie beispielsweise durch ein Passwort entsperrt werden. Des Weiteren wird festgelegt, wo die Adressen, Termine, E-Mails und Multimedia-Dateien des Anwenders gespeichert werden. Zudem stellt es auch den Browser, den Multimedia-Player, die Digitalkamera bzw. die Videokamera, die GPS-Funktionen sowie die Bluetooth- und WLAN-Kommunikation bereit. [31]
Grundsätzlich ist die Systemarchitektur der mobilen Betriebssysteme in mehreren Schichten aufgebaut. In der Regel ist sie durch einen Kern konstruiert, eine Schicht für grundlegende Funktionen und Bibliotheken, sowie weitere Schichten auf denen Anwendungen ausgeführt werden. Der detaillierte Aufbau dieser Schichten ist betriebssystem-spezifisch und bildet eines der Abgrenzungskriterien unter den verschiedenen mobilen Systemen.[32]

[3] Unter Mobilgeräten versteht man alle elektronischen Geräte, die für mobile Daten-, Sprach- und Bildkommunikation sowie Navigation, Datenspeicherung und das Mobile Computing, entwickelt wurden. In Mobilgeräten, wie Smartphones und Tablets werden verschiedene Funktionalitäten zusammengefasst, wodurch beispielsweise Navigation und Videotelefonie ermöglicht wird.[39]

[4] Mit einer Firmware ist eine fest installierte Softwareroutine gemeint, die direkt in dem Gerät gespeichert ist. Mit dieser Software werden die Grundfunktionen eines Gerätes bzw. Moduls festgelegt.

[5] Multitasking bedeutet, dass mehrere Aufgaben bzw. Prozesse in der gleichen Zeit ausgeführt werden.

3 Verteilung der Betriebssysteme auf dem Markt

Um einen Überblick über die mobilen Betriebssysteme und deren Marktanteile zu schaffen, werden für die Jahre 2010 bis 2014 Untersuchungen durchgeführt. Es wird zum einen die Verbreitung für Deutschland und zum anderen die Verbreitung weltweit ermittelt und grafisch dargestellt. Die Verteilung auf dem Markt wird anhand der verkauften Geräte mit dem jeweils installierten Betriebssystem gemessen.

3.1 Verbreitung der Betriebssysteme weltweit

In Abbildung 1 auf Seite 10 ist zunächst die Entwicklung von mobilen Betriebssystemen in den Jahren 2010 bis 2013 dargestellt. Aus dieser Abbildung ist zu entnehmen, dass im Jahr 2010 37,6% der verkauften Smartphones das *Symbian*-System installiert hatten, womit das Betriebssystem Marktführer war. Mit einem um ca. 15% geringeren Anteil folgt das *Android*-System mit 22,7%. Die mobilen Betriebssysteme von *Blackberry RIM* und *Apples iOS* hatten 2010 einen etwa gleich großen Marktanteil. *Blackberry RIM* erreichte 16% und *Apples iOS* 15,7%. Die Systeme der Firma *Microsoft* können in diesem Zeitraum, mit 4,2% Marktanteil, keinen großen Einfluss im Bereich der mobilen Betriebssysteme verzeichnen. Betriebssysteme wie *Bada*[6] konnten sich 2010 nicht am Markt etablieren. Diese und andere Systeme erreichen einen Marktanteil von 3,8% und werden unter dem Punkt „Sonstige“ aufgeführt.
Aus Abbildung 1 ist festzustellen, dass das *Android*-System der Firma Google seinen Marktanteil zwischen 2010 und 2011 um 26,5% steigern konnte. Auch das *Apple iOS*-Betriebssystem konnte seinen Marktanteil in dieser Zeit vergrößern. Es ist zu entnehmen, dass *Microsoft* im Jahr 2011 einen Marktanteil von 1,8% verzeichnet. Grund dafür ist, dass *Microsoft* das *Windows Phone*-Betriebssystem neu entwickelt hat, welches erst im Februar 2010 auf dem Markt erschienen ist. Der Vorgänger dieses Systems war *Windows Mobile*, welcher im Jahr 2010 einen Marktanteil von 4,2% hatte. *Blackberry RIM* hat im Jahr 2011 nur noch einen Marktanteil von 10,3%, was eine Verringerung um 5,7% im Vergleich zum Vorjahr bedeutet. Das Betriebssystem *Symbian* hat zwischen 2010 und 2011 den stärksten Rücklauf zu verzeichnen. Es verlor in diesem Zeitraum 20,7% an Marktanteil.
Die Betrachtung der Jahre 2012 und 2013 zeigt deutlich, dass das Betriebssystem *Android* die höchste Marktpräsenz aufweist. In diesem Zeitraum konnten *Android* seinen Marktanteil stetig ausbauen und erreichte 2013 einen Anteil von 78,6%. Das *Apple iOS*-System hat an Einfluss verloren, aber hält mit 15,2% den zweitgrößten Anteil. Im Jahr 2013 haben *Blacberry RIM* und *Symbian*, im Vergleich zum Vorjahr, weiter an Markteinfluss verloren. *Blackberry RIM* erreicht einen Marktanteil von 1,9%. Das *Symbian*-System verzeichnet einen Anteil von 0,2% und ist somit fast komplett vom Markt verschwunden.

[6] Bei *Bada* handelt es sich um ein Betriebssystem für mobile Geräte. Dieses System wurde von *Samsung Electronics* zwischen 2010 und 2013 auf deren Smartphones in Asien und Europa eingesetzt. Von Samsung wurde 2013 bekannt gegeben, dass sie dieses System nicht weiter entwickeln. Teile des *Bada*-Systems und dessen Apps brachte *Samsung* in das Nachfolger-Betriebssystem *Tizen* ein.[74]

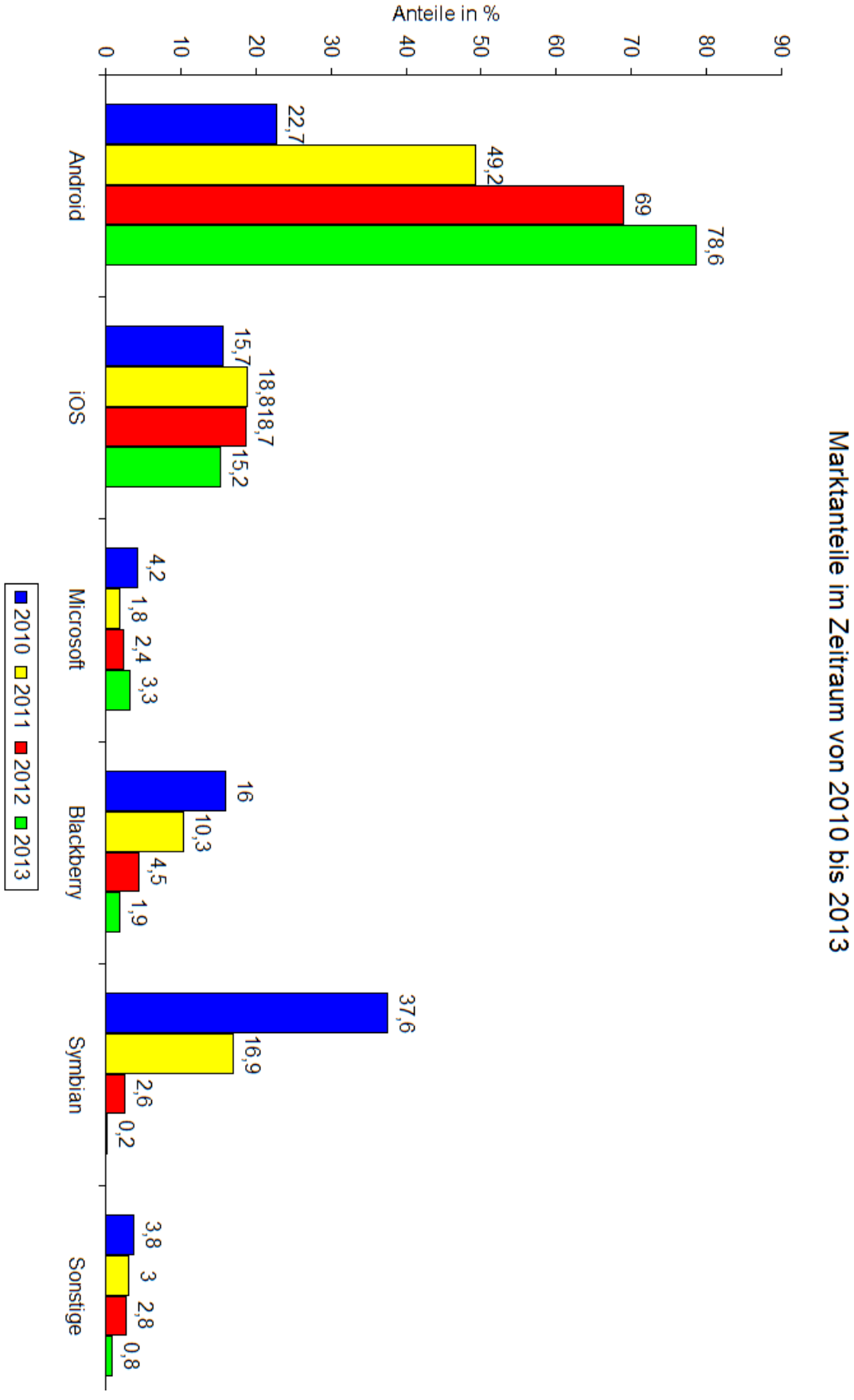

Abb. 1: Marktanteile weltweit von 2010 bis 2013 [63] [65] [63]

In der nachfolgenden Abbildung 2 ist eine Prognose zu den Marktanteilen der Betriebssysteme am Absatz von Smartphones weltweit für die Jahre 2014 und 2018 dargestellt. Aus dieser Prognose ist zu entnehmen, dass das *Android*-Betriebssystem geringfügig an Markteinfluss verliert, jedoch weiterhin den mit Abstand größten Marktanteil besitzt. Einen deutlichen Anstieg kann das Betriebssystem *Windows Phone* von *Microsoft* verzeichnen. Dieses System zeigt einen Anstieg von 3,5% auf 6,4%. Es ist weiterhin zu erkennen, dass das *Blackberry*-Betriebssystem fast vollständig vom Markt verschwunden ist. Es weist im Jahr 2018, laut der Prognose, nur noch 0,3% Marktanteil auf.

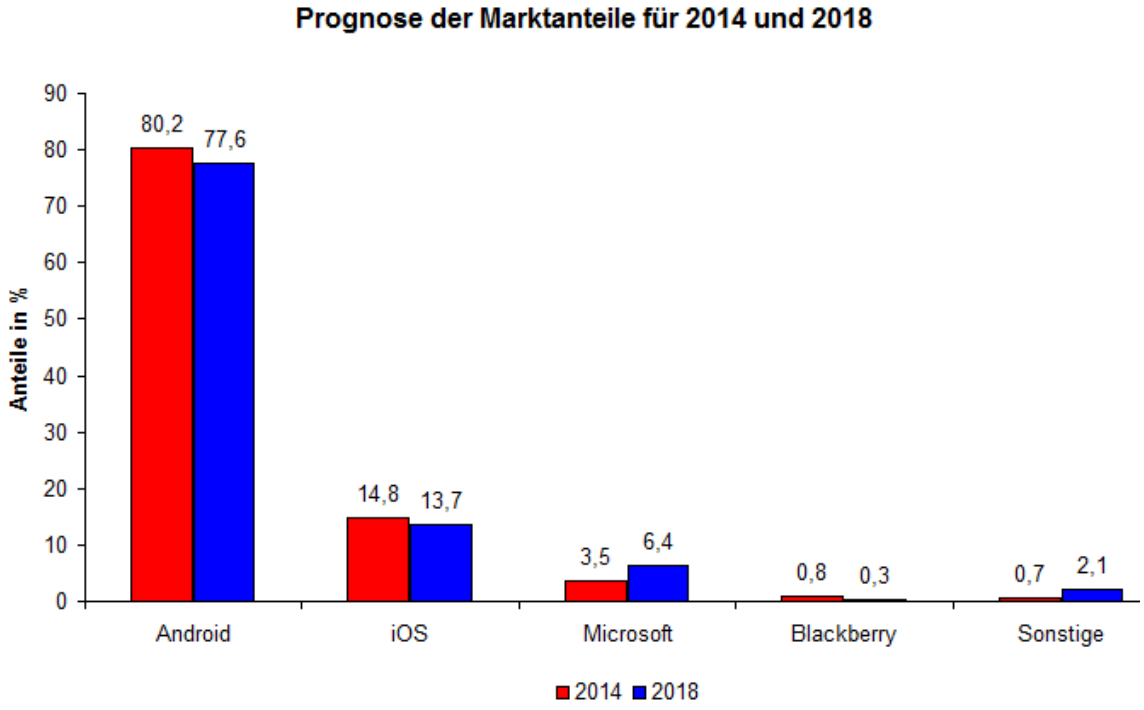

Abb. 2: Prognose der Marktanteile weltweit für 2014 und 2018 [68]

Aus der Prognose ist für das Jahr 2018 zu entnehmen, dass das *Apple iOS*-System ca. 1,1% an Marktanteil verliert. Die Systeme, welche in Abb. 2 unter dem Punkt „Sonstige“ dargestellt sind, können in den Jahren von 2014 bis 2018 ihren Marktanteil vergrößern. Diese mobilen Betriebssysteme haben im Jahr 2018 einen Marktanteil von 2,1%.

3.2 Verbreitung der Betriebssysteme in Deutschland

In Abbildung 3 auf Seite 13 ist die Marktentwicklung der unterschiedlichen mobilen Betriebssysteme für Deutschland im Zeitraum von 2010 bis 2014 dargestellt. Aus dieser Abbildung geht hervor, dass im Jahr 2010 das Betriebssystem *Symbian* den größten Anteil am Markt verzeichnen konnte. Das System war zu diesem Zeitpunkt mit 48% Marktanteil vertreten. Ebenfalls auf dem Markt vertreten waren im Jahr 2010 *Android* mit 11%, *Apple iOS* mit 20%, *Microsoft* mit 12% und das Betriebssystem von *Blackberry* mit 5%.
Aus der Abbildung geht weiter hervor, dass das Betriebssystem *Android* im Jahr 2011 bereits einen Marktanteil von 33% verzeichnen konnte. Dies ist ein Zuwachs von 22%. In diesem Zeitraum konnte *Apples iOS* mit 20,4% seinen Anteil am Markt geringfügig vergrößern. Die mobilen Betriebssysteme von *Microsoft* und *Blackberry* haben im Gegensatz zum Vorjahr an Markteinfluss verloren. *Microsoft* hat in dem Zeitraum von 2010 bis 2011 einen Verlust von 4% zu verzeichnen und erreicht somit einem Marktanteil von 8%. *Blackberry* verringert seinen Anteil im Vergleich zum Vorjahr um 2%. Den größten Verlust hat 2011 das *Symbian*-System. Dieses verliert 16% und kommt auf 32% Marktanteil.
Für das Jahr 2012 ist aus Abbildung 3 zu entnehmen, dass das *Android*-System seinen Marktanteil auf 50,1% weiter ausbauen konnte. Das *iOS*-Betriebssystem von *Apple* konnte genauso wie 2011 seinen Anteil etwas vergrößern. *Microsoft*, *Blackberry RIM* und *Symbian* haben im Jahr 2012 weiter an Einfluss auf dem Markt verloren. *Microsoft* verzeichnet einen Marktanteil von 5,8%. Dies ist ein Verlust von 2,2% zum Vorjahr. *Blackberry RIM* besitzt einen Anteil von 2,4%. Dies sind 0,6% weniger als im Jahr davor. Das Betriebssystem *Symbian* hat in diesem Zeitraum den stärksten Rücklauf. Dieses System musste in den Jahren von 2011 bis 2012 einen Verlust von 24,4% verzeichnen und erreicht dadurch 17,6% Marktanteil.
Im Jahr 2013 konnte *Android* seine Stellung auf dem Markt, wie schon die Jahre zuvor, um 13,7% weiter ausbauen. Dieses System hält in dem Jahr einen Marktanteil von 63,8%. Das *Apple iOS*-System hat in diesem Zeitraum einen leichten Rücklauf von 1,5%, genauso wie *Microsoft* mit 0,5% und *Blackberry RIM* mit 1,1%. Das Betriebssystem *Symbian* hat seinen Marktanteil weiter stark verringert. Dieses muss einen Verlust von 9% verzeichnen und erlangt dadurch einen Anteil am Markt von 8,6%.
Das *Android*-System konnte im Jahr 2014 seinen Marktanteil weiter ausbauen. Das System konnte in den Jahren von 2013 bis 2014 seine Anteil um 4,4% erhöhen. Mit 68,2% hat dieses Betriebssystem mit Abstand die größte Marktpräsenz. Das *iOS*-System hat im Gegensatz zum Vorjahr seinen Marktanteil geringfügig vergrößert und erreicht 20%. Die Betriebssysteme von *Microsoft* und *Blackberry* haben in diesem Zeitraum weiter an Markteinfluss verloren. *Microsoft* hält 5,2% Marktanteil und *Blackberry RIM* 0,3%. Das *Blackberry RIM*-System ist somit fast komplett vom Markt verdrängt worden. Das Betriebssystem *Symbian* hat genauso wie die Jahre vorher weiter an Markteinfluss verloren und erreicht im Jahr 2014 einen Anteil von 3,9%.

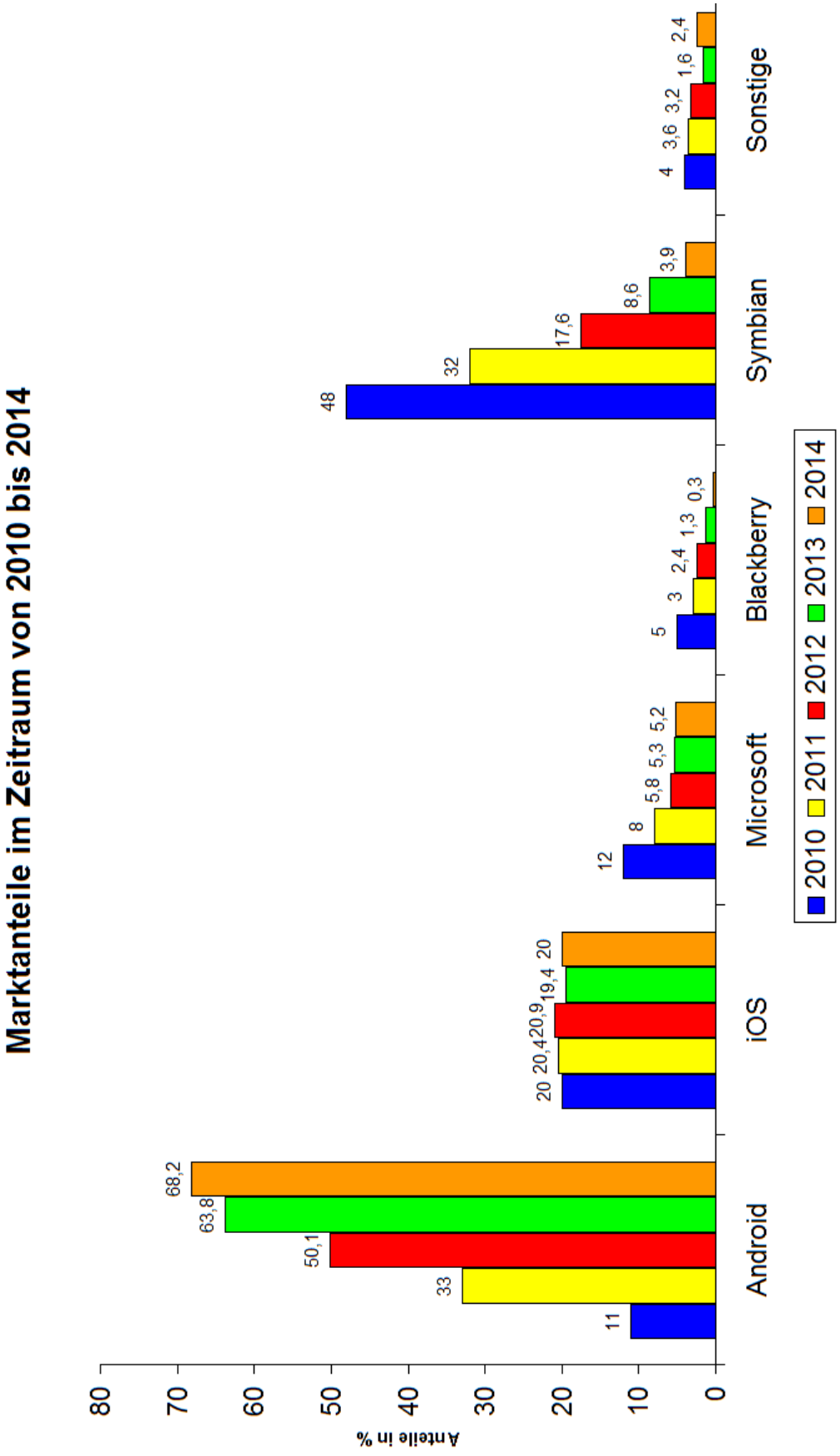

Abb. 3: Marktanteile in Deutschland von 2010 bis 2014 [11] [30] [66] [67]

4 Übersicht der mobilen Betriebssysteme

In diesem Kapitel werden die mobilen Betriebssysteme, die auf dem Markt die größte Präsenz aufweisen, genauer untersucht. Die zu untersuchenden Betriebssysteme sind *Android*, *Apple iOS*, *Blackberry-OS*, *Symbian* und *Windows Phone*. Das *Symbian*-System hat derzeit fast keinen Einfluss mehr auf dem Markt, wird aber dennoch mit untersucht, da es im Jahr 2010 weltweit Marktführer auf dem Gebiet der mobilen Betriebssysteme war (siehe Abb.1, Seite 10).

4.1 Android

Android ist ein von der *Open Handset Alliance* (gegründet von Google[7]) entwickeltes Betriebssystem sowie Software-Plattform für mobile Geräte. Bei dieser Software handelt es sich um eine freie Software, welche quelloffen entwickelt wird. Der Programmcode des Systems ist frei verfügbar, sodass Anbieter von Mobilgeräten den Quellcode für ihre eigenen Produkte anpassen können. Für Updates und Software-Aktualisierungen sind die Anbieter der Geräte zuständig. Der Grund dafür ist, dass eine aktualisierte *Android*-Version auf die unterschiedlichen Geräte der Hersteller angepasst werden muss. Somit sind die Anbieter für Mobilgeräte dafür zuständig Software-Aktualisierungen für ihre Produkte bereit zu stellen. Zusatzprogramme bzw. Applikationen, kurz: Apps werden in einer großen Anzahl in *Googles Play Store* bereit gestellt. Zudem können auch Apps von anderen Quellen installiert werden. [73] [76]
Obwohl *Android* einen Linux-Kernel beinhaltet, ist es keine klassische Linux-Distribution. Für das Betriebssystem wurden teilweise grundlegende Eingenschaften, die man von einem unixoiden System[8] erwarten würde, stark verändert. Da *Android* viele Eigenschaften mit zahlreichen Embedded-Linux-Distributionen teilt, wird es teilweise als Linux-Distribution angesehen. [88]

4.1.1 Aufbau und Funktionsweise von Android

Das *Android*-Betriebssystem basiert auf einem Linux-Kernel, welcher die erforderlichen Hardwaretreiber z.B. für Bildschirm, Kamera, Tastatur usw. zur Verfügung stellt. In diesem Kernel finden ebenfalls Verwaltungsprozesse statt, die z.B. Energie- und Speicherverwaltung vornehmen (siehe Abb. 5, Seite 16). Das Herzstück der Android-Laufzeitumgebung ist die *Dalvik Virtual Machine(DVM)*[9], die vom *Google*-Mitarbeiter Dan Bornstein entwickelt wurde. Die *Dalvik Virtual Machine* basiert auf der quelloffenen *Java-Virtual-*

[7] Bei Google handelt es sich um ein US-amerikanisches Unternehmen, welches 1998 von Larry Page und Sergey Brin gegründet wurde. Dieses Unternehmen betriebt die Internet-Suchmaschine Google.[28]

[8] Bei einem *unixoiden System* handelt es sich um ein Betriebssystem für einen herkömmlichen PC oder ein mobiles Gerät, welches versucht die Verhaltensweisen von Unix zu implementieren. Unix ist ein Mehrbenutzer-Betriebssystem, das 1969 entwickelt wurde.[20]

[9] Die *Dalvik Virtual Machine* ist eine für Mobilgeräte virtuelle Ausführungsumgebung. In der *DVM* wird Software, die für eine *Java Virtual Machine (JVM)* übersetzt wurde, ausgeführt. Um die Software auszuführen, muss der Code in Bytecode-Format konvertiert werden. Die *DVM* wurde so entworfen, dass in ihr Registermaschinencode ausgeführt werden kann, womit diese ressourcenschonend und schnell ist.[14]

Machine(JVM)[10] *Apache Harmony*. Der Aufbau und Funktionsumfang der JVM wurde dabei an die Anforderungen mobiler Endgeräte angepasst. Jede ausgeführte Anwendung wird vom *Android*-System in einer DVM gestartet und bekommt zudem eine weitere DVM zur Verfügung gestellt.[5, S.15]
Die *Dalvik Virtual Machine* arbeitet mit einem eigenen Bytecode[11]. Die Entwicklung von Programmen erfolgt in der Programmiersprache Java und wird in Java-Bytecode kompiliert. Dieser erzeugte Java-Bytecode muss, um das Programm in der DVM lauffähig zu machen, von einem Tool in dex-Bytecode (Dalvik Excecutable Bytecode) für die virtuelle Maschine übersetzt werden. Verdeutlicht wird dieser Vorgang in Abb. 4. Der nach unten gerichtete Pfeil beschreibt den Deployment-Prozess auf das Mobilgerät.[5, S.17]

Die nachfolgende Abbildung zeigt die Erstellung einer *Android*-Anwendung.

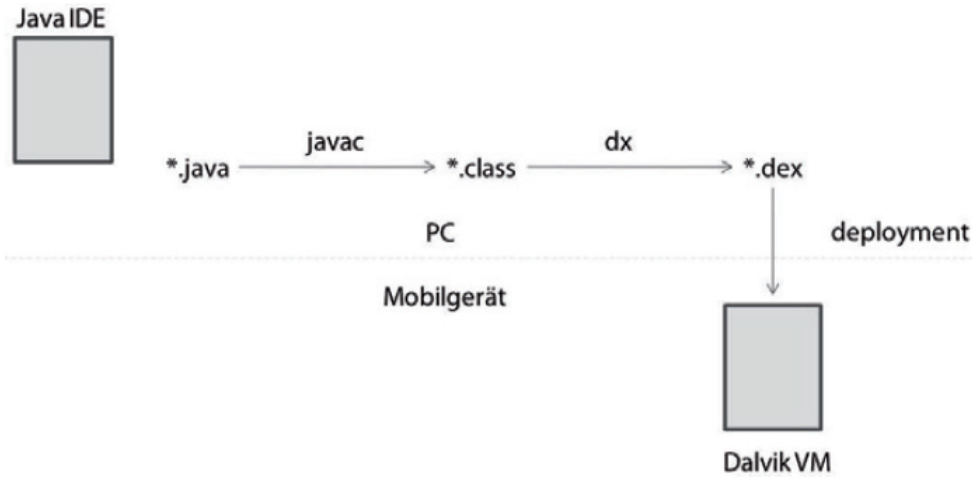

Abb. 4: Erstellung einer *Android*-Anwendung[5, S.17]

Die *Android*-Architektur setzt sich aus folgenden Schichten zusammen: Linux-Kernel, *Android*-Laufzeitumgebung, Bibliotheken, Anwendungsrahmen und Anwendungsschicht. In Abb. 5 auf Seite 16 ist dieser Systemaufbau dargestellt. Die dargestellten Schichten aus Abb. 5 werden nachfolgend erläutert.

Linux-Kernel:
Der Linux-Kernel beschreibt die unterste Schicht in der *Android*-Architektur. Von diesem Kernel werden Gerätetreiber zur Verfügung gestellt und darüber hinaus werden Verwaltungsprozesse, wie Energie- und Speicherverwaltung geregelt.

Android-Laufzeitumgebung:
Das Ausführen in einer *Dalvik Virtual Machine (DVM)* ist mit einem erheblichen Ressourcenaufwand verbunden, bietet jedoch Vorteile bezüglich Sicherheit und Verfügbarkeit.

[10] Bei der *Java Virtual Machine* (JVM) handelt es sich um eine Software, die Bestandteil der Java Runtime Environment ist und zum Ausführen von Java-Programmen benötigt wird. Die JVM ist nicht betriebssystemabhängig, sondern ein natives Programm für das jeweilige System. Es können Java-Programme auf jedem System ausgeführt werden, für das eine den Spezifikationen entsprechende JVM verfügbar ist.[37]
[11] Ein Bytecode ist in der Informatik eine Sammlung von Befehlen für eine virtuelle Maschine.[36]

Durch die DVMs sind die einzelnen Anwendungen voneinander getrennt und teilen sich daher keine gemeinsamen Speicher. Sollte ein Prozess einer Anwendung nicht mehr ausführbar sein (z.B. Deadlock[12]), muss lediglich die dazugehörige Anwendung beendet werden. Weitere Anwendungen sind davon nicht betroffen und können somit weiterhin ausgeführt werden.[5, S.15]

Die nachfolgende Abbildung zeigt den Aufbau des *Android*-Betriebssystems.

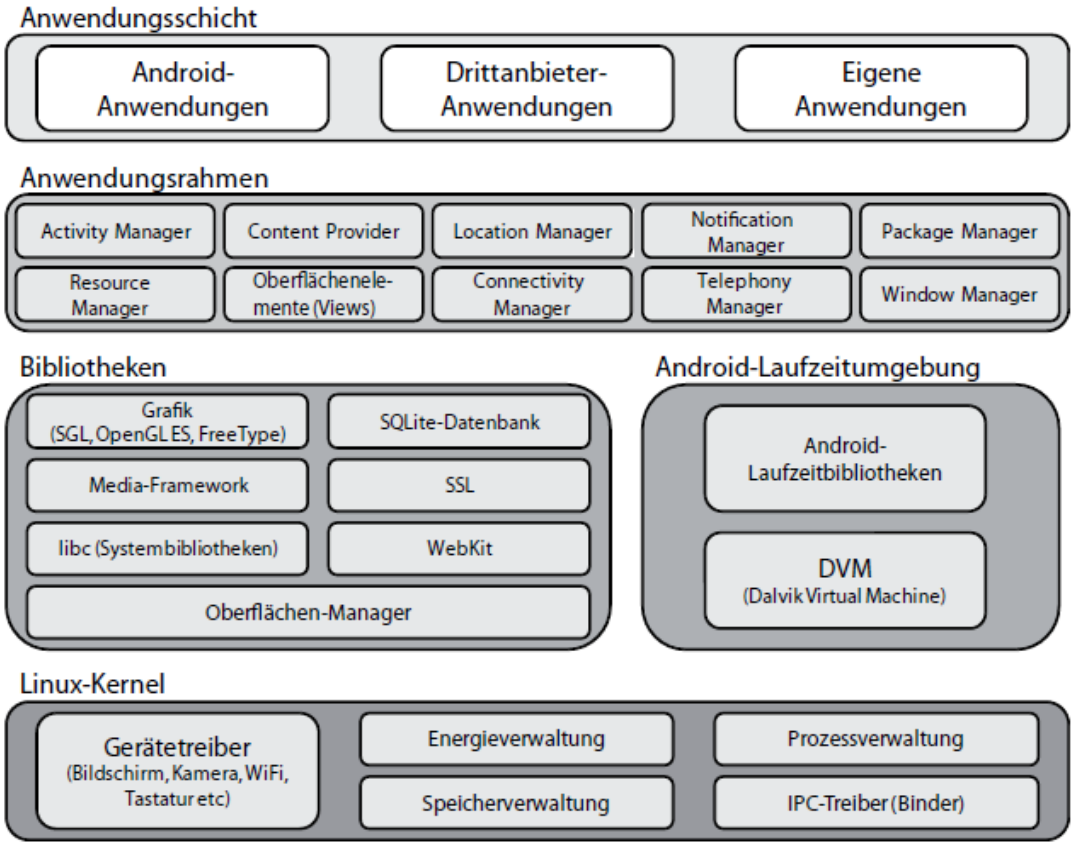

Abb. 5: Systemaufbau des *Android*-Betriebssystems [5, S.15]

Bibliotheken:
In den Bibliotheken werden alle erforderlichen Funktionalitäten, wie Datenbank, 3D-Grafikbibliotheken usw. für *Android*-Anwendungen bereit gestellt. Diese Bibliotheken sind in C bzw. C++ geschrieben. Ein fester Bestandteil des Systems sind Standardbibliotheken, die von Anwendungsentwicklern nicht verändert werden können.[5, S.16]

Anwendungsrahmen:
Durch den Anwendungsrahmen wird vom System die Kommunikation zwischen einzelnen Anwendungen ermöglicht.[5, S.16]

[12]Ein Deadlock [deutsch: Verklemmungen] kann auftreten, wenn ein Prozess auf eine Bedingung wartet, die nicht mehr erfüllt werden kann. [69, S.152]

Anwendungsschicht:
In dieser Schicht findet die Kommunikation zwischen Benutzer und Anwendungen statt, sowie die Kommunikation der einzelnen Anwendungen untereinander.[5, S.16]

Ausführung einer Anwendung:
Ein Programm wird vom *Android*-System in einer Sandbox[13] ausgeführt. Dadurch wird dem Programm ein eingeschränkter Bereich vom System zur Verfügung gestellt, in welchem die Java-Anwendung ausgeführt wird. Das Sicherheitskonzept der Sandbox basiert auf Betriebssystemfunktionen, wie Gruppen- und Benutzerberechtigungen. Jede Anwendung wird als eigener Prozess gestartet und ist genau einem Benutzer auf Betriebssystemebene zugeordnet. Es wird vom Betriebssystem für jede Anwendung automatisch eine eindeutige Linux-User-ID zugewiesen. Auf diese Weise benutzt *Android* für die Sandbox das Berechtigungskonzept des Betriebssystems. Um Zugriff auf Systemfunktionen und Ressourcen zu erhalten, werden Berechtigungen bei der Programmierung im *Android-Manifest*[14] festgelegt, damit sie zum Zeitpunkt der Installation bekannt sind. Durch das *Android-Manifest* wird der *Android*-Laufzeitumgebung weiterhin mitgeteilt, aus welchen Komponenten sich eine Anwendung zusammensetzt. Auf diese Weise kann z.B. der Zugriff auf Arbeitsspeicher, Speichermedien, Netzwerk, Telefonfunktionen, SMS usw. reglementiert werden. Wenn keine Berechtigungen im *Android-Manifest* vergeben wurden, kann das Programm nicht außerhalb der Sandbox agieren. Durch die Vergabe von Berechtigungen an ein Programm wird die Sandbox, in der das Programm läuft, „aufgeweicht“. Die Anwendung kann mit einer entsprechenden Berechtigung auf Speicherbereiche zugreifen, die nicht der Sandbox zugeteilt wurden. Durch das Verlassen des Speicherbereichs der Sandbox, wird das Sandbox-Prinzip aufgelöst und die Anwendungen beschränken sich nicht mehr auf einen isolierten Bereich.[5, S.24] Für das Android-Betriebssystem ist jede gestartete Anwendung ein eigener Prozess. Um die laufenden Prozesse zu unterscheiden, wird vom System jedem Prozess eine *Prozess ID (PID)*[15] zugeteilt. Nach dem Start eines Prozesses wird in diesem immer ein Thread[16] gestartet, der sogenannte *UI-Thread* oder *user interface thread*. Dieser Thread wird so bezeichnet, da er für die sichtbaren Bestandteile der Anwendung zuständig ist. In einem Prozess können mehrere Threads gestartet werden. Sie laufen parallel zur Hauptanwendung und sind dabei immer an einen Prozess gekoppelt. Aufgrund dieser Eigenschaft werden sie automatisch beendet, sobald der übergeordnete Prozess beendet wird.[5, S.106]

[13] Sandbox ist die englischsprachige Bezeichnung für Sandkiste und bezeichnet allgemein einen isolierten Bereich, innerhalb dessen jede Maßnahme keinerlei Auswirkung auf die äußere Umgebung hat.

[14] Bei *Android* benötigt jede erstellte Anwendung die Datei *Manifest.xml*. Über diese Datei werden dem Betriebssystem Informationen und Eigenschaften über die Anwendung mitgeteilt. In dem *Manifest* stehen auch die Berechtigungen, die eine Anwendung für die Funktionalität benötigt. Das *Manifest* muss dem System vorher der Installation bekannt sein, damit die Anwendung auf dem System lauffähig ist.[3]

[15] Bei der *Process ID* (*Process identifier*, deutsch: Prozesskennung) geht es in der Informatik darum, dass unterschiedliche Prozesse eindeutig identifiziert werden können. Die Identifikation erfolgt über einen einzigartigen Schlüssel.[24]

[16] Threads sind in einem Prozess eigenständige Aktivitäten, welche unabhängig von verschiedenen Prozessteilen ausgeführt werden. Threads sind eine Art Verarbeitungsstrang, die wie jeder andere Betriebssystemprozess, einen eigenen Prozesskontext besitzt. Verschiedene Threads können hoch oder nieder Prior sein. Threads mit niedriger Priorität können von Threads mit höherer Priorität ausgelöst werden. Bei der Ausführung von mehreren Threads spricht man von Multithreading.[35]

Aufbau des Dateisystems:
Da das *Android*-System auf einem Linux-Kernel basiert, wird auch das Linux-Dateisystem verwendet. Es kann durch eine Speicherkarte erweitert werden, deren Speicherplatz nicht an das Gerät gebunden ist. Um das externe Dateisystem nutzen zu können, muss dem System mit Hilfe des Linux-Befehls *mount* bekannt gemacht werden. Ressourcen und private Anwendungsdateien können nicht von einer SD-Karte aus genutzt werden, da eine Anwendung sie stets in ihrem privaten Dateisystem, auf dem installierten Speichermedium des Geräts, benötigt. Durch die Installation einer Anwendung auf dem *Android*-Gerät, wird automatisch ein *privates Anwendungsverzeichnis* erstellt. In diesem Verzeichnis darf nur die Anwendung selbst Schreiboperationen ausführen. Eine von einer Anwendung erzeugte Datei oder ein Verzeichnis erhält standardmäßig nur Lese- und Schreibrechte für die jeweilige Anwendung selbst. Innerhalb des Verzeichnisses werden z.B. alle Datenbanken der Anwendung abgelegt. Vom System wird schließlich für jede Anwendung ein Schlüssel vergeben, der diese eindeutig identifiziert. [5, S.155]

Die *Android*-Plattform ist in der Lage in die Ausführungsdauer von Prozessen einzugreifen und diese bei drohendem Ressourcenmangel zu beenden. Da das Betriebssystem Prozesse eigenständig beenden kann, unterliegen die Lebenszyklen der Komponenten ebenfalls der Kontrolle der *Android*-Plattform. Bedingt durch diese Eigenschaft, ist ein Datenverlust möglich, sobald ein Prozess durch die *Android*-Plattform beendet wird. Das Betriebssystem vergibt an die laufenden Prozesse sogenannte Prioritäten[17], welche anhand des Zustands der im Prozess laufenden Komponenten vergeben werden. Bei großer Ressourcenauslastung und den dadurch entstehenden Ressourcenmangel werden Prozesse mit der geringsten Priorität beendet. Prozesse, die für die Nutzer etwas visualisieren, bekommen durch das System eine hohe Priorität zugeteilt. Anwendungen, die länger nicht verwendet wurden, haben eine niedrige Priorität und werden im Notfall als erstes vom Betriebssystem beendet.[5, S.219]

Das *Android*-Betriebssystem unterscheidet zwischen unterschiedlichen Zuständen von Prozessen. Diese Zustände, werden im nachfolgenden erläutert, beginnend mit der niedrigsten Priorität.

Leere Prozesse:
Diese Prozesse werden von der *Android*-Plattform im Hintergrund weiter ausgeführt, auch wenn alle Komponenten bereits beendet wurden. Damit die Komponenten schnell gestartet werden können, werden diese im Cache[18] gespeichert. Diese Prozesse werden bei knappen Ressourcen sofort beendet.

Hintergrundprozesse:
Prozesse, die zur Zeit nichts anzeigen oder die keine laufenden Services enthalten, nennt man Hintergrundprozesse. Die Komponenten des Prozesses sind nicht beendet aber inaktiv und für den Anwender nicht sichtbar. Es werden Prozesse, die die längste Ausführungsdauer aufweisen, zuerst beendet.

[17] Priorität heißt im Allgemeinen den Vorrang einer Sache. Es kann damit festgelegt werden, in welche Reihenfolge mehrere Aufgaben bearbeitet werden. Ein ähnlicher Begriff ist „Wichtigkeit“.
[18] Cache ist ein schneller Puffer-Speicher, der wiederholte Zugriffe auf ein langsames Hintergrundmedium oder aufwändige Neuberechnungen zu vermeiden hilft.[18]

Serviceprozesse:
Diese Prozesse werden vom System im Hintergrund ausgeführt und sind für die Nutzer nicht sichtbar. Dieses kann z.B. das Abspielen von Musik sein. Sie haben keine weitere Verbindung zu anderen Prozessen.

Sichtbare Prozesse:
Bei sichtbaren Prozessen handelt es sich um Prozesse, die etwas für den Nutzer anzeigen oder auf Eingaben reagieren. Es ist allerdings möglich, dass ein Prozess von einem anderen überlagert wird. Wenn z.B. Prozess A von Prozess B überlagert wird, dann gilt Prozess A als inaktiv, da dieser Prozess den Fokus verloren hat und nicht mehr auf Eingabe des Anwenders reagiert.[5, S.220]

4.2 Apple iOS

Das Betriebssystem *iOS* wurde von der Firma *Apple* für die eigenen Produkte *iPhone*, *iPod* und *iPad* entwickelt. Das System wird nur auf der eigenen Hardware von *Apple* eingesetzt und ist nicht für andere Hardware-Hersteller lizenziert. Das *iOS*-Betriebssystem basiert auf einem *Mac OS X*[19] - Kern bzw. Darwin-Betriebssystem[20], welches auf einem Unix-Kern aufgebaut ist.
Für den Download von Anwendungen stellt *Apple* den *App Store* zur Verfügung. Eine Installation von Apps, die von Drittanbietern angeboten werden, wird von *Apple* nicht gestattet. Der Speicher eines *iOS*-Systems kann nicht durch eine Speicherkarte erweitert werden, da der Zugriff auf den Speicher von *Apple* stark begrenzt wird. Daraus resultierend haben Nutzer keinen kompletten Zugriff auf das Dateisystem. Um Daten mit dem PC auszutauschen, wird das Programm *iTunes* benötigt. Bis zur *iOS*-Version 4 war *iTunes* darüber hinaus zur Aktivierung eines Geräts notwendig. Ab *iOS* Version 5 ist es möglich ein *Apple*-Geräte auch ohne PC und *iTunes* in Betrieb zu nehmen. [76]

4.2.1 Aufbau und Funktionsweise von Apple iOS

Das *Apple iOS* Betriebssystem ist eine Weiterentwicklung des *Mac OS X*. Die Systemarchitektur besteht aus vier Schichten, die in Abbildung 6 dargestellt sind.

Architektur:

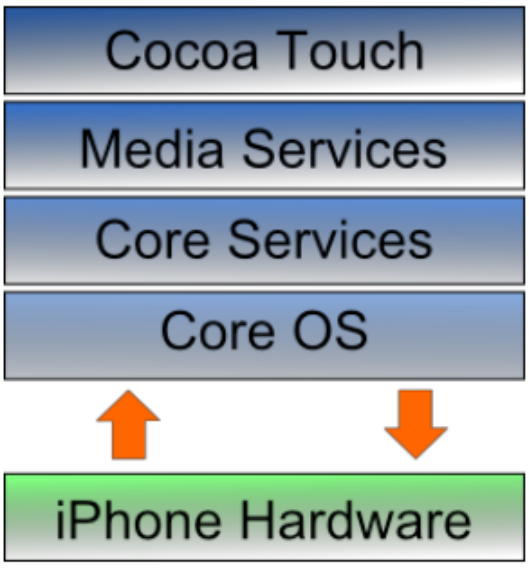

Abb. 6: Struktur des *Apple iOS* [83]

[19] *Mac OS X* ist ein Betriebssystem von der Firma *Apple*, welches für die hauseigenen *Macintosh*-Computer entwickelt wurde und erstmals im Jahr 2001 erschien. Es basiert auf einem unixoiden Betriebssystem und kommt in abgewandelter Form als *iOS* für die Mobilgeräte der Firma zum Einsatz.[4]
[20] Darwin ist ein freies unixoides Betriebssystem des Unternehmen *Apple* und bildet die Grundlage für *Mac OS X*.[81]

Jede der dargestellten Software-Schichten stellt Schnittstellen und sogenannte Frameworks[21] für die Entwicklung von Anwendungen zur Verfügung. Die Systemstruktur und die einzelnen Schichten werden im folgenden Abschnitt ausführlich erläutert.

- **Core OS:** Der Core OS stellt den Kern des Systems dar. Diese Schicht ist funktional direkt über der Hardware angesiedelt und stellt verschiedene Dienste, wie z.B. Netzwerkfunktionalität (CFNetwork Framework), Zugriff auf externes Zubehör (External Accessory Framework) und andere Dienste zur Verfügung. Weiterhin werden Betriebssystemfunktionalitäten wie Sicherheitsfunktionen (Security Framework mit öffentlich/privaten Schüsseln, Verschlüsselungsmethoden und Sicherheitsrichtlinien) bereit gestellt. Darüber hinaus werden Verwaltungsprozesse, wie Dateisystemverwaltung, Speicherverwaltung, Threads usw. geregelt.

- **Core Service:** Der Core Service dient zur Bereitstellung von grundlegenden Systemeigenschaften. Dazu gehören z.B. Datenbankfunktionen (SQLite library), der Zugriff auf den *App Store* und die Datenverwaltung (Core Data Framework).

- **Media Services:** Hierbei handelt es sich um ein High-Level-Framework, das zur Bereitstellung von Audiofunktionalität(AV Foundation Framework), Videofunktionalität (Media Player Framework) und Grafikfunktionalität (OpenGL ES Framework) zuständig ist. Ebenfalls werden zweidimensionale Elemente (Core Graphics Framework) und Animationen (Quartz Core Framework) dargestellt.

- **Cocoa Touch:** Diese Schicht beschreibt die oberste Softwareschicht und beinhaltet die meisten von Anwendungsentwicklern genutzten Frameworks. Cocoa Touch basiert auf der Cocoa API[22] von Mac OS X und ist zum größten Teil in Objective-C geschrieben. Diese wurde auf die Anforderungen des *iPhones* bzw. *iPads* angepasst und enthält zudem das *UIKit-Framework*. Dieses Framework dient der Entwicklung von schnellen und komplexen Web-Interfaces. Dem Entwickler wird dadurch eine umfassende Sammlung von einfach einsetzbaren und erweiterbaren *HTML-, CSS-* und *JavaScript*-Komponenten geboten. Die wichtigsten Funktionen sind das Erstellen und Verwalten von grafischen Oberflächen (Textfelder, Buttons, Farben, Schrift usw.), die Behandlung von Ereignissen und Progammzyklen (z.B. Interaktion mit dem Touchscreen), die Datenverwaltung, die Kommunikation zwischen unterschiedlichen Prozessen (IPC), sowie die Nutzung der Daten von Hardwarekomponenten (z.B. Batterie, Gyroskop, Kamera usw.) [86, S.2], [83]

Die Ausführung von Applikationen findet in einer sogenannten Sandbox statt. Anwendungen sind auf dem *iOS*-Betriebssystem nur dann ausführbar, wenn diese mit einem Apple Zertifikat signiert sind. Durch dieses Zertifikat wird vom Hersteller sicher gestellt, dass eine Anwendung nicht manipuliert wurde. Bei jeder ausgeführten Anwendung wird zunächst das Zertifikat entnommen und mit dem Zertifikat des letzten Aufrufs dieser Anwendung

[21] Der Softwarehersteller Ashton Rate prägte 1984 den Begriff des Frameworks, als dieser ein entsprechendes Software-Paket vorstellte. Mit einem Framework ist ein Programmiergerüst gemeint, welches als Rahmen für die Entwicklung von Software benutzt wird. Es werden den Entwicklern bestimmte Bausteine zur Designstruktur zur Verfügung gestellt. [43]

[22] Das *Application Programming Interface* ist eine Programmierschnittstelle, die die Anbindung an das System zur Verfügung stellt. Bei dieser Programmierschnittstelle ist nur die Anbindung auf Quelltext-Ebene definiert. Mit der Schnittstelle ist der Zugriff auf Datenbanken, Festplatte, Grafikkarte usw. möglich. Durch diese Schnittstelle ist es z.B. bei Windows erst möglich, dass Fremdfirmen ihre Programme auf dem System installieren können.[27]

verglichen. Wird bei dieser Überprüfung ein Unterschied festgestellt, wird davon ausgegangen, dass die Anwendung verändert bzw. manipuliert wurde. Sie wird folglich vom System als nicht vertrauenswürdig eingestuft. [86, S.3] Durch das Ausführen von Apps in einer Sandbox läuft jede Anwendung isoliert von anderen ab und ist infolgedessen vom System Kernel getrennt. Anwendungen von Drittanbietern besitzen keine Möglichkeit diese Isolierung zu umgehen und verfügen über die selben limitierten Rechte. Durch diesen Mechanismus sind sie durch das *iOS* und den Benutzer kontrollierbar. Anwendungen haben selbst keinen Einfluss darauf, welchen Systemstatus sie vom Betriebssystem zugewiesen bekommen.[86, S.4]

iOS erlaubt jeder Anwendung auf folgende Ressourcen uneingeschränkt zuzugreifen:

- **Adressbuch und Kalendereinträge**
- **ID des Geräts**
- **Telefonnummern**
- **Musik, Fotos und Videos**
- **Browserverlauf**
- **Kommunikation zu jedem Computer im Wireless Internet und Verbindungs Logs** [86, S.4]

Bevor die Entwickler eine *iOS*-Anwendung im *App Store* veröffentlichen können, wird sie zuvor von *Apple* einer Prüfung unterzogen. Die Entwickler müssen jede ihrer Anwendungen mit einem *Apple* Zertifikat signieren. Durch Anwendungen dürfen z.B. keine Daten gespeichert und gesendet werden, die auf die Tätigkeiten der Benutzer an bestimmten Orten hinweisen. Es darf weiterhin keine eigene Werbung geschaltet werden und die App darf weder Schaden anrichten noch Pornografie beinhalten.
Besteht die Anwendung die Prüfung und besitzt ein Zertifikat, kann sie im *App Store* veröffentlicht werden.[86, S.4]

Jailbreak:
Wenn ein sogenannter Jailbreak auf einem *iOS*-Gerät installiert ist, wird das Signieren von Anwendungen ausgeschaltet. Somit besteht die Möglichkeit, auch Anwendungen von Anbietern außerhalb des *App Stores* zu installieren. Durch einen Jailbreak wird das Betriebssystem auf dem Mobilgerät dahingehend modifiziert, dass die Nutzer Root-Rechte auf interne Funktionen und darüber hinaus Zugriff auf das Dateisystem erhalten. Da *iOS* auf einem Darwin-Betriebssystem basiert, erhalten die Nutzer nach einem Jailbreak Administratorrechte auf ein vollwertiges Unix-System. Ein Jailbreak ist nicht illegal und bietet den Nutzern die Möglichkeit das Gerät nach eigenen Wünschen anzupassen. Es ist jedoch festzuhalten, dass dadurch das grundlegende Sicherheitsprinzip von *iOS*-Systems umgangen wird.[86, S.5], [10]

4.3 Blackberry OS

Blackberry OS ist ein kostenlos nutzbares Multitasking-Betriebssystem, dass von dem kanadischen Anbieter *Blackberry* (vormals *Research in Motion, RIM*) für dessen Geräte der Marke *Blackberry* entwickelt wurde. Das System bietet spezielle Synchronisations- und E-Mail-Funktionen, die eine entsprechende Netzwerk-Infrastruktur voraussetzen. Hierfür war lange Zeit ein spezieller Handytarif notwendig. Der Nachfolger von *Blackberry OS* ist *Blackberry 10*. Seit der *Blackberry OS* Version 10 wird kein spezieller Tarif mehr benötigt. Die Nutzer können Dienste von *Blackberry*, sowie uneingeschränkt auch andere E-Mail-,Kalender- und Adressbuch-Dienste nutzen. Zur direkten Synchronisation des Mobilgeräts wird von der Firma *Blackberry* ein Programm bereit gestellt. Für Installationen von Apps bietet *Blackberry* den Online-Shop *Blackberry App World* an. Es können jedoch auch Apps aus anderen Quellen außerhalb des Stores installiert werden. [64] [76]

4.3.1 Aufbau und Funktionsweise von Blackberry OS

Bei *Blackberry OS* handelt es sich um ein Multitasking-Betriebssystem, dass in C++ programmiert ist. Anwendungs- und Dienstprogramme können über eine spezielle Schnittstelle in eine Java-Umgebung eingebunden werden.[64]

Funktionsweise *Blackberry OS 5*
Applikationen für *Blackberry* werden mit der *Java Micro Edition*[23] *(Java ME)* entwickelt. In Java ME gibt es ein spezielles Profil, dass auf die Fähigkeiten von Mobiltelefonen und Mobilgeräten ausgelegt ist. Dieses Profil wird *MIDP (Mobile Information Device Profile)* genannt. *MIDP* unterstützt die Funktionen und Abfrage von Tastatur, Bildschirm sowie flüchtigem und nicht-flüchtigem Speicher im KByte-Bereich. Bei *MIDP* handelt es sich um ein Sandboxmodel, dass einen soliden Schutz gegenüber Schadsoftware bietet, da auf das Dateisystem außerhalb der Sandbox nur sehr eingeschränkt zugegriffen werden kann. Bei den MIDP-Applikationen auch *MIDlets* genannt, handelt es sich um Software für Mobilgeräte, die dem speziellen Profil des *MIDP* entspricht.[47, S.1] Es können jeweils mehrere *MIDlets* zu einer sogenannten *MIDlet-Suite* zusammen gefasst werden. Durch das Zusammenfassen, entsteht der Vorteil, dass sich die einzelnen *MIDlets* Programmcode und Ressourcen teilen können. *MIDlets* können auf den *Blackberry*-Geräten ausgeführt werden, können aber nur auf eine geringe Anzahl der APIs von Blackberry zugreifen. Damit der volle Umfang der APIs von *Blackberry* benutzt werde kann, hat *Blackberry* einen eigenen Typ entwickelt, dieser wird *RIMlet* genannt. Durch das Verwenden von *RIMlets* wird die Funktionalität des *Blackberry*-Mobilgeräts erhöht.[54]

[23] *Java ME* wurde früher auch als *Java Platform 2, Micro Edition* oder kurz *J2ME* bezeichnet, ist die Umsetzung der Programmiersprache Java für Mobilgeräte. Durch *J2ME* ist es möglich, dass Anwendungen auf den mobilen Geräten ausgeführt werden können, unabhängig vom Hersteller des Geräts, des Modells oder dem Betriebssystem. Dies ist möglich, da diese Anwendung von der Java Runtime unabhängig ausgeführt wird.[40]

Für die im vorangegangenen Abschnitt erläuterten Applikationen gibt es verschiedene Möglichkeiten, wie diese von Entwicklern zur Verfügung gestellt werden:

- **MIDlets:** Hierbei handelt es sich um JAR-Dateien, die von der Java Applikationen verwendet werden. Prinzipiell ist eine JAR-Datei eine ZIP-Datei, die die vom Java Compiler erstellten Klassen beinhalten. Damit das extrahieren des erzeugten Quellcodes erschwert wird, wird in der Regel Code-Obfuscation[24] eingesetzt, wodurch der Code schwer lesbar ist. Das Signieren eines *MIDlets* ist nicht möglich. Daraus folgt, dass nur auf bestimmte *Blackberry* APIs zugegriffen werden kann.
- **RIMlets:** Bei *RIMlets* wird ein spezielles Dateiformat (COD) von *RIM* verwendet, welches das Signieren der Applikationen und somit den Zugriff auf alle APIs zulässt. Im Gegensatz zu den *MIDlets* kann der Code nicht mehr ausgelesen werden.
- **MIDlets -> RIMlets:** Es besteht die Möglichkeit *MIDlets* in *RIMlets* zu konvertieren, wodurch diese eine Signatur enthalten.[77, S.16]

Sicherheitslücke Bufferoverflow
Sogenannten Bufferoverflows[25] stellen eine ernstzunehmende Sicherheitslücke dar und waren die Ursache für einen Großteil der erfolgten Angriffe in den letzten Jahren. Es existieren verschiedene Maßnahmen, die verhindern, dass diese Lücke ausgenutzt wird, es sind jedoch aufgrund des proprietären[26] Betriebssystems nur wenige Details über das Ausnutzen der Sicherheitslücke bekannt.[77, S.16] Dadurch, dass *MIDlets* und *RIMlets* auf dem *Blackberry OS* in einer *Java VM* laufen müssen, entfallen die typischerweise durch unorganisierten geschriebenen C-Code bekannten Bufferoverflows.[77, S.17]

Verschlüsselung der Daten
Blackberry-Geräte unterstützen die Verschlüsselung der Daten auf dem Gerät. Bei der Datenverschlüsselung spielen Verschlüsselungseinstellungen und Passwortrichtlinien eine entscheidende Rolle. Der Schlüssel für die Datenverschlüsselung wird mit Hilfe der verwendeten Passwörter erzeugt. Beim internen Speicher wird ein symmetrischer Advanced Encryption Standard(AES)[27]-Schlüssel verwendet, der den Flash-Speicher des Geräts verschlüsselt, sobald das Gerät vom Nutzer gesperrt wird. Der Schlüssel dafür wird vom

[24] Der Begriff *obfuscate* heißt übersetzt „verdunkeln“, „verwirren“. Mit *Code-Obfuscation* ist somit das Verschleiern des Quellcodes gemeint. Dadurch sollen, in erster Linie, Programme vor Reverse Engineering geschützt werden. Bei Reverse Engineering wird versucht durch den Quellcode eines Programms das Programm nachzubauen.[21]

[25] Ein Bufferoverflow [deutsch: Pufferüberläufe] tritt auf, wenn in einen Puffer, welcher sich auf einen Abschnitt im Speicher bezieht, mehr Informationen geschrieben werden, als dieser umfasst. Dadurch besteht die Möglichkeit, dass Informationen außerhalb der Sektion geschrieben werden, wodurch Daten beschädigt werden, ein laufendes Programm nicht weiter ausführbar ist oder Schadsoftware installiert werden kann.[44]

[26] Als proprietäre Software wird Software bezeichnet, die auf herstellerspezifischen Standards basiert.(Umgangssprachlich auch „unfreie Software“)

[27] Der *Advanced Encryption Standard* ist ein Verschlüsselungs-Algorithmus, welcher im Jahr 2001 als neuer Standard AES bekannt wurde. Die Entwickler des Algorithmus waren Joan Daemen und Vincent Rijmen, daher wird *AES* auch Rijndael-Algorithmus genannt. Die Verschlüsselung wird auch Blockverschlüsselung genannt, da eine Reihe von Byteersetzungen (Substitutionen), Verwürfelung (Permutationen) und linearen Transformationen auf Datenblöcke von 16 Byte ausgeführt werden. Die unterschiedlichen Bezeichnungen AES-128, AES-192 und AES-256 geben an, ob der Schlüssel eine Länge von 128-, 192- oder 256-Bit hat. Bei AES handelt es sich um einen der am meist verwendeten und sichersten Verschlüsselungsverfahren.[7]

Passwort des Benutzers generiert, welchen er zum Entsperren des Geräts benötigt. Das Verschlüsseln der Daten ist ebenfalls für externen Speicher möglich.[77, S.18]

Innerhalb des Betriebssystems kann festgelegt werden, dass der Speicher nach einer bestimmten Anzahl an falschen Eingaben des Passworts gelöscht wird. Zudem kann dieser mit Hilfe einer Fernüberwachung gelöscht werden. Das Markenzeichen von *RIM* ist die Verschlüsselung für einen sicheren Datenverkehr innerhalb der *Blackberry Enterprise Solution*. Nach korrekter Implementierung des Sicherheitskonzepts hat weder *RIM* noch Dritte Zugriff auf die Daten.[87]

Das Sicherheitskonzept
Die nachfolgende Abbildung 7 zeigt den grundsätzlichen Aufbau der *Blackberry*-Sicherheitsarchitektur:

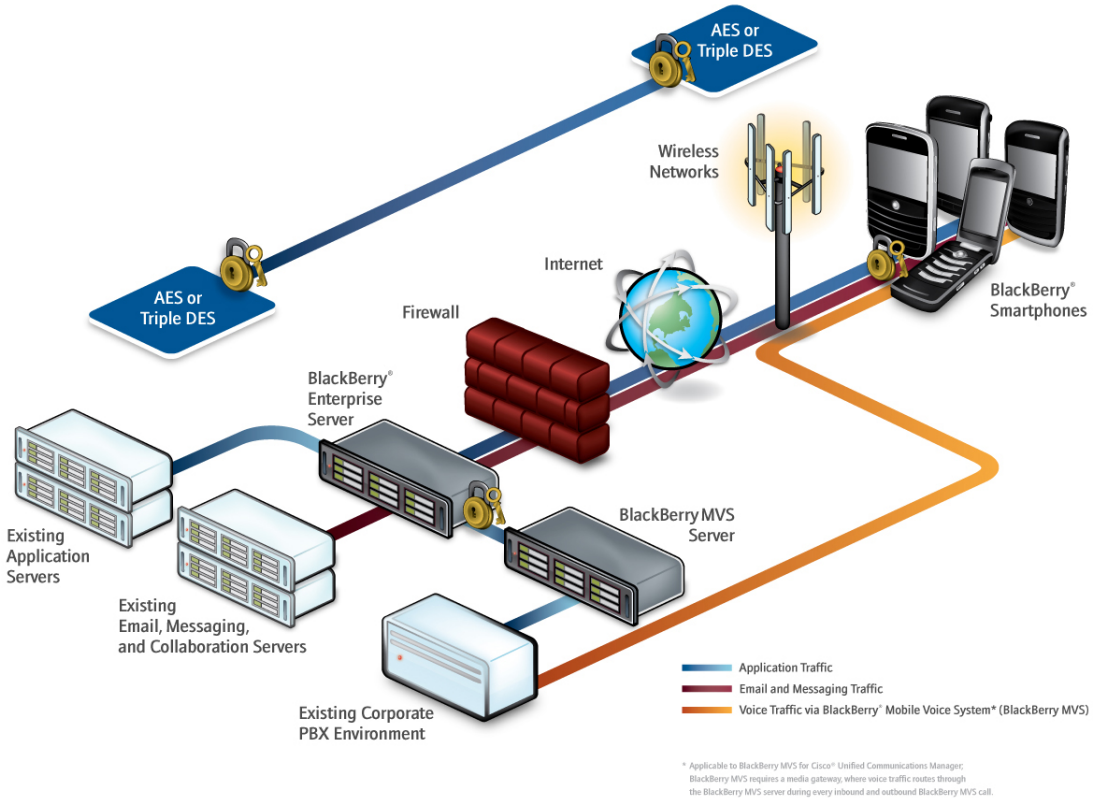

Abb. 7: Sicherheitsarchitektur von *Blackberry* [87]

Aufgrund der Tatsache, dass ein mobiler Datenverkehr stets Sicherheitsrisiken mit sich bringt und somit die Gefahr besteht, dass das Gerät ungewollte Zugriffe zulässt, bietet *RIM* eine sichere Komplettlösung, die das bestehende System bereits voll integrieren. Mit diesem System wird auf dem gesamten Übertragungsweg (vom Unternehmensserver bis zum Endnutzergerät) ein nahtloser Schutz von Daten erreicht. Daten, die zwischen E-Mail-Server des Unternehmens und dem mobilen Endnutzergerät übertragen werden, sind standardmäßig mit dem Advanced Encrytion Standard (AES) 256 verschlüsselt. Die

meisten Lösungen zum Sichern von Daten setzen auf einen VPN[28]-Tunnel. Bei *Blackberry* setzt die Systemarchitektur auf eine Chiffrierung jedes einzelnen Nutzdatenpakets mit einem individuellen 256-Bit-Schlüssel, der für jedes Endnutzergerät existiert und zyklisch erneuert wird. Ein Aussetzen der Datenverbindung oder ein Verbindungsabbruch ist für den Erhalt der Daten irrelevant, da die Verschlüsselung unabhängig vom Transportprotokoll stattfindet. Für das Verschlüsselungsverfahren wird jedem Endnutzergerät ein Schlüssel zugewiesen, der auf einem privaten symmetrischen Schlüssel basiert. Es existieren keine Master-Schlüssel, die Administratoren oder Dritten Zugriff von außen auf das Endgerät geben können. Der Schlüssel bleibt immer direkt beim Kunden, d.h. auf dem jeweiligen *Blackberry*-Gerät des Nutzers. Nur die IT-Abteilung des Nutzers kann auf die Konfiguration der einzelnen Anwender zugreifen, beispielsweise, um diese zu löschen oder zu ändern, wenn das Endnutzergerät verloren gegangen ist. Sofern die kryptographischen Algorithmen korrekt implementiert worden sind, gilt diese Sicherheitslösung als absolut sicher, da der Schlüssel zufällig erzeugt wird und nicht kompromittierbar ist.[87]

Network Operating Center
Für den gesamten Datenverkehr unterhält *RIM* zwei *Network Operating Center* (NOC) in Kanada und England. Da nicht unendlich viele IP-Adressen existieren, wurden von der *Internet Assigned Numbers Authority* (kurz IANA) drei private Adressbereiche festgelegt. Diese IP-Adressen sind im Internet nicht sichtbar und können daher auch nicht geroutet[29] werden. Durch die nicht direkte Erreichbarkeit dieser privaten Netze aus dem Internet, wird implizit ein Schutz vor Fremdzugriffen erreicht. Dieser IP-Adressbereich wird auch in GPRS-Netzen privater Mobilfunkbetreiber verwendet. Da diese Netze vom öffentlichen Internet getrennt sind, wurden von den Netzbetreibern so genannte Proxy/Gateway-Server eingerichtet. Mit Hilfe der Servern werden die privaten Adressen auf öffentliche Internetadressen übersetzt und folglich ist das Endgerät im Internet sichtbar. Das *NOC* dient bei *Blackberry* als eine Art Riesen-Router bzw. Gateway. Zusammengefasst ist ein *NOC* die Schnittstelle zwischen *Blackberry* und Mobilfunknetz. [87]

Aktuelles System *Blackberry 10*
Das aktuelle Betriebssystem *Blackberry 10* basiert auf einer QNX-Neutrino-Architektur. Bei dieser Architektur liegt die Idee zugrunde, dass Großteile des Systems in Form von Prozessen ausgeführt werden. Jeder Prozess läuft in einem separatem Speicherbereich, der von der *Memory Management Unit* (MMU)[30] verwaltet wird. Es spielt dabei keine Rolle ob es sich bei dem Prozess um eine Applikation oder um einen Treiber handelt. Versucht Prozess A versehentlich Daten oder Code von Prozess B zu überschreiben, so wird der QNX-Microkernel von der MMU darüber informiert und beendet Prozess A oder startet diesen neu während Prozess B davon unberührt bleibt und weiter ausgeführt wird.[57] Bei QNX für Mobilgeräte wird der Kernel als Microkernel implementiert. Die Hauptmerkmale des Microkernels sind seine geringe Größe und sein sehr kleiner Aufgabenbereich. Der

[28] Das *Virtual Private Network*, kurz VPN, diente zunächst dazu, einen Teilnehmer eines geschlossenen privat Netzwerks mit einem anderen geschlossenen privat Netzwerk zu verbinden. Dadurch kann beispielsweise ein Mitarbeiter von zuhause aus auf das Firmennetzwerk zugreifen, als wenn er vor Ort wäre.[19]

[29] Unter Routing versteht man die Wegfindung eines Datenpakets in einem Netzwerk.

[30] Bei der *Memory Management Unit* (*MMU*) handelt es sich um eine Hardwarekomponenten eines Computers, die die Speicherverwaltung und den Zugriff auf den Arbeitsspeicher steuert. Ihre Aufgaben sind das Umrechnen von virtuellen in physikalische Adressen. Durch die *MMU* hat das Betriebssystem vollen Zugriff auf den virtuellen Adressraum.[33]

Großteil der Funktionen ist in unterschiedliche Module ausgelagert. Innerhalb von QNX wird vom sogenannten Message Passing[31] exzessiv Gebrauch gemacht, wodurch zwischen Prozessen bzw. Threads auf vielfache Weise kommuniziert werden kann. Die benötigten, aber nicht im Microkernel enthaltenen Funktionalitäten, müssen mit Hilfe von Modulen realisiert werden. So muss z.B. für eine Netzwerkverbindung ein Netzwerkmodul vorhanden sein und geladen werden. Soll auf das Dateisystem zugegriffen werden, muss das entsprechende Modul geladen werden. Die Module, welche unterschiedliche Funktionen bereit stellen, können zu einer späteren Laufzeit des Systems geladen oder beendet werden. Bei diesen Modulen handelt es sich um reguläre Programme, welche ihren eigenen Adressraum haben und in der Lage sind, Threads zu verwalten. Tritt ein Fehler in einem Modul auf, ist lediglich der Prozess, nicht aber der Kernel davon betroffen. Der Microkernel bearbeitet vor allem das Message Passing, die Verarbeitung von Signalen und das Thread Scheduling. Andere Funktionen wie Shared Memory, Process creation and destruction oder Pipes, werden durch Module verwaltet.

[31] Bei Message Passing handelt es sich um die Kommunikation zwischen Prozessen und Threads. Diese beruht auf dem Versenden von Nachrichten(Funktionsaufruf, Signale,Datenpakte) zu den Empfängern.

4.4 Symbian

Symbian ist ein Betriebssystem, dass einerseits als offen und vielseitig, aber andererseits als kompliziert gilt. Inzwischen wird es nur noch von dem finnischen Telekommunikationskonzern *Nokia* eingesetzt. Um Apps auf dem System zu installieren, kann entweder *Nokias Ovi Store* oder alternativ eine andere Quelle, wie z.B. *Get Jar Apps* verwendet werden. Für Synchronisationsvorgänge von Kalender und Adressen setzt das System ab Version *Symbianˆ3* hauptsächlich Internetdienste ein. Mit der PC Software *Ovi Suite* kann z.B. Navigations-Kartenmaterial auf das Mobilgerät übertragen werden. [76]

4.4.1 Aufbau und Funktionsweise von Symbian

Bei der *Symbian*-Software handelt es sich um eine Multi-Threading-Architektur. Zu den Softwarekomponenten, auf denen das *Symbian*-OS basiert, gehören Abgrenzung durch Privilegien und Prozesse, sowie die Abgrenzung zwischen Anwendungen und DLLs (Dynamic Link Libraries)[32]. Bei dem Betriebssystem dienen die Grenzen der einzelnen Softwarekomponenten als Schutz vor unerlaubtem Zugriff einer Komponente auf Speicherbereiche einer anderen Komponente oder auf Hardware.[56, S.29]

Abbildung 8 gibt eine Übersicht über die unterschiedlichen Systemkomponenten und ihre jeweiligen Abgrenzungen zueinander.

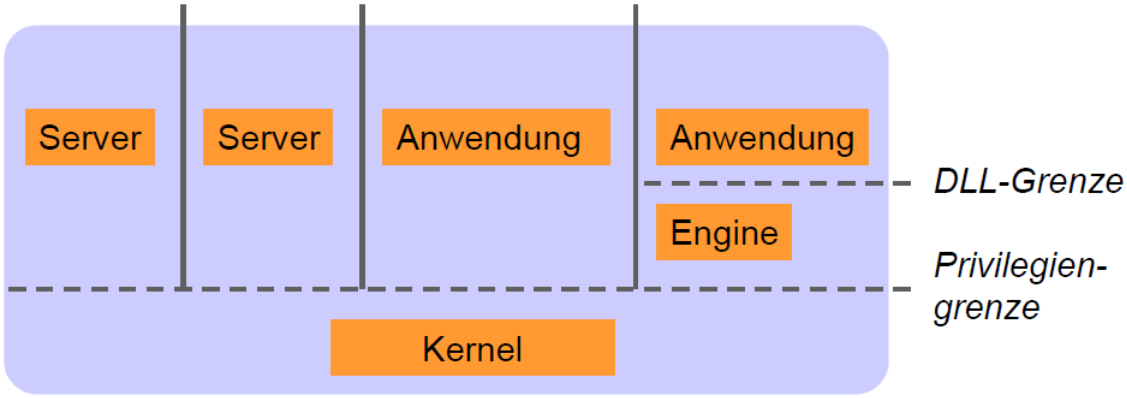

Abb. 8: Übersicht der Abgrenzung [56, S.30]

Die grundsätzlichen Komponenten setzen sich aus Kernel des Betriebssystems, Server und Anwendungen sowie der Engine zusammen. Die Grenze zwischen den Komponenten und dem Kernel wird Privilegiengrenze genannt, wodurch ein Programm, welches ohne Privilegienrechte ausgeführt wird, nur über den Kernel (mit Hilfe einer API) auf Hardwareressourcen zugreifen kann. Diese verschiedenen Komponenten werden nachfolgend detaillierter erläutert.

[32] Eine *Dynamic Link Library* bezeichnet eine dynamische Programmbibliothek. Der Zweck einer DLL ist es, den Speicherplatz auf der Festplatte zu reduzieren. In einer DLL kann Programmcode stehe, der von mehreren Anwendungen benötigt wird. Diese DLL wird nur ein mal auf der Festplatte benötigt und wird nur ein mal in den Hauptspeicher geladen, wenn mehrere Programme auf die Bibliotheken zugreifen.[59]

Kernel:
Der Kernel übernimmt die Verwaltung der Hardwareressourcen wie z.B. Speicher oder Ein- und Ausgabe. Darüber hinaus regelt und kontrolliert er den Zugriff von Softwarekomponenten auf die Hardwareressourcen. Wenn eine Softwarekomponente das Zugriffsrecht auf eine bestimmte Hardwareressource hat, gibt der Kernel die entsprechende Hardware frei. Ist keine entsprechende Berechtigung erteilt, ist der Kernel in der Lage den Zugriff zu verweigern. Der Kernel verwendet durch die Hardware unterstütze Privilegien, um auf Hardwareressourcen zuzugreifen. Die CPU führt folglich bestimmte privilegierte Instruktionen nur für den Kernel aus (im sogenannten Kernel-Mode). User-Mode-Programme werden ohne Privilegienrechte ausgeführt und können nur über den Kernel (mit Hilfe einer Kernel API) auf die Hardwareressourcen zugreifen. Die Privilegiengrenze ist die Grenze zwischen dem Kernel und allen anderen Komponenten. [56, S.30]

Anwendung:
Bei einer Anwendung handelt es sich um ein Programm, dass eine Benutzerschnittstelle aufweist. Bei *Symbian*-OS wird jede Anwendung innerhalb eines eigenen Prozesses gestartet. Jeder dieser Prozesse hat einen eigenen virtuellen Adressraum. Dadurch, dass die Adressräume der einzelnen Anwendungen voneinander getrennt sind, ist es nicht möglich, dass eine Anwendung aus Versehen oder absichtlich Daten einer anderen Anwendung überschreibt. Die Grenzen zwischen zwei Anwendungen wird Prozessgrenze genannt. [56, S.30]

Server:
Bei einem Server handelt es sich um ein Programm ohne Benutzerschnittstellen, der für die Verwaltung einer oder mehrerer Ressourcen zuständig ist. Mit Hilfe einer API wird den Clients der Zugriff auf die Dienste des Servers ermöglicht. Ein Client kann z.B. eine Anwendung oder ein anderer Server sein. Wie auch die Anwendungen, laufen die Server in der Regel in einem eigenen Prozess. Folglich besteht zwischen Server und Client eine Prozessgrenze. Um Dienste zur Verfügung zu stellen, werden bei *Symbian* sehr häufig Server eingesetzt. [56, S.30]

Engine:
Bei einer Engine handelt es sich um einen Teil einer Anwendung, der für die Programmlogik und die Speicherverwaltung zuständig ist. Anwendungen können in einen Bereich für die Benutzerschnittstelle (GUI) und einen Bereich für die Engine eingeteilt werden, was für komplexere Anwendungen geeignet ist, wodurch eine gute Übersicht geschaffen wird. Eine Anwendungs-Engine kann ein getrenntes Quellcode-Modul sein, eine eigene DLL besitzen oder sie kann mehrere DLLs umfassen. Die Grenze zwischen Anwendung und dessen Engine wird Modulgrenze bzw. DLL-Grenze genannt. [56, S.31]

DLL-Grenzen bzw. Modulgrenzen dienen zur Unterstützung der Systemintegrität durch Modularisierung und Kapselung. Durch die Privilegiengrenzen wird die Systemintegrität durch Trennung des Kernels und der Hardware von User-Mode-Zugriffen unterstützt. Durch die Prozessgrenzen wird die Systemintegrität und Systemstabilität unterstützt, da Speicherbereiche der einzelnen Anwendungen getrennt werden. Die Prozesse des *Symbian*-OS leisten eine Art Schutzfunktion für Programme, da jeder Prozess in einem eigenen Speicherbereich läuft und somit von anderen Prozessen getrennt ist. Die virtuellen Adres-

sen, die ein Prozess verwendet, werden in physikalische Adressen im ROM und RAM übersetzt. Das Übersetzen der Adressen wird von der sogenannten Memory Management Unit (MMU) durchgeführt. Dadurch kann auf den Speicherbereich eines Prozesses nicht von einem anderen Prozess zugegriffen werden. Jeder Prozess kann einen oder mehrere Threads besitzen, die im gleichen Adressraum wie der Prozess selbst ausgeführt werden. Jeder Thread eines Prozesses wird unabhängig von anderen Threads ausgeführt. Da die Threads alle im gleichen Adressbereich ausgeführt werden, können diese absichtlich oder aus Versehen die Daten eines anderen Threads ändern. Aufgrund dessen sind Threads im Vergleich zu Prozessen weniger gut voreinander geschützt. [56, S.31] In Abbildung 9 sind zwei Prozesse dargestellt, die einen bzw. zwei Threads gestartet haben, um zu verdeutlichen, dass Threads in einem Prozess parallel ausgeführt werden.

Um ein Multitasking zu erreichen, werden einzelne Threads vom Kernel präemptiv verwaltet, d.h. jedem Thread wird eine Priorität zugewiesen. Der Thread mit der höchsten Priorität wird vom Kernel ausgeführt. Ein Thread, der nicht für die Ausführung geeignet ist z.B., weil er gerade auf Daten eines I/O Gerätes oder eines anderen Ereignisses (Events) wartet, wird vom Kernel in einer Warteliste gehalten. Während der Zeit, in der ein Thread auf ein Ereignis wartet, könnten andere Threads ausgeführt werden. Sobald das Ereignis auftritt, wird der Thread aus der Warteliste entnommen und weiter ausgeführt. Nach jedem Hinzufügen oder Entnehmen eines Threads aus der Warteliste wird vom Kernel geprüft, welches der Thread mit der höchsten Priorität ist, welcher als nächstes ausgeführt werden soll. Das Unterbrechen eines Threads durch einen Thread mit höherer Priorität nennt man „Präemption“. Die Ausführungskontrolle der einzelnen Threads wird Kontextübergabe genannt. Es ist darauf zu achten, dass bei der Kontextübergabe zwischen zwei Threads möglichst wenig Performance eingebüßt wird. Einer Kontextübergabe zwischen zwei Threads aus unterschiedlichen Prozessen ist ebenfalls möglich. Für die Kontextübergabe aus unterschiedlichen Prozessen, wird im Vergleich zur Kontextübergabe in einem Prozess, mehr Performance benötigt. Der Grund dafür ist, dass für diesen Vorgang eine Änderung in der Memory Management Unit vorgenommen werden muss.

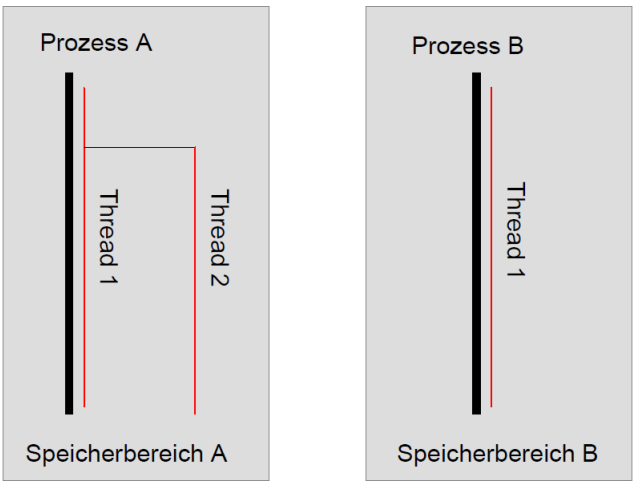

Abb. 9: Prozesse und Threads [56, S.32]

Die MMU passt die Abbildung der virtuellen Adressen eines neuen Prozesses auf die echten physikalischen Adressen im Speicher an. Eine Kontextübergabe von zwei Threads im selben Prozess ist somit ressourcenschonender. [56, S.32]

In Abbildung 10 wird gezeigt, wie die virtuellen Adressräume zweier Prozesse in den physikalischen Speicher durch die MMU abgebildet werden.

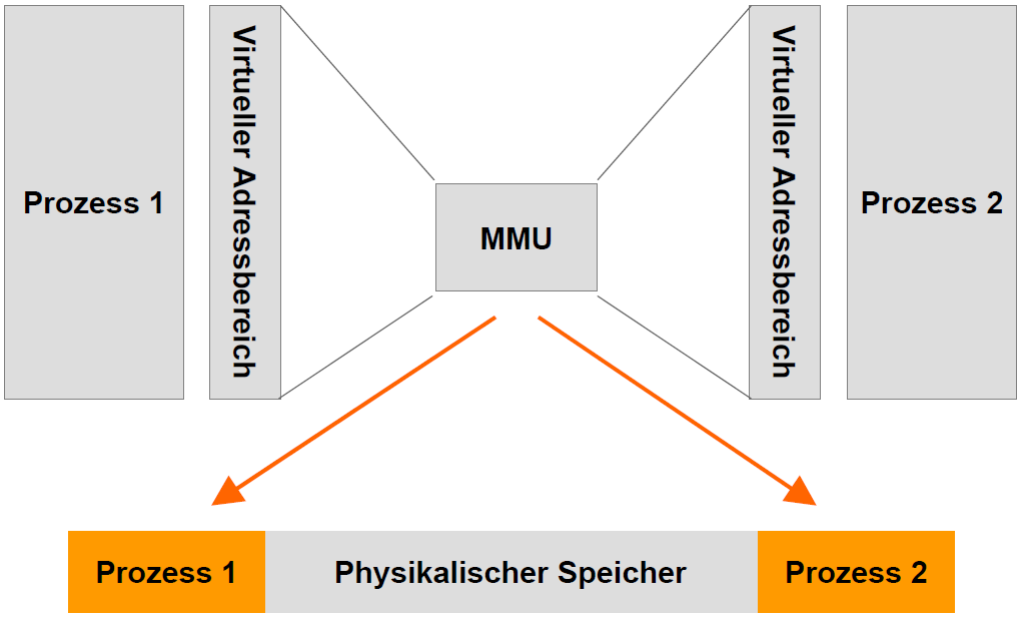

Abb. 10: Virtueller Adressbereich und MMU [56, S.33]

In der Regel verwendet jede Anwendung bei *Symbian* ihren eigenen Prozess und genau einen Thread. Die Server verwenden ausschließlich ihren Prozess und nur einen Thread. Es kann allerdings vorkommen, dass z.B. mehrere Server eng miteinander gekoppelt sind. Dieses wird so realisiert, dass mehrere Server in einem einzigen Prozess zusammengefügt werden, der mehrere Threads beinhaltet, um die Kontextübergabe gering zu halten. Die Integration von mehreren Servern in einen Prozess wird z.B. für die Kommunikation bei Serial-, Socket- oder Telefonserver verwendet.[56, S.33]

Das *Symbian*-OS verfügt über eine sogenannte Mikrokernel-Architektur. Durch diese Architektur kann ein speichersparender Kernel mit einem sogenannten „kleinen Footprint“ realisiert werden. Im Kernel sind nur die notwendigsten Systemfunktionen untergebracht, wie z.B. Memorymanagement (d.h. die Speicherallokierung für Prozesse im Kernel-Mode und User-Mode), Gerätetreiber und das Power-Management. Die Server-Komponenten befinden sich in der Middleware[33]. Die Server stellen dem Betriebssystem erweiterte Funktionen zur Verfügung, wie z.B. Kommunikation, grafische Benutzerführung (GUI) und den Fileserver. Das Betriebssystem ist durch diesen Aufbau robust, bootet schnell und hat sehr kurze Reaktionszeiten. Das Mikrokernel-Betriebssystem bietet durch seinen modularen Aufbau eine gute Erweiterbarkeit und Sicherheit, da nur ein kleiner Teil des Betriebssystems im privilegierten Modus ausgeführt wird. Das Auftreten von Fehlern oder Systemabstürzen kann dadurch auf ein Minimum reduziert werden. [56, S.36]

[33] Bei der Middleware handelt es sich um eine Schicht in einer komplexen Software-Struktur, welche die Aufgabe hat, Zugriffsmechanismen auf unterhalb angeordnete Schichten zu vereinfachen. Die Middleware stellt zudem Funktionen zur Verteilung sowie Dienste zur Unterstützung von Anwendungen bereit, womit Anwendungsprogramme entlastet werden.[38]

In Abb. 11 ist der Aufbau des Kernels und den verschiedenen Systemkomponenten dargestellt. Das *Symbian*-OS kann in verschiedene Bereiche aufgeteilt werden, die im Folgenden näher beschrieben werden.

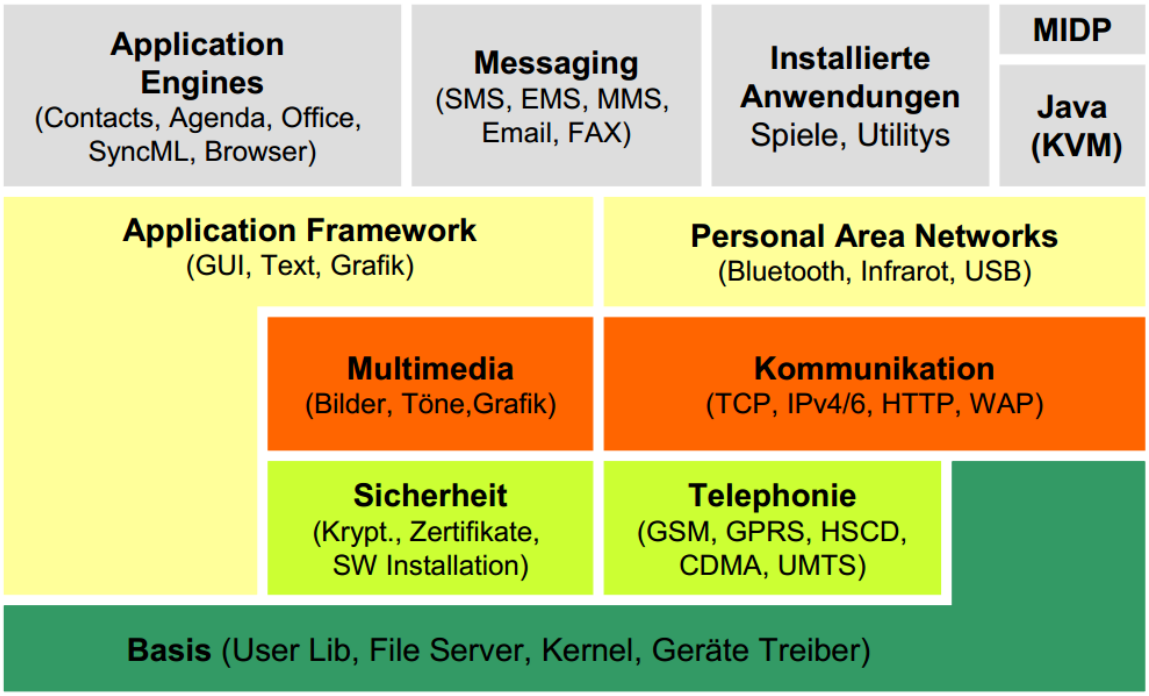

Abb. 11: Kernel und Systemkomponenten [56, S.39]

Basis:
Die Basis besteht aus dem Kernel, der Userlibrary und den Basisdiensten wie Fileserver und Sicherheit. Hier dargestellt unter dem Kernelmode, welcher Mikrokernel und Gerätetreiber beinhaltet.

Middleware:
In der Middleware werden zusätzliche Sicherheits- und Telephoniekomponenten bereitgestellt. Zudem enthält sie APIs für Daten-Management, Text, Zwischenablage, Grafik, Internationalisierung und die Core-GUI-Komponenten. Während durch die Multimediakomponenten Grafiken und Töne bearbeitet werden, wird durch die Kommunikationskomponenten der Zugriff auf verschiedene Protokolle und den Datenaustausch mit der Außenwelt ermöglicht.

Anwendungskomponenten:
Die Anwendungskomponenten befinden sich über der Middleware, die sogenannte Application-Engines und Application-Services, wie die Kontakte- und Terminverwaltung oder die Anwendungsinstallation, beinhaltet. Zudem bietet *Symbian* eine systeminterne Unterstützung für Java-Anwendungen. [56, S.37]

Die Verwaltung des Systemspeichers wird, wie bereits auf Seite 30 und 31 beschrieben, von der Memory Management Unit durchgeführt. Der Speicherinhalt des ROMs besteht vollständig aus Dateien, die auf feste Adressen abgebildet sind. Aufgrund dieser Tatsache können Programme direkt im ROM ausgeführt werden. [56, S.41]

In *Symbian* wird genauso wie das ROM das RAM von der MMU verwaltet. Das physikalische RAM wird von der MMU in 4K-Blöcke aufgeteilt. Folgende Inhalte können diesem Speicherbereich zugeordnet werden:

- Der virtuelle Adressraum eines User-Prozesses, wovon es mehrere geben kann.
- Der virtuelle Adressraum des Kernel Server Prozesses.
- RAM-Disk als *Laufwerk C*. Auf diesen Speicherbereich kann ausschließlich der Fileserver Prozess zugreifen.
- DLLs, die nicht von ROM geladen werden. Das RAM wird nach dem Laden der DLLs als Read-only makiert. Für jeden Thread erscheint jede DLL, der sie verwendet, an exakt der gleichen Stelle.
- Die Mapping-Tabelle der MMU, die für die Verwaltung zwischen virtuellen und physikalischen Adressräumen benötigt wird. Sie ist drauf optimiert, diese möglichst klein zu halten. Eine Grenze für die Anzahl an Prozessen gibt es in *Symbian*-OS nicht.
- Free-List Blöcke. Dies sind die Blöcke, welche noch nicht allokiert wurden.

Wenn für eine Anwendung Speicher aus dem RAM benötigt wird, dann wird er aus der Free-List entnommen. Darüber hinaus tritt bei einer Allokierung von Speicher, wenn keiner mehr zur Verfügung steht, eine Fehlermeldung „out-of-memory“ oder „disk-full“ auf. [56, S.42]
Immer, wenn ein *„.exe“*-Modul gestartet wird, wird ebenfalls ein Prozess gestartet. In diesem Prozess wird mindestens ein Thread ausgeführt. Es besteht außerdem die Möglichkeit, dass zur Laufzeit weitere Threads in dem Prozess ausgeführt werden, die für die Verarbeitung Speicherplatz benötigen.

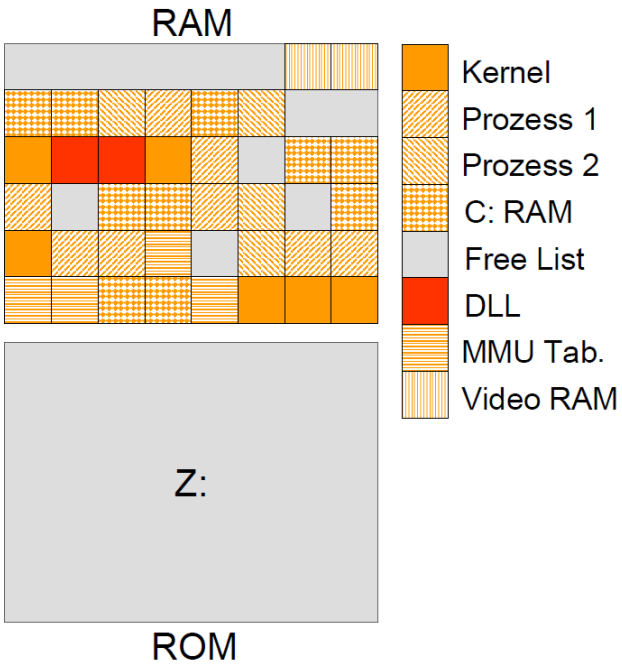

Abb. 12: Speicherverwaltung *Symbian* [56, S.42]

Der Adressraum eines Prozesses hält Speicher für folgende Bereiche bereit:

- Speicher, welcher systemübergreifend ist. Hierzu gehören das ROM und die in das RAM geladenen shared DLLs.
- Speicher, welcher prozessübergreifend ist. Hierzu gehört z.B. das Image des „.exe“ Moduls mit seinen beschreibbaren statischen Daten.
- Speicher für jeden Thread: einen sehr kleinen Stack und einen default Heap. Die maximale Speichergröße des Heap kann von *Symbian* festgelegt werden und beträgt in der Regel 2 MBytes.

Nach dem Ausführen eines Threads kann der Stack die vorgegebene Speichergröße nicht mehr verändern. Wenn auf den Stack eine größere Datenmenge abgelegt wird, als dieser umfasst, tritt ein Stack-Überlauf auf. Aufgrund dieser Tatsache löst der Thread eine Systemaktion aus (der Thread ”panicked”) und wird sofort beendet. Die initial Größe des Stacks beträgt 12 KBytes. Daraus folgt, dass große Variablen in der Regel auf dem Heap abgespeichert werden sollten, da er mehr Speicher zur Verfügung hat. [56, S.43]

4.5 Windows Phone

Das Betriebssystem *Windows Phone* ist eine Entwicklung der Firma *Microsoft*[34] und wird ähnlich wie *Android* von verschiedenen Mobilgeräteanbietern verwendet. Es ist im Jahr 2010 erschienen und ist der Nachfolger von *Windows Mobile*. Die Anbieter von Mobilgeräten, die das Betriebssystem *Windows Phone* verwenden, müssen für die Nutzung eine Lizenzgebühr an *Microsoft* zahlen. Im Gegenzug kümmert sich *Microsoft* stetig um Systemaktualisierungen, wobei das Unternehmen jedoch enge Vorgaben für die verwendete Hardware der Geräte vorgibt. Für den Download von Applikationen wird von *Microsoft* der sogenannte *Windows Phone Store* zur Verfügung gestellt. Eine Installation von Anwendungen aus Quellen von Drittanbietern ist nicht vorgesehen. Die Synchronisation von Adressbuch und Kalender wird hauptsächlich über das Internet vorgenommen. [76]

4.5.1 Aufbau und Funktionsweise von Windows Phone

Bei dem Aufbau von *Windows Phone 7* handelt es sich um ein geschlossenes System, wodurch kein direkter Zugriff auf das Dateisystem möglich ist. In Abbildung 13 ist die Architektur des Betriebssystems dargestellt, die sich aus Kernel, App Model, UI Model, Cloud Integration und Applications zusammen setzt.

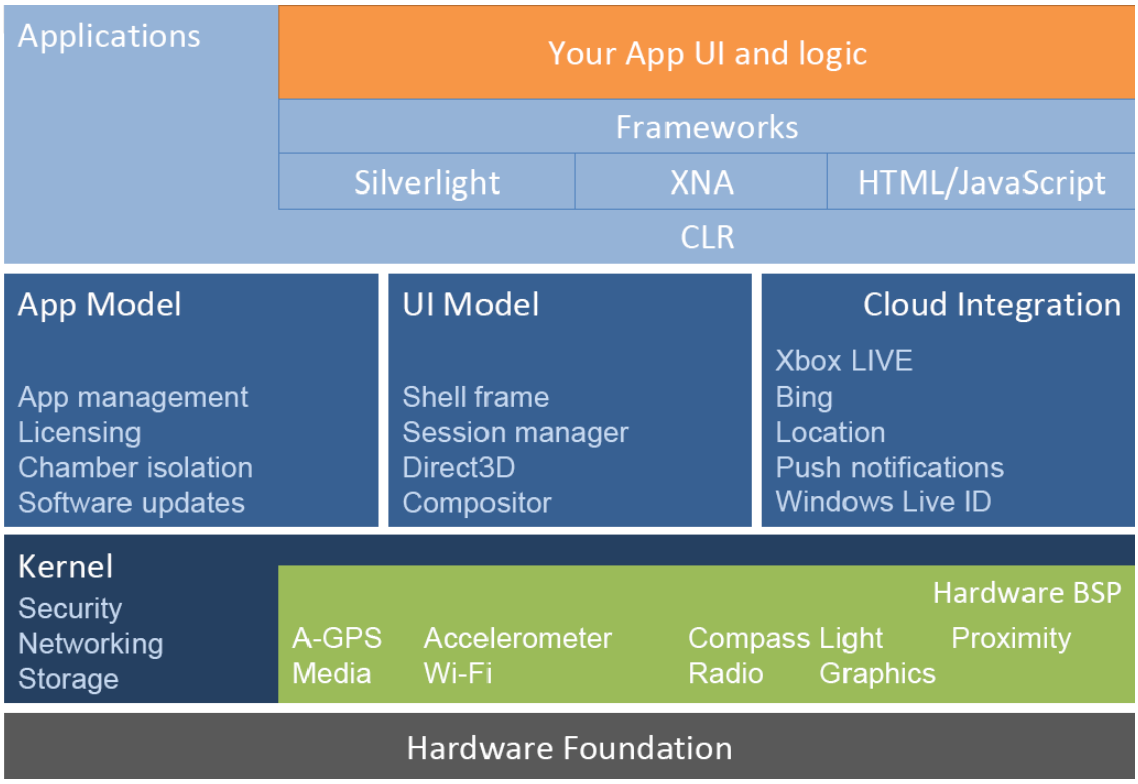

Abb. 13: *Windows Phone 7* Architektur [82, S.14]

[34] Bei *Microsoft* handelt es sich um eines der führenden Softwareunternehmen, welches im Jahre 1975 von Bill Gates und Paul Allen gegründet wurde. *Microsoft* bietet Betriebssysteme und Anwenderprogramme, sowie Hardwareprodukte an.[45]

Die beinhalteten Komponenten des *Windows Phone 7*-Betriebssystems werden nachfolgend detaillierter erläutert.

- **Kernel:** In dem Kernel, der direkt über der Hardware angesiedelt ist, werden Speicherverwaltung, sicherheitsrelevante Konzepte, sowie Netzwerkfunktionen geregelt. [82, S.14]
- **App Model:** Zur Verwaltung von Anwendungen und deren Lizenzierung wird die Komponente App Model benötigt. In ihr wird der Sandbox-Mechanismus bereit gestellt und es werden Softwareupdates verwaltet und geregelt. Von dem App Model werden ausschließlich Anwendungen gestartet, die eine gültige Lizenz aufweisen. Die Installation und Ausführung einer Anwendung erfolgt in einer Sandbox, womit Übergriffe von Schadsoftware verhindert werden.
- **UI Model:** In dem UI Model werden grundlegende Funktionen bereit gestellt, die unter anderem zur Darstellung von Anwendungsseiten benötigt werden. Außerdem wird von der Komponenten der Aufruf für unterschiedlichen Seiten verwaltet. Bei dem *Windows Phone*-Betriebssystem gibt es Applikationen, Sessions und Seiten. Eine Applikation besteht aus mehreren Seiten, die von Nutzern aufgerufen werden. Der Verlauf der unterschiedlichen Seiten wird von der Session gespeichert, sodass immer zu einer früheren Seite zurück gewechselt werden kann.
- **Cloud Integration:** Es werden verschiedene APIs zur Kommunikation mit bestehenden Web 2.0 Services (z.B. Windows Live, Xbox Live, bing) angeboten. Darüber hinaus besteht die Möglichkeit, eigene Services in einer Anwendungen zu nutzen. [82, S.15]

Um Anwendungen für *Windows Phone 7* zu konzipieren hat *Microsoft* eine Version vom *.net-Framework* entwickelt, wie man es vom PC kennt, das auf die Eigenschaften von mobilen Geräten angepasst ist. Bei der neu entwickelten Version handelt es sich um das sogenannte *Microsoft Silverlight Framework*. Es werden unterschiedliche Anwendungen ausschließlich in einem eigenen Prozess gestartet, die jeweils verschiedene Speicherbereiche zur Verfügung gestellt bekommen. Durch diese Eigenschaft kann eine Anwendung nicht auf den Speicherbereich einer anderen Anwendung zugreifen und diesen verändern. [84] Zur Entwicklung von Spielen für das *Windows Phone 7*-Betriebssystem dient das *XNA-Framekwork*. Durch dieses Framework werden Entwicklern verschiedene Programmierschnittstellen, wie *Direct3D* aus *DirectX* für die Darstellung von 2D und 3D Grafiken zur Verfügung gestellt. Das *Microsoft Silverlight Framework* sowie das *XNA-Framekwork* bilden zusammen mit den *Windows Phone*-Komponenten die Grundlage zur Entwicklung einer Anwendung.

Die Komponenten zur Erstellung einer Anwendung für das *Windows Phone 7*-Betriebssystem sind im nachfolgenden dargestellt.

- **Silverlight Framework:** Dieses Framework dient zur Erstellung des Anwendungsinterfaces (die Einbettung von Videos, die Nutzung von *Windows Phone* Bedienelementen).
- **XNA Framework:** Das *XNA-Framework* bietet Software, Dienste und Ressourcen für die Spieleentwicklung.

- **Sensors:** Über diese Komponente können Entwickler Daten von der verwendeten Hardware auslesen und verarbeiten (z.B. Kompass, Multi-Touch Eingabe usw.).
- **Media:** Es wird vom System eine API zur Verfügung gestellt, womit die Wiedergabe von Musik, Grafiken und Animationen realisiert werden kann.
- **Data:** Es wird für jede Anwendung ein virtueller Ordner angelegt, wodurch der Speicher nach dem Sandboxing-Prinzip verwaltet wird.
- **Location:** In Location wird den Entwicklern der Zugriff auf bestimmte physikalische Daten des Benutzers erlaubt. [86, S.3]

Auf den Mobilgeräten von *Microsoft* werden ausschließlich Anwendungen zugelassen, die aus dem eigenen *Windows Phone Store* stammen. Vor dem Bereitstellen einer Applikation im *Windows Phone Store*, wird diese von *Microsoft* geprüft. Vor Ausführung der Anwendung wird untersucht, ob der Quellcode den Sicherheitsregeln der *Common Language Runtime (CLR)* des *Microsoft .net-Framework* entspricht. Es soll mit der Prüfung vermieden werden, dass Programmierfehler, wie beispielsweise Pufferüberläufe, in dem Quellcode vorhanden sind. Nach der Prüfung, die in der Regel 5 Tage dauert, steht die Anwendung zum Download bereit. Im Gegensatz zum Vorgänger-Betriebssystem *Windows Mobile* hat *Windows Phone 7* weniger Management- und Sicherheitsfähigkeiten integriert. Aufgrund der Tatsache, dass *Windows Phone* hauptsächlich für Privatnutzer konzipiert ist, beinhaltet es wenige Sicherheitsrichtlinien. Um mehr Sicherheit zu erreichen, werden einige *Exchange ActiveSync (EAS)*[35] Richtlinien unterstützt.[86, S.4]

Die Sicherheitsrichtlinien, die vom *Windows Phone 7*-System bereit gestellt werden, sind im Nachfolgenden dargestellt.

- **Password Required:** Zum Entsperren des Geräts kann eine Abfrage eines PIN-Codes erfolgen.
- **Minimum Password Length:** Hiermit wird die minimale Länge des Passworts festgelegt.
- **Idle Timeout Frequency Value:** Hiermit wird festgelegt, in welcher Zeitspanne das Mobilgerät automatisch gesperrt wird.
- **Device Wipe Threshold:** Es wird festgelegt, wieviele Fehlversuche bei der PIN-Eingabe gemacht werden dürfen, bis sämtliche Daten auf dem Gerät gelöscht werden.
- **Allow Simple Password:** Es wird erlaubt oder untersagt, ob einfache Passwörter erlaubt sind.
- **Password Expiration:** Es wird eine Zeitspanne festgelegt, in der das Passwort erneuert werden muss.
- **Password History:** Es wird verhindert, dass der Benutzer dasselbe Passwort erneut verwendet. [86, S.5]

[35] *Exchange ActiveSync* ist ein Protokoll, das auf XML-basiert und über HTTP und HTTPS kommuniziert. Das Protokoll wurde entwickeln, um Daten, wie E-Mails, Kontakte usw. zwischen einem mobilen Gerät und einem Nachrichten-Server zu synchronisieren. Zudem können mit dem Protokoll Geräte-Policies sowie Geräteeigenschaften eingestellt werden.[46]

Windows Phone 7 bietet Zugangsschutz per PIN und Passwort-Richtlinien für *Exchange ActiveSync (EAS)* an. Die übertragenden Daten werden per Secure Sockets Layer (SSL)[36] verschlüsselt, je nach Server-Verbindung mit 128 oder 256 Bit. Bei Mobilgeräten mit *Windows Phone*-Betriebssystem ist es nicht möglich eine SD-Karte zu installieren (es ist nur ein Festeinbau möglich). Mit diese Maßnahme verhindert *Microsoft*, dass durch die Entnahme der Speicherkarte ein unerwünschter Zugriff auf die Benutzerdaten besteht. Darüber hinaus wird vermieden, dass Schadsoftware durch die SD-Karte in das System integriert wird. [84]

[36] *Secure Sockets Layer* ist eine Transportsicherheitsschicht, wovon der Nachfolger *Transport Layer Security (TLS)* ist. Hierbei handelt es sich im ein Verschlüsselungsprotokoll zum Sichern von Datenübertragungen im Internet. Bekannte Implementierungen sind *OpenSSL* und *GnuTLS*.[34]

5 Vergleich der Betriebssysteme

In diesem Kapitel soll ein Überblick verschafft werden, was für die App-Entwicklung benötigt wird, damit eine entwickelte Anwendung zur Installation für das jeweilige Betriebssystem bereit gestellt wird und es wird darüber hinaus dargestellt, welche Sicherheitsfunktionen in den untersuchten Systemen bereits integriert sind.

5.1 Anwendungsentwicklung für mobile Betriebssysteme

Es sind in der nachfolgenden Tabelle 1 die mobilen Betriebssysteme aufgeführt, welche im vorherigem Kapitel untersucht wurden. In der Tabelle wird dargestellt, ob beispielsweise Kosten für die Entwicklung einer Anwendung auftreten oder ein bestimmtes Entwicklungsprogramm verwendet werden muss.

Die nachfolgende Tabelle zeigt eine Gegenüberstellung der mobilen Betriebssysteme.

	iOS	*Android*	*Windows Phone 7*	*Blackberry*	*Symbian*
Registrierung	Ja	Ja	Ja	Ja	Ja
Kosten	90 € pro Jahr	einmalig 25 €	Nein	Nein	Nein
Entwicklungsprogramme	Apple Tools	Java Entwicklungsumgebung	Java Entwicklungsumgebung	Windows Phone Development Tools	C++ Entwicklungsumgebung
System	*Mac* OS X	beliebig	beliebig	*Windows*	beliebig
Prüfen der Anwendung	Ja	Nein	Ja	Ja	Ja
Nutzbare Endgeräte	nur iDevices	alle *Android* Handys	alle *Windows Phones*	*Blackberry* Geräte	*Symbian* Geräte
Autoritäre Signierung	Ja	Nein	Ja	Ja	Ja

Tab. 1: Übersicht der App-Entwicklung [62],[78],[86, S.5]

Aus Tabelle 1 ist zu entnehmen, dass für eine Bereitstellung einer Anwendung für jedes System eine Registrierung benötigt wird. Kosten treten ausschließlich bei *Apples iOS* und *Android* auf, die bei *Apple* 90 € pro Jahr und bei *Android* einmalig 25 € betragen. Für die Entwicklung von *iOS* wird das *Mac OS X*-Betriebssystem und für *Blackberry* das *Windows*-Betriebssystem benötigt, wobei die Wahl des verwendeten Betriebssystems für die Entwicklung der weiteren mobilen Betriebssysteme beliebig ist. Außerdem wird aus der Tabelle 1 deutlich, dass ausschließlich *Android*-Anwendungen vor dem Bereitstellen keiner Prüfung unterzogen werden und diese auch keine autoritäre Signierung beinhalten müssen.

5.2 Integrierte Sicherheitsfunktion von mobilen Betriebssystemen

Um die Nutzerdaten auf den Mobilgeräten zu schützen, sind in den mobilen Betriebssystemen bereits Sicherheitsfunktionen integriert. Außerdem besteht die Möglichkeit, dass bestimmte Sicherheitsfunktionen durch externe Server geregelt werden können. Die integrierten Funktionen werden im nachfolgenden erläutert.

- **On-device encryption/over-the-air data encryption:** Durch die Sicherheitsfunktion werden die Nutzerdaten auf dem Gerät und bei der Übertragung verschlüsselt. Bei z.B. dem Workspace *Samsung Knox*[37] werden die Daten von internem und externem Speicher mit einem 256-Bit-AES Verschlüsselungs-Algorithmus verschlüsselt. Dieser Schlüssel wird abgeleitet vom Passwort des Benutzers.[58]
- **Complex passwords:** Es kann vorgegeben werden, ob ein komplexes Passwort verwendet werden muss.
- **Enforce password policies:** Es werden bestimmte Passwort-Richtlinien durchgesetzt, wie z.B. das Verwenden von Umlauten, Zahlen, Groß- und Kleinbuchstaben.
- **Remote wipe:** Durch die Unterstützung von *Remote Wipe* haben die Nutzer des Mobilgeräts die Möglichkeit die Daten auf dem Gerät aus der Ferne zu löschen.
- **Remote lockout:** Es kann aus der Ferne das Ausloggen und Passworteingabe erzwungen werden.

In der nachfolgenden Tabelle 2 sind die mobilen Betriebssysteme mit den integrierten Sicherheitsfunktionen
dargestellt.

	iOS	*Android*	*Windows Phone 7*	*Blackberry*	*Symbian*
On-device encryption	Ja	Ja	Nein	Ja	Ja
Over-the-air data encryption	Ja	Ja	Ja	Ja	Ja
Complex passwords	Ja	Ja	Nein	Ja	Ja
Enforce password policies	Ja	Ja	EAS	BES	Ja
Remote wipe	Ja	Ja	EAS	BES	Ja
Remote lock-out	Ja	Ja	EAS	BES	Ja

Tab. 2: Unterstützung von Sicherheitsfunktionen [8],[15],[16],[52],[86, S.6]

Aus Tabelle 2 ist zu erkennen, dass die Betriebssysteme *Apple iOS*, *Android* und *Symbian* alle aufgeführten Sicherheitsfunktionen unterstützen. Es wird deutlich, dass bei *Windows Phone 7* eine Verschlüsselung der Nutzerdaten auf dem Gerät nicht vorgenommen wird. Außerdem wird auch nicht die Eingabe eines komplexen Passworts verwendet. Weitere

[37] Bei Samsung KNOX handelt es sich um eine Sicherheitsfunktion für Android-Systeme. Diese basiert auf dem NSA-System „Security Enhanced Android“ (SE Android). Es wird ein besonders gesicherter Bootloader und eine speziell gesicherte Hardware verwendet. Dadurch, dass von *Samsung KNOX* z.B. Secure Boot verwendet wird, kann bereits beim Start des Betriebssystems sichergestellt werden, dass keine Schadsoftware geladen wird. Zudem soll ein Modul Integrity Measurement Architecture (TIMA) sicherstellen, dass keine Sicherheitsprobleme vorliegen.[13]

Einstellungen, wie Passwort-Richtlinien oder das Verwalten des Mobilgeräts aus der Ferne werden über *Exchange ActiveSync* (EAS) geregelt. Das *Blackberry*-Betriebssystem unterstützt die Verschlüsselung der Nutzerdaten auf dem Gerät und bei der Übertragung sowie das Verwenden eines komplexen Passworts. Ähnlich wie bei *Windows Phone 7* werden Passwort-Richtlinien und die Verwaltung aus der Ferne über einen externen Server vorgenommen, was bei *Blackberry* über den *Blackberry Enterprise Server*[38] realisiert wird.

[38] Der *Blackberry Enterprise Server* dient dazu, dass Benutzerdaten, wie z.B. Adressbuch, Kalender usw. mit dem mobilen Gerät synchronisiert werden. Bei BES aktivierten Geräten werden E-Mails, Kalendereinträge, Notizen und Adressbucheinträge per Push-Dienst vom Server an das Endgerät gesendet. Bevor die Daten an das Endgerät gesendet werden, werden diese komprimiert, um die Datenmenge möglichst gering zu halten.[51]

6 Schadsoftware

In diesem Abschnitt wird untersucht, wie sich die Zahl der entdeckten Schadprogramme in den Jahren von 2010 bis 2014 entwickelt hat und welche mobilen Betriebssysteme davon betroffen sind. Des Weiteren wird erläutert, welche Möglichkeiten es für Schadecode-Entwickler gibt, ihre Software zu verbreiten und wie sie beispielsweise damit in der Lage sind, Nutzerdaten zu stehlen. Es wurde für die unterschiedlichen Betriebssysteme geprüft, welche Schadprogramme es bereits gegen die Systeme gab und wie diese ausgeführt werden. Aufgrund der gewonnenen Erkenntnisse wird dargelegt, worauf geachtet werden sollte, damit eine Infizierung von Schadcode vermieden wird.

6.1 Verbreitung von Schadsoftware

Aufgrund der Tatsache, dass Mobilgeräte immer mehr in den Alltag integriert werden, erreichen die Betriebssysteme der Geräte eine immer größere Verbreitung. Bei mobilen Geräten handelt es sich um sehr leistungsstarke Computer, die genauso wie ein Heimcomputer, von Schadsoftware infiziert werden kann. Durch die stetig wachsende Verbreitung sowie die zunehmende Leistungsfähigkeit dieser Systeme sind sie attraktiv für Hacker und Behörden geworden. In den nachfolgenden Abbildungen 14 - 16 sind die Anzahl der Bedrohungen von Schadsoftware für die jeweiligen Betriebssysteme dargestellt. In Abbildung 14 ist prozentual die Anzahl der entdeckten Schadsoftware 2010 dargestellt. Es ist zu entnehmen, dass im Jahr 2010 80 neue Schadsoftware-Familien entdeckt wurden, wovon sich 62,5% gegen das Betriebssystem *Symbian* richtete. 2010 lag der Absatz von Smartphones mit *Symbian*-Software bei 37,6% (siehe Abb. 1, Seite 10). Der hohe Schadsoftwareanteil ist aufgrund der starken Marktpräsenz des *Symbian*-Systems in den Jahren von 2005 bis 2010 zu erklären. Von den entdeckten Schadsoftware-Familien wurden 23,75% für *Windows Mobile* und 11,25% für das *Android*-Betriebssystem entwickelt. Bereits im Jahr 2011 wurden 195 neue Schadsoftware-Varianten bekannt, womit sich die Anzahl, im Gegensatz zum Vorjahr, mehr als verdoppelte. Es waren ca. 30% der entdeckten Software auf das *Symbian* ausgerichtet. Durch eine Marktpräsenz des *Android*-Systems von 49,2% (siehe Abb. 1, Seite 10) im Jahr 2011 richtete sich immer mehr Schadsoftware gegen das System. Aus Abbildung 15 ist zu entnehmen, dass sich ca. 2/3 (66,7%) der erkannten Schadprogramme gegen das Betriebssystem *Android* richtete. Das System *Windows Mobile* verzeichnete 1%, wobei für die Systeme *Blackberry* und *Apple* keine Schadsoftware entdeckt wurde.

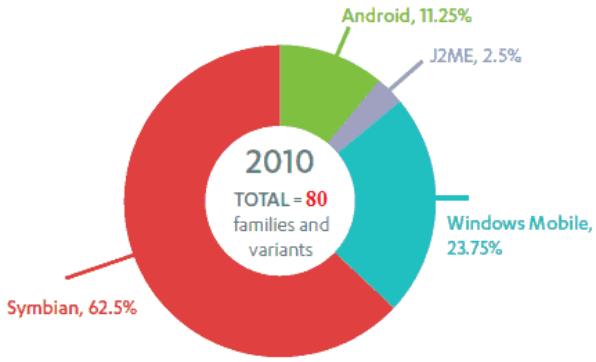

Abb. 14: Schadsoftware 2010 [9]

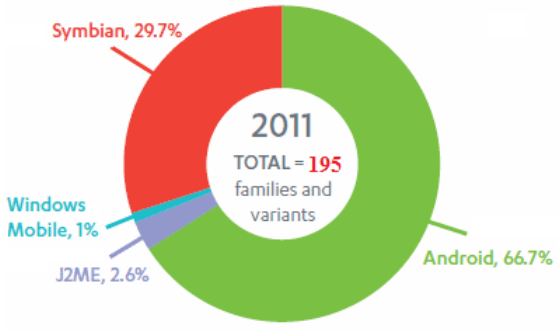

Abb. 15: Schadsoftware 2011 [9]

In dem Jahr 2012 konnte ein weiterer Anstieg im Vergleich zum Vorjahr, von 195 auf 301 entdeckte Schadsoftware-Familien, verzeichnet werden. Aufgrund der starken Marktpräsenz von 69% (siehe Abb. 1, Seite 10) richteten sich 2012 79% der gefunden Schadprogramme gegen das *Android*-System. Der Marktanteil von *Symbian* hatte sich in diesem Jahr weiter verringert und lag bei 2,6% (siehe Abb. 1). Dennoch waren 19% der erkannten Schadsoftware auf das System ausgerichtet. Für *Blackberry* und *Apple* wurden zum ersten Mal Schadprogramme entdeckt. Um Zugriff auf Nutzerdaten zu bekommen, werden in der Regel Schadprogramme entwickelt, die von Nutzern selbständig installiert werden. Es besteht jedoch ebenfalls die Möglichkeit, dass durch das "Klicken" auf einen Werbebanner einer Internetseite Schadsoftware installiert wird. Aus dem Sicherheitsreport von *F-Secure* für das zweite Halbjahr 2013 geht hervor, dass die Schadsoftware für das Betriebssystem *Android* von 2012 bis 2013 um 18% auf 97% gestiegen ist. Die restlichen 3% breiten sich laut *F-Secure* auf *Symbian*-basierte Geräte aus. Für die Plattformen *iOS* oder *Windows Phone* sind keine neuen Schadprogramme gefunden worden. Der Schadsoftwareanteil in *Googles Play Store* beträgt 0,1%, wobei in alternativen App-Stores, die überwiegend aus China stammen, mehr Schadprogramme verzeichnet werden. In den Stores bei z.B. *Anzhi*, *Mumayi*, *Baidu* und *eoeMarket* wurde ein Schadsoftwareanteil von 10% festgestellt. Die höchste Schadsoftware-Rate liegt laut *F-Secure* in dem App-Store *Android159*, indem 33,3% gefunden wurden. [71] Im ersten Quartal 2014 haben die Forscher von F-Secure-Labs insgesamt 277 neue Familien und Varianten mobiler Bedrohungen festgestellt. Davon richteten sich 275 gegen die Plattform *Android*, was 99% der entdeckten Schadsoftware sind. Jeweils ein Schadcode war auf *iOS* und *Symbian* ausgerichtet.

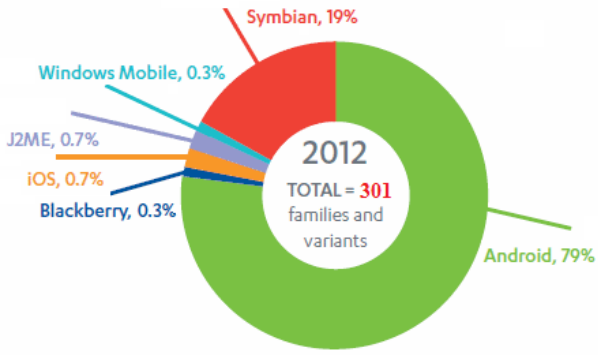

Abb. 16: Schadsoftware 2012 [9]

6.2 Arten der Angriffe

Bei Mobilgeräten, wie Smartphones, handelt es sich in der Regel nicht mehr um Mobiltelefone, sondern eher um kleine Notebooks, mit denen telefoniert werden kann. Dadurch, dass die Betriebssysteme für Mobilgeräte stetig komplexer und leistungsfähiger werden, werden diese somit auch angreifbarer. Durch die Mobilität der Geräte besteht nicht die Möglichkeit, dass sie, wie beispielsweise stationäre PC vom Firmen-Netzwerk und deren IT-Sicherheit, geschützt werden. Die Angriffswege gegen Mobilgeräte gleichen wegen ihrer Computer-Eigenschaften immer mehr den Angriffen gegen traditionelle PCs. Diese werden im Nachfolgenden erläutert.

6.2.1 Einschleusen von Schadsoftware

Ein großes Sicherheitsrisiko stellt für Mobilgeräte die Installation von Schadsoftware dar, wofür es keine Rolle spielt, ob die Installation von Nutzern selbst oder auf anderen Wegen ausgeführt wird.

Für die Installation von Schadsoftware lassen sich drei relevante Wege unterscheiden:

a) Vom Nutzer selbständig installierte Apps, die Schadcode enthalten. Dieses sind Programme die unter dem Deckmantel von glaubwürdigen Anwendungen wie z.B. Updates installiert werden.

b) Spionagesoftware, die durch physischen Zugriff auf das Mobilgerät durch Dritte unbemerkt installiert wird.

c) Installation durch das Ausnutzen von Schwachstellen in den Betriebssystemen oder deren Hilfsprogrammen, sogenannten Exploits.

Um Schadsoftware auf dem Mobilgerät zu installieren, wird am häufigsten das Unterschieben einer Anwendung, die Schadcode enthält, verwendet, die von den Nutzern selbständig installiert wird. Den Herstellern der mobilen Betriebssysteme ist dieses Vorgehen bekannt, sodass diese durch verschiedene Mechanismen, wie z.B. das Prüfen einer App bevor sie zum Download bereit gestellt wird, die Wahrscheinlichkeit von Schadcode-Angriffen über Apps minimieren. Es werden jedoch stetig Lücken in den Systemen entdeckt, wodurch das Einschleusen von Schadsoftware ermöglicht wird. Um mobile Geräte mit Schadprogrammen zu infizieren oder ihre Kommunikation zu manipulieren, stehen verschiedene Wege zur Verfügung, die nachträglich erläutert werden.

6.2.2 Drive-by-Angriff

Bei einem drive-by-Angriff werden Internet-Nutzer auf Internetseite verwiesen, auf der Cyber-Kriminelle Schadsoftware hinterlegt haben. Im Jahr 2012 wurde so ein Angriff von dem ehemaligen *McAfee*-Manager George Kurtz und Dmitri Alperovitch demonstriert, wobei ein Bug im *Android*-Browser „WebKit“ ausgenutzt wurde. Der Angriff wurde durch eine angebliche SMS des Mobilgeräteherstellers gestartet, in der ein Link enthalten war um ein wichtiges Update herunterzuladen zu können. Durch das Aufrufen der Internetseite aus dem Link wurde unwissentlich Schadsoftware heruntergeladen, welche auf dem Mobilgerät den sogenannten Loader ausführte. Das Gerät wird dadurch zum Abstürzen gebracht, woraufhin ein Neustart durchgeführt wird, der dann die Schadsoftware installiert. Es besteht nach der Installation die Möglichkeit Telefonate mitzuschneiden und die Kamera zu aktiviert. Außerdem können gewählte Telefonnummern, gespeicherte SMS sowie der aktuelle Standort festgestellt und an einen Server übertragen werden.[29] Das unwissentliche Herunterladen der Schadsoftware von der Internetseite wird „drive-by-Download“ genannt, was häufig zur Industriespionage verwendet wird. Der drive-by-Angriff erfolgt in mehreren Schritten. Im ersten Schritt wird beim öffnen der Internetseite der Typ und die Version des Browsers, sowie das verwendete Betriebssystem ermittelt. Browser und Betriebssystem werden daraufhin auf Sicherheitslücken untersucht und bei einem Fund wird eine Verbindung zu einem Server aufgebaut, von dem die passende Schadsoftware heruntergeladen wird. Dieser Vorgang geschieht, ohne das die Anwender etwas davon mit bekommen.

6.2.3 Schwachstellen im Betriebssystem

Durch die stetig steigende Funktionalität der Betriebssysteme steigen auch die Angriffsmöglichkeiten. Für mobile Betriebssysteme gibt es heutzutage sehr viele Abspielmöglichkeiten von Multimedia-Inhalten. Die unterschiedlichen Abspielkomponenten sind in der

Regel nicht direkt vom Hersteller sondern werden von Dienstleistern zugekauft. Unterschiedliche Software für diese Abspielkomponenten besitzen meistens die gleichen Kern-Bibliotheken (Quellcode des Programms). Sobald diese Programme eine Schwachstelle aufweisen, weist auch das Betriebssystem eine Schwachstelle auf. Ein Blick auf die Erweiterung zum Abspielen von Videos „ffmpeg“ zeigt, wie groß diese Schwachstellen sein können. Mitte 2012 wurden nach dieser Erweiterung 56 Schwachstellen entdeckt, wovon 28 als sehr schwerwiegend eingestuft wurden, da sie eine enorme Auswirkung auf das System haben könnten. Ein weiteres Beispiel ist die HTML-Rendering Engine „WebKit“, dass zur Implementierung von Web-Browser sowie zur Erstellung von Benutzeroberflächen dient. Da sich „WebKit“ in *iOS*, *Android* und *Blackberry* befindet, wird mit einer Schwachstelle eine Angriffsmöglichkeit auf eine Vielzahl von mobilen Betriebssystemen ermöglicht. Wie schwerwiegend eine Lücke im „WebKit“ für ein Betriebssystem ist, hängt davon ab, wie stark es in das jeweilige System eingebettet ist. Bei *Android* ist es so tief integriert, dass nach einem erfolgreichen Angriff auf Funktionen des Mobilgeräts zugegriffen werden kann.[60]

6.2.4 Angriffe über Gerätesynchronisation

Mobilgeräte werden immer stärker in Geschäfts- und Kommunikationsprozesse eingebunden, wobei Daten synchronisiert oder als Backup mit der unternehmenseigenen Infrastruktur regelmäßig ausgetauscht werden. Zur Synchronisation wird in der Regel der stationäre PC verwendet, in der Schwachstellen bisher von IT-Fachleuten wenig in Betracht gezogen wurden. Es besteht die Möglichkeit, dass Schwachstellen bei dem Datenaustausch zwischen Mobilgerät zu Endgerät ausgenutzt werden, womit Angriffe in beide Richtungen realisiert werden können. Dadurch kann ein Mobilgerät Schadsoftware in das lokale Netz einschleusen oder ein Mobilgerät wird bei der Synchronisation von einem Schadprogramm, das in dem lokalen Netz abgelegt ist, infiziert. Es hat sich für die Identifikation von Schwachstellen in Betriebssystemen und Softwarekomponenten ein Markt entwickelt, auf dem zahlreiche Käufer vorhanden sind. Laut „Washington Post“ kauft alleine die NSA Informationen über derartige Lücken für Angriffe auf Computersysteme für 25 Millionen Dollar pro Jahr ein.[60]

6.2.5 Umleitung von Daten „Man-in-the-middle“

Bei einer „Man-in-the-middle“-Attacke verschafft sich ein Angreifer Zugriff auf die Kommunikation zwischen zwei Teilnehmern. Realisiert werden kann es, indem ein Angreifer beispielsweise auf seinem Notebook einen schädlichen Router einrichtet, um die Zugriffe des Angriffsziels zu kontrollieren. Hierfür kann die drahtlose Verbindung des Notebooks so konfiguriert werden, dass diese wie ein W-LAN-Hotspot funktioniert, wie sie an Flughäfen oder Cafés verwendet werden. Wenn Nutzer sich mit dem Router verbinden und sich z.B. auf Internetseiten mit ihrem Login einloggen, besteht für den Angreifer die Möglichkeit, die Benutzerdaten mitzulesen und somit zu stehlen. Ein weiterer Angriff könnte erfolgen, indem ein Angreifer eine Schwachstelle in der Konfiguration oder in der Verschlüsselung eines bestehenden Hotspots identifizieren kann. Mit der Sicherheitslücke besteht ebenfalls die Möglichkeit, die Kommunikation zwischen Anwender und Router abzuhören oder die gesendeten Daten zu manipulieren. [41]

6.3 Android

Durch die starke Marktpräsenz von *Android*, im Jahr 2013 mit ca. 80% Weltmarktanteil (siehe Abb. 1 auf Seite 10), ist das Betriebssystem sehr attraktiv für Angreifer, da durch einen Angriff eine Vielzahl an potentiellen Opfern existiert. Der Anteil der neu entdeckten Schadsoftware lag im Jahr 2010 bei 11,25% (siehe Abb. 14 auf Seite 42) und stieg bis zum ersten Quartal 2014, laut den Sicherheitsexperten von F-Secure auf 99% an. Auf die Installation von Schadsoftware durch den Nutzer hat *Google* in der Regel keinen Einfluss. Wird eine Anwendung auf dem *Android*-System installiert, müssen die Nutzer die Berechtigungen der App, für beispielsweise Nutzung und Weitergabe von sensiblen Daten oder den Zugriff auf Schnittstellen, bestätigen. In der Regel wird Schadsoftware auf dem *Android*-System durch Unwissenheit von Nutzern installiert.

6.3.1 Schadsoftware für Android

In den meisten Fällen wird Schadsoftware in Anwendungen integriert, die unter einem Decknamen angeboten werden. Es besteht dadurch z.b. die Möglichkeit, dass eine Taschenlampen-App angeboten wird, die Zugriff auf Bilder und Kontaktdaten fordert, welche von den Nutzern bestätigt werden müssen, damit diese Berechtigungen erteilt werden. Des Weiteren werden Sicherheitslücken, um Schadsoftware zu installieren, in den Betriebssystemen ausgenutzt. Im Jahr 2012 wurde eine Sicherheitslücke entdeckt, die in einer Code-Bibliothek vorhanden war, wodurch mit Hilfe einer präparierten Internetseite und einem Java-Script, der Zugriff auf interne Funktionen des Betriebssystems ermöglicht wurde. Das hinterlegte Java-Script wurde so entwickelt, dass es sich beim Öffnen der Internetseite, in den Browser und in Apps, die die Code-Bibliothek verwenden, integrieren und somit für Angreifer ein Fernzugriff auf das Mobilgerät ermöglichen. Durch den Zugriff aus der Ferne können Nutzerdaten ausgespäht und gestohlen werden. Mit der *Android*-Version 4.2 Jellybean wurde die Sicherheitslücke behoben, die jedoch nicht von allen *Android*-Nutzern installiert werden kann, da die Mobilgerätehersteller, die die aktuelle Version an die verwendete Hardware der Geräte anpassen müssen, für Updates ihrer Produkte zuständig sind. Durch diese Update-Politik müssen einige Nutzer länger auf Updates warten als andere. Es werden bislang 27% der Mobilgeräte mit *Android*-Version 4.2 oder höher verwendet, wodurch 73% für den genannten Angriff anfällig sind. Selbst bei der Version 4.2 sowie der höheren Versionen besteht die Möglichkeit, dass die Sicherheitslücke, durch eine installierte Anwendung, die mit einer älteren Code-Bibliothek entwickelt wurde, in dem System integriert wird. [70] Heutzutage gelangen Schadprogramme nicht direkt durch einen Angriff auf die Mobilgeräte. Der Zeus-Trojaner, welcher im Jahr 2012 von dem Antivir-Hersteller Kaspersky entdeckt wurde, wird über einen stationären PC mit *Windows*-Betriebssystem eingeleitet. Durch die Infizierung eines *Windows*-PCs mit dem Zeus-Trojaner, wird beim Öffnen der Webseite der Bank ein Meldung angezeigt, dass neue Zertifikate für die Sicherung des Mobilgeräts installiert werden müssen. Die angeblichen Zertifikate werden per Download oder via MMS an die Handynummer des Nutzers gesendet. Nach der Installation der empfangenen Datei werden SMS-Nachrichten, welche den TAN für das Online-Banking beinhalten, an eine Nummer im Ausland weitergeleitet. [61]

6.4 Apple iOS

Die Nutzer von *Apples iOS* sind in der Regel eher selten von Schadsoftware betroffen, da *Apple* mit dem *iOS*-Betriebssystem ein geschlossenes und unveränderbares System

entwickelt hat, was nur Installationen aus dem *App-Store* zulässt. Durch die restriktive und strenge *App Store*-Strategie sowie der Qualitätssicherung wird ein relativ sicheres System geboten. Es besteht jedoch für Nutzer die Möglichkeit, mit Hilfe eines Jailbreak das Gerät dahingegen zu modifizieren, dass Nutzer den Zugriff (sogenannte Root-Rechte) auf interne Funktionen sowie das Dateisystem bekommen, womit die Sicherheitsfunktionen von *Apple* nicht mehr wirksam sind. Durch diese Modifikation wird die Softwareverwaltung des Geräts angepasst, sodass Anwendungen von Drittanbietern installiert werden können. Die Firma *Apple* hat in den Jahren 2007 bis 2013 ca. 700 Millionen *iOS*-Geräte auf den Markt gebracht, wovon bis März 2013 ca. 23 Millionen einen Jailbreak installiert hatten.

6.4.1 Schadsoftware für Apple iOS

Nach der Installation eines Jailbreak wird in der Regel „Cydia“ oder eine andere Alternative zum *App-Store* auf den *Apple*-Geräten installiert, womit die Installation einer Anwendung von Drittanbietern ermöglicht wird. Die Quellen der alternativen Stores werden, genauso wie beim *App-Store*, einer Zensur durch den Quellenbetreiber unterzogen, womit die Verbreitung von Schadsoftware minimiert werden soll. Der Zugriff auf illegale Inhalte, wie Raubkopien, wird durch einen Jailbreak geschaffen, da die Quellen des alternativen Stores manuell erweitert werden können, wodurch die Gefahr steigt, dass Schadsoftware auf dem Gerät installiert wird. Bei einem *Apple iOS*-Gerät mit installiertem Jailbreak kann über den SSH-Dienst *OpenSSH*[39] Schadsoftware ausgeführt werden. Aufgrund der Tatsache, dass die *iOS*-Geräte vom Werk aus das Root-Passwort „alpine“ besitzen, kann ein Angreifer mit Root-Berechtigung auf die Geräte zugreifen, wenn die Nutzer aus Unwissenheit oder Bequemlichkeit das Passwort nicht geändert haben. Anfang 2009 hat ein Hacker mit dem *iPhone*-Wurm „ikee“ große Bekanntheit erlangt, indem er mit seinem Wurm die Sicherheitslücke des SSH-Daemons ausnutzte.[50]
Bereits im Sourcecode war vermerkt:

„Why?: Boredom, because i found it so stupid the fact that on my initial scan of my 3G optus range i found 27 hosts running SSH daemons, i could access 26 of them with root:alpine. Doesn't anyone RTFM anymore? “ [50]

Übersetzung:

„Warum?: Langeweile, da ich es so unglaublich dumm fand, dass ich bei meinem ersten Scan innerhalb meiner 3G-Funkzelle 27 Geräte gefunden habe, die SSH-Daemons installiert hatten, und ich mich bei 26 von diesen mit root:alpine einloggen konnte. Liest denn keiner mehr die verdammte Gebrauchsanleitung?“

Durch den *Iphone*-Wurm „ikee“ wurde lediglich das Hintergrundbild des Geräts verändert und kein Schaden angerichtet, da hiermit nur auf das Sicherheitsrisiko des SSH-Dienstes aufmerksam gemacht werden sollte. [50] Es gibt jedoch eine hohe Anzahl an Nachfolgern, die nach dem gleichen Prinzip funktionieren um die Bankdaten von Nutzern zu stehlen. [22] Im Jahr 2014 wurde eine Schadsoftware für das *Mac OS X*-System entdeckt, die aus einem App-Store eines Drittanbieters aus China stammt. Es handelt sich hierbei um den Schädling ”WireLurker“, der laut Palo Alto Networks in 467 unterschiedlichen *OS*

[39] Bei OpenSSH handelt es sich um ein Programmpaket für *Secure Shell (SSH)*, welches das *SSH File Transfer Protocol*, Clients, Dienstprogramme und einen Server beinhaltet. SSH ist ein Netzwerkprotokoll, sowohl ein Programm, womit die Möglichkeit besteht, eine verschlüsselte und somit sichere Netzwerkverbindung mit einem anderen Gerät aufzubauen.[55]

X-Programmen, welche in 6 Monaten ca. 350.000 heruntergeladen wurden, integriert ist. Nach der Installation der Schadsoftware auf einem *Mac* prüft dieser andauernd, ob ein *iOS*-Mobilgerät angeschlossen ist. Wenn ein Gerät angeschlossen wird, nutzt die Schadsoftware die Trusted-Pairing-Verbindung zwischen *Mac* und dem Mobilgerät aus, wobei die Software eine Schnittstelle für Entwickler verwendet, womit sie Apps direkt auf dem Gerät installieren und den üblichen Weg über den *App Store* umgehen. Durch diese Verbindung kann die Software auf das Gerät zugreifen und existierende Programme überschreiben. Laut dem Sicherheitsspezialisten Jonathan Zdziarski sammelt die Software Daten über die Nutzer zur eindeutigen Identifizierung und zur Überprüfung, welche nicht lizenzierten Programme benutzt werden. Es handelt sich hierbei um den ersten Bug, der auch *Apple*-Geräte infiziert, die keinen aktiven Jailbreak aufweisen. [42] [89]
Der Zeus-Trojaner, welcher unter „Schadsoftware für Android“ beschrieben wurde, funktioniert für *Apple iOS* nicht, da es grundsätzlich von Apple nicht vorgesehen ist, Programme von Drittanbietern zu installieren, sondern ausschließlich über den *App Store*.

6.5 Blackberry OS

Mit dem sehr geringen Marktanteil, der im Jahr 2013 bei 1,9% lag (siehe Abb. 1 auf Seite 10), des *Blackberry*-Betriebssystems und der damit verbundenen Sicherheitsinfrastruktur, ist die Anzahl an Schadprogrammen eher gering. Dennoch werden in diesem System Schwachstellen entdeckt und ausgenutzt.

6.5.1 Schadsoftware für Blackberry OS

Eine Schwachstelle zeigte sich z.B. in der *Blackberry Device Software* 6.0.0, die durch den *Blackberry*-Browser und eine manipulierte Internetseite ausgenutzt wurde. Mit dem Betreten der Internetseite wurde ein DoS-Angriff (Denial of Service)[40] gestartet, die den Browser zum Abstürzen bringt und in einigen Fällen das Gerät, durch die starke Belastung, neu gestartet werden muss. [48] Im Jahr 2011 wurde in bestimmten Versionen der *Blackberry Enterprise Software*, die zur Aufbereitung von Bildern dient, eine Schwachstelle identifiziert, in der Mobilgeräte nicht direkt betroffen waren, jedoch zur Infizierung mit Schadsoftware dienen konnten. Betroffen waren in der Regel Unternehmen, die über eigene Server mit *Blackberry*-Software verschlüsselte Verbindungen zwischen den *Blackberrys* der Mitarbeitern und internen Anwendungen herstellten. Die Schwachstelle konnte mit Hilfe eines manipulierten Bildes ausgenutzt werden, womit Schadsoftware auf dem Server installiert wurde, sodass Angreifer unbefugten Zugriff auf Teile des Netzwerks bekommen haben und dort eigene Programme ausführen konnten. Für das Ausnutzen dieser Schwachstelle bestehen zwei Möglichkeiten, wobei einerseits eine Internetseite erstellt werden kann, die ein Bild mit dem nötigen Schadcode enthält. Benutzer können daraufhin per E-Mail oder Chat-Programm den Link zu der Internetseite erhalten, womit bei Betreten der Seite die Schadsoftware installiert wird. Andererseits kann *Blackberry*-Nutzern ein Bild mit Schadcode per E-Mail gesendet werden, das auf dem Mobilgerät nicht geöffnet werden muss, um die Infizierung der Schadsoftware auszulösen, da es automatisch ausgeführt wird, wenn der *Blackberry*-Server das Bild, zum Weiterversand an den Empfänger, aufbereitet. [53]

[40] Denial of Service sind Angriffsversuche auf Rechner, Server oder ganze Netzwerke, die durch eine hohe Anzahl an Verbindungsversuchen überlastet werden und dadurch nicht mehr erreichbar sind, da sie die Menge an eingehenden Daten nicht verarbeiten können.[17]

6.6 Symbian

Die *Symbian*-Plattform ist im Bereich der mobilen Betriebssysteme Vorreiter gewesen und war im Jahr 2010 Marktführer mit 37,6% (siehe Abb. 1 auf Seite 10). Es war bis dahin das am meist genutzte mobile Betriebssystem und somit sehr attraktiv für Hacker und Schadsoftware-Entwickler geworden. Für eine Installation einer Anwendung auf dem *Symbian*-System müssen die Berechtigungen, die eine App fordert, bestätigt werden, damit diese installiert wird. Genauso wie bei *Android* wird in der Regel Schadsoftware durch Unwissenheit der Nutzer selbständig installiert.

6.6.1 Schadsoftware für Symbian

Im Juni 2004 wurde die erste Proof-of-Concept[41] Schadsoftware für die *Symbian*-Plattform entdeckt, wobei es sich um den Wurm[42] „EPOC.Cabir“ handelt. Dieser Wurm wurde nicht entwickelt, um Schaden anzurichten, sondern sollte lediglich zeigen, dass für mobile Betriebssysteme Schadsoftware existieren kann. Die Funktion des Wurms bestand darin, dass nach der Installation der Name im Display des Gerätes angezeigt wurde und dieser außerdem permanent über die Bluetooth-Schnittstelle nach Bluetooth-Geräten suchte. Sofern weitere Bluetooth-Geräte gefunden wurden, verschickte sich der Wurm selbständig als .SIS-Archiv. [6, S.115] Um die Datei zu erhalten, musste im Mobilgerät der Empfang von Daten zugelassen werden. Der Wurm ist nicht in der Lage sich selbständig zu installieren und muss von den Nutzern drei mal bestätigt werden. Hinzu kommt, dass zwei mal bestätigt werden muss, da es sich um eine unsichere Datei handelt. Ein weiteres Schadprogramm stellt das Trojanisches Pferd[43] „SymbOS/Skulls“ dar, welches über Shareware-Seiten angeboten wurde und sich unter dem Decknamen „Extended Theme Manager“ tarnte, das nach der Installation unterschiedliche Systemdateien überschreibt und dadurch die *Symbian*-Programme, die nach der Infizierung mit einem Totenkopf angezeigt wurden, unausführbar macht. Eine neue Ebene von Schadsoftware wurde durch den gefundenen *Symbian*-Wurm „CommWarrior“ erreicht. Dieser Wurm, der von Nutzern installiert werden muss, da er selbst nicht in der Lage dazu ist, war so konzipiert, dass er sich mit Hilfe des MMS-Protokolls weiter verbreiten konnte, was einer Verbreitung durch E-Mail gleich kommt. Er versendet sich über das Protokoll selbständig und verspricht beim Öffnen des Anhangs freie Software, Sexbilder, Betriebssystemupdates, Sicherheitssoftware oder Spiele. Eine Betreffzeile war beispielsweise „Norton AntiVirus: Released now for mobile, install it!“. Durch seine wechselnden Werbebotschaften wurden Nutzer neugierig und förderten somit die Verbreitung des Wurms. Der Wurm versendete sich an alle Telefonnummern im Telefonbuch des Systems. Durch die damaligen Kosten von ca. 50Cent pro MMS konnte dadurch ein großer finanzieller Schaden entstehen. [49]

Die Funktionsweise des Zeus-Trojaners, welche schon unter „Schadsoftware für Android“ beschrieben wurde, ist mit gleicher Funktion für die *Symbian*-Plattform verfügbar.

[41] Proof-of-Concept ist ein Prinzip, womit die Durchführbarkeit eines bestimmten Vorhabens überprüft wird. In der Regel ist mit diesem Prinzip ein Prototyp verbunden, der benötigte Kernfunktionen aufweist.

[42] Bei einem Wurm handelt es sich um ein Schadprogramm, das in der Lage ist, sich selbständig zu kopieren und zu versenden, was durch einen Massenversand enorme Netzwerkressourcen benötigt.[12]

[43] Ein Trojanisches Pferd ist ein Programm, das schädliche Funktionen beinhaltet, die im Hintergrund ausgeführt werden, um Nutzerdaten zu stehlen oder das System zu beeinflussen und zu stören.[79]

6.7 Windows Phone

Der Anteil an Schadsoftware für *Windows Phone*, früher *Windows Mobile*, ist sehr gering, da *Windows* ausschließlich Anwendungen zur Installation zulässt, die aus dem *Windows Phone Store* stammen. Des Weiteren ist der Marktanteil, der im Jahr 2013 bei 3,3% liegt (siehe Abb. 1 auf Seite 10), zu minimal. Der Aufwand um Schadsoftware zu schreiben ist dadurch zu groß, da nur relativ wenig Benutzer damit erreicht werden können.

6.7.1 Schadsoftware für Windows Phone

Durch den Antivirus-Hersteller Trend-Micro wurden in *Windows Mobile* (Vorreiter von *Windows Phone*) Schwachstellen entdeckt, welche anfällig für DoS (Denial of Service) Angriffe sind, die im Internet Explorer von *Windows Mobile 5.0* sowie *Windows Mobile 2003* integriert sind. Durch das Ausnutzen dieser Schwachstelle besteht die Möglichkeit, dass durch einen Speicherüberlauf der Explorer zum Abstürzen gebracht wird, wodurch das System instabil wird und das Gerät neu gestartet werden muss. Wie der Speicherüberlauf verursacht wird, wird durch Trend Micro jedoch nicht bekannt gegeben. Eine weitere Schwachstelle, welche ebenfalls durch einen DoS ausgelöst werden kann, bezieht sich auf die Anwendung „Bilder und Videos“, wobei präparierte Bilder das System so beeinflussen können, dass es ca. eine virtel Stunde lange nicht mehr reagiert, da das System versucht, das präparierte Bild zu verarbeiten. Der Nutzer wird im Unklaren gelassen, warum das Gerät nicht mehr reagiert, da Eingaben in der Zeit nicht möglich sind. Genauso wie bei „Bilder und Videos“ wurde Anfang 2007 von den Sicherheitsleuten der Firma Trend-Micro eine ähnliche Sicherheitslücke in der Anwendung „Resco Photo Viewer“ identifiziert. Es lässt sich durch ein präpariertes .PNG Bild in der Versionen 4.11 und 6.01 ein Speicherüberlauf generieren, der sich ausnutzen lässt, um Schadcode auf dem Gerät zu installieren.[85] Im Jahr 2011 wurde durch die Internetseite Winrumors bei *Windows Phone 7.5 Mango* ein SMS-Bug bekannt, der in der Lage ist, durch eine bestimmte Zeichenfolge in einer SMS den Messaging Hub zum Abstürzen zu bringen. Ebenfalls als Chat-Nachricht oder über den *Windows Live Messenger* kann diese Zeichenfolge Schaden anrichten. Das Mobilgerät kann nur durch komplettes Zurücksetzen auf Werkseinstellung wieder funktionsfähig gemacht werden. [23]

6.8 Schutz und Vermeidung von Schadsoftware

In der Regel ist ein 100% Schutz vor Schadsoftware auf Mobilgeräten nicht möglich, da Schadprogramme, die unter einem vertrauenswürdigem Decknamen in den Stores oder auf Internetseiten zum Download angeboten werden, von Nutzern selbständig installiert werden. Aufgrund dieser Tatsache sollte darauf geachtet werden, dass ausschließlich Anwendungen aus vertrauenswürdigen Quellen installiert werden. Für Anwendungen von unbekannten Herstellern sind Indizien für ein „sauberes“ Programm, das die App eine große Anzahl an Downloads aufweist und eine hohe Anzahl an positiven Kommentaren im App-Store und außerhalb aufweist. Dies gilt hauptsächlich für *Android*-Nutzer, da es ihnen gestattet wird auch Anwendungen von Dritten zu installieren, was für Nutzer von *iOS* und *Windows Phone* nicht zulässig ist. Bevor Apps in den Stores von *Apple* und *Microsoft* veröffentlicht werden, werden diese einem umfangreichen Prüfprozess unterzogen, das jedoch keinen kompletten Schutz vor Schadsoftware bietet, da in den Stores ebenfalls schon Schadsoftware entdeckt wurde.[75] Für mobile Betriebssysteme existieren, genauso

wie für PCs und Notebooks, Antiviren-Programme, die sich aber in der Funktionsweise unterscheiden. Dadurch, dass Anwendungen auf den Mobilgeräten in einer Sandbox ausgeführt werden, besteht für das Antiviren-Programm nicht die Möglichkeit, das System auf Schadsoftware zu untersuchen, da es außerhalb seines Speicherbereichs keine Rechte besitzt. Die Programme arbeiten nach dem Prinzip, dass sie ermitteln, welche Anwendungen insgesamt auf dem mobilen Gerät installiert sind, die daraufhin mit einer Liste von Apps verglichen werden, die vom Hersteller als „schädlich“ eingestuft wurden. Dadurch wird Schadsoftware nicht präventiv erkannt und verhindert, sondern wird in der Regel erst im Nachhinein festgestellt.[80] Im Nachfolgenden werden die Antivirenprogramme für mobile Betriebssysteme aufgeführt, welche laut dem Experten-Portal „www.techfacts.de“, diverse Tests mit der Note „sehr gut“ bestanden haben.

- **F-Secure Mobile Security:** Das Antivirenprogramm ist für die Systeme *Android*, *Symbian* sowie *Windows Phone* ausgelegt. Durch die Anwendung ist es möglich, das Mobilgerät aus der Ferne zu sperren oder sogar zu löschen, wenn z.B. das Gerät gestohlen wurde. Pro Jahr fallen 30 Euro für eine Lizenzgebühr an.
- **Kaspersky Mobile Security Lite:** Bei dieser Anwendung handelt es sich um eine kostenlose Virenschutz-App für das Betriebssystem *Android*. Das Programm warnt bereits beim Download, wenn sich Schadsoftware einschleusen will. Mit der Lite Version kann das Mobilgerät nicht nachträglich auf Schadsoftware gescannt werden, wodurch bereits auf dem Gerät befindliche Schadprogramme mit der Kaspersky Mobile Security Lite App nicht erkannt werden.
- **avast! Mobile Security:** Diese Anwendung ist für das *Android*-System entwickelt, völlig kostenlos und hat einen umfangreichen Funktionsumfang. Die Erkennungsrate ist etwas niedriger als bei den Apps von F-Secure und Kaspersky.
- **AVG Antivirus Free:** Es handelt sich hierbei ebenfalls um eine Anwendung für *Android*-Geräte, die kostenfrei erhältlich ist. Die Erkennungsrate ist nicht so hoch wie bei den vorgenannten Antivirusprogrammen.
- **McAfee Enterprise Mobility Managment:** McAfee ist ein Anbieter, der einen Virenschutz für das *iPhone* in *Apples App Store* anbietet. Bei einem Smartphone mit Jailbreak ist diese Anwendung sinnvoll. Ohne den Jailbreak ist es fraglich, ob so eine Anwendung bei einem geschlossenen System wie bei dem von *Apple* verwendet werden muss.

Am besten kann man sich vor Schadsoftware schützen, indem man regelmäßig prüft, ob neue Firmware-Aktualisierungen vorhanden sind und diese installiert. Außerdem sollte das Laden von Anwendungen aus nicht vertrauenswürdigen Quellen sowie das Öffnen von MMS und SMS von unbekannten Absendern unterlassen werden.[72] Bei Mobilgeräten der Firma *Apple* sollte auf einen Jailbreak verzichtet werden.
Gegen Sicherheitslücken im mobilen Betriebssystem oder in Programmen, wie z.B. Java kann als Nutzer nichts unternommen werden. In diesem Fall müssen sich die Hersteller der Software bemühen diese Lücke zu schließen.

7 Entwicklung einer eigenen Android-Anwendung

Da sich die meiste entdeckte Schadsoftware gegen das *Android*-System richtet, entschied man sich dafür, das Betriebssystem genauer zu untersuchen. In der Regel wird Schadsoftware von den Nutzern auf den Mobilgeräten selbständig installiert, weil Programme angeboten werden die Schadsoftware enthalten, was jedoch nicht sofort ersichtlich ist. Bei *Android*-Geräten müssen bei der Installation einer Anwendung die Berechtigungen, die eine zu installierende Anwendung fordert, von den Nutzern bestätigt werden, wodurch Nutzer die Möglichkeit haben, unerwünschte Eigenschaften einer Anwendung zu erkennen. Da sie die geforderten Berechtigungen einer Anwendung in vielen Fällen nicht korrekt einschätzen können, wird ein Programm entwickelt, das dabei hilft, die Rechte von Apps zu prüfen. Das Programm soll die vergebenen Berechtigungsgruppen einer Anwendung ermitteln und den Nutzern deutlich darstellen, ob Berechtigungen für die Funktion notwendig, bedingt notwendig oder überflüssig sind. Z.B. braucht eine Taschenlampen-App keinen Zugriff auf die Kontaktdaten oder Ortungsdienste, sondern nur auf die Kamera bzw. auf den Kamerablitz. Um eine konkrete Anwendung zu erstellen muss zunächst untersucht werden, in welcher Form die Berechtigungen einer zu installierenden App von den Nutzern bestätigt werden müssen. Hierzu wird überprüft, welche Berechtigungen in dem *Play Store* von *Google* vergeben werden. Nach einem Update der Version des *Play Stores* wurde von *Google* eine sogenannte „einfache Berechtigung“ eingeführt, womit die *Android*-Nutzer nicht mehr einzelne Rechte, sondern Berechtigungsgruppen bei der Installation einer App bestätigen müssen. Dabei wurden von *Google* alle Rechte, die im *Android-Manifest* vergeben werden können in 16 unterschiedliche Gruppen eingeteilt. Bei der Installation einer Anwendung aus dem *Play Store* werden den Nutzern die benötigen Berechtigungsgruppen angezeigt, die die jeweilige App fordert. Im Nachfolgenden sind die Gruppen aufgeführt, welche von *Google* angelegt wurden:

- **In-App Käufe:** Dies bedeutet, dass innerhalb einer App zum Kauf einer anderen App oder zum Kauf von Zusatzpaketen aufgefordert werden kann.
- **Geräte- und App-Verlauf:** Mit dieser Berechtigung darf eine App vertrauliche Protokolldaten lesen, den Systeminternen Status abrufen, Lesezeichen für Webseiten und das Webprotokoll lesen sowie aktive Apps abrufen.
- **Einstellung für Mobilfunkdaten:** Dadurch darf eine App auf Einstellungen zur Steuerung der mobilen Datenverbindung und auf empfangenden Daten zugreifen.
- **Identität:** Hiermit kann eine App auf Konto- und Profilinformationen des Geräts zugreifen. Es darf außerdem nach vorhanden Konten suchen, die Kontaktkarte lesen, z.B. Name und Kontaktdaten, sowie Kontakte ändern und Konten entfernen oder hinzufügen.
- **Kontakte:** Eine App darf die Kontakte des Geräts verwenden, wodurch sie gelesen oder geändert werden können.
- **Kalender:** Eine App kann die Kalenderinformationen des Geräts verwenden, womit Kalendertermine und vertrauliche Informationen gelesen werden können. Zudem können, ohne das Wissen des Nutzers, Kalendertermine hinzugefügt und geändert oder auch E-Mails versendet werden.

- **Standort:** Hiermit kann eine App den Standort des Geräts bestimmen. Es gibt einen ungefähren Standort (dieser ist netzwerkbasiert), sowie einen genauen Standort (dies erfolgt über GPS). Es darf zudem auf zusätzliche Standortanbieterbefehle zugreifen.
- **SMS:** Eine App mit dieser Berechtigung darf den Textnachrichten-Dienst (SMS) und Multimedia Messaging-Dienst (MMS) verwenden. Hierzu gehört, dass die App SMS sowie MMS lesen, bearbeiten, senden und empfangen kann. Zudem können WAP-Nachrichten empfangen werden.
- **Telefon:** Dies bedeutet, dass eine App die Anruffunktion und die Anrufliste verwenden darf, wodurch Telefonnummern direkt angerufen werden können. Des Weiteren kann die Anrufliste bearbeitet oder gelesen werden und es besteht ebenfalls die Möglichkeit, dass ausgehende Anrufe umgeleitet werden können. Zudem kann der Telefonstatus geändert und ohne das Zutun der Nutzer Anrufe getätigt werden.
- **Fotos/Medien/Dateien:** Mit dieser Berechtigung darf eine App auf die gespeicherten Daten und Dateien zugreifen, womit Speicherinhalte gelesen, geändert oder gelöscht werden können. Es kann zudem der externe Speicher formatiert und bereitgestellt werden.
- **Kamera/Mikrofon:** Hiermit darf eine App auf die Kamera und das Mikrofon zugreifen. Es können dadurch Bilder,Videos und Audio-Dateien aufgezeichnet werden.
- **W-LAN-Verbindungsinformationen:** Eine App kann mit dieser Berechtigung auf die W-LAN-Informationen des Gerätes zugreifen und prüfen, wann das W-LAN eingeschaltet ist, sowie die Namen der angeschlossenen Geräte im W-LAN ermitteln.
- **Informationen zur Bluetooth-Verbindung:** Es können die Bluetooth-Einstellungen auf dem Gerät festgelegt werden. Zudem können Informationen über Geräte abgerufen werden, die sich in Reichweite befinden. Außerdem können Informationen gesendet und empfangen werden.
- **Wearable-Sensordaten/-Aktivitätsdaten:** Hiermit kann eine App auf die Daten von Wearable-Sensoren zugreifen, z.B. Herzfrequenzmessgerät. Zudem können regelmäßig Updates über die physische Aktivität abgerufen werden.
- **Geräte-ID & Anrufinformationen:** Eine App kann auf die Geräte-ID und auf die Telefonnummer zugreifen, wodurch ermittelt werden kann, ob gerade telefoniert wird und mit welcher Nummer.
- **Sonstiges:** Mit dieser Berechtigung kann eine App benutzerdefinierte Einstellungen des Geräteherstellers oder anwendungsspezifische Berechtigungen verwenden. Es können soziale Streams gelesen werden (in einigen sozialen Netzwerken), es kann auch in sozialen Streams geschrieben werden (in einigen sozialen Netzwerken) und es darf auf abonnierte Feeds zugreifen.[26]

Nach der Einführung der „einfachen Berechtigung“ wurde von *Google* nicht bekannt gegeben, wie die einzelnen Rechte, welche im *Android-Manifest* einer Anwendung vergeben werden, in die unterschiedlichen Rechtegruppen eingetragen wurden. Dadurch, dass von den Nutzern nicht mehr einzelne Rechte, sondern Berechtigungsgruppen bestätigt werden, besteht die Möglichkeit, dass sich bereits installierte Apps mit einem Update weitere Rechte aus einer bereits bestätigten Berechtigungsgruppe einholen. [25]

Bei der Installation einer Anwendung wird den Nutzern die Berechtigungsgruppe „Sonstiges“ nie angezeigt, obwohl diese verwendet wird. Es wird als sehr kritisch betrachtet, dass die Gruppe „Sonstiges“ nicht angezeigt wird, da somit von Nutzern verschiedene Berechtigungen aus der Gruppe unwissentlich bestätigt werden. Welche Gruppen von einer Anwendung wirklich gefordert werden, ist unter dem Punkt „Berechtigungsdetails“ zu sehen, der sich im *Play Store* ganz unten befindet.

Im Nachfolgenden werden für die Berechtigungsgruppen folgende Abkürzungen verwendet:

- **In-App Käufe = In-App**
- **Geräte- und App-Verlauf = Verlauf**
- **Einstellung für Mobilfunkdaten = Mobilfunkdaten**
- **Identität = Identität**
- **Kontakte = Kontakte**
- **Kalender = Kalender**
- **Standort = Standort**
- **SMS = SMS**
- **Telefon = Telefon**
- **Fotos/Medien/Dateien = Daten**
- **Kamera/Mikrofon = Kamera**
- **W-LAN-Verbindungsinformationen = W-LAN**
- **Informationen zur Bluetooth-Verbindung = Bluetooth**
- **Wearable-Sensordaten/-Aktivitätsdaten = Sensor**
- **Geräte-ID & Anrufinformationen = Geräte-ID**
- **Sonstiges = Sonstiges**

7.1 Schematischer Ablauf einer Installation bei Android

In den meisten Fällen werden von den Nutzern die Anwendungen aus dem *Google Play Store* heruntergeladen und installiert. Dieser Vorgang wird schematisch in Abbildung 17 dargestellt.

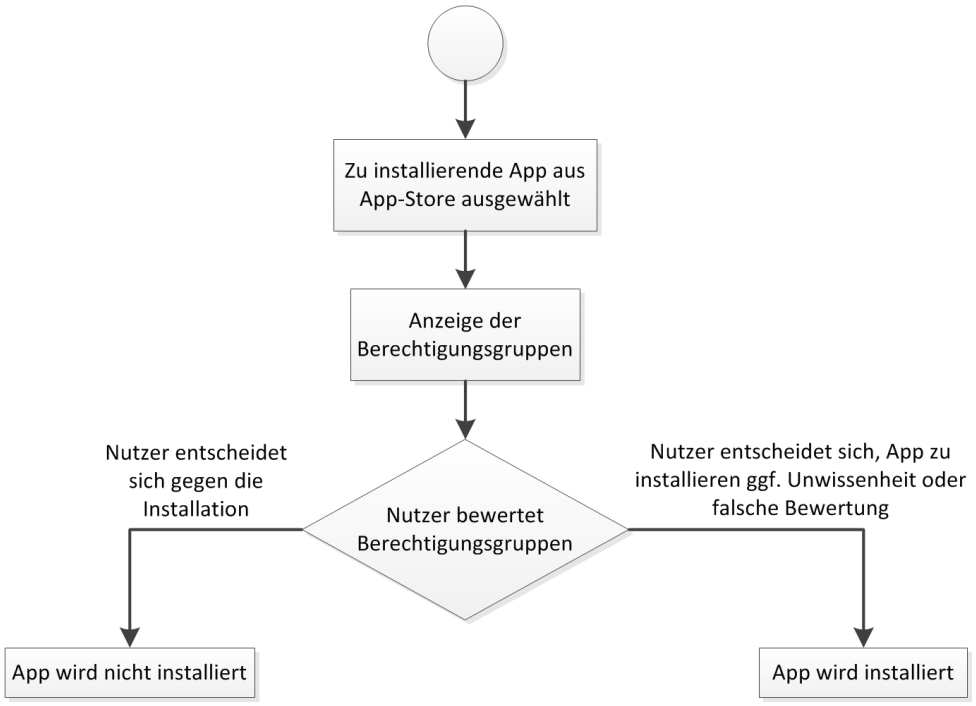

Abb. 17: Installation einer Anwendung

In der Regel werden Anwendungen, welche von Nutzern aus dem *Play Store* ausgewählt wurden, auf dem Gerät installiert. Beim Drücken auf „installieren“ werden den Nutzern die Berechtigungsgruppen angezeigt, welche von der gewählten Anwendung gefordert werden. Daraufhin entscheiden die Nutzer, ob die geforderten Berechtigungsgruppen für die Anwendung „akzeptabel“ sind. Bei dieser Entscheidung wird von den Nutzern der größte Fehler gemacht, da sie meisten nicht korrekt einschätzen können, ob eine Anwendung bestimmte Berechtigungen benötigt, wodurch es passieren kann, dass unwissentlich Schadsoftware auf dem Gerät installiert wird.

Damit Nutzer einen besseren Eindruck über die vergebenen Berechtigungsgruppen einer Anwendung bekommen, wird eine App entwickelt, die es ermöglicht, die vergebenen Berechtigungen besser einschätzen zu können. Dies wird schematisch in Abb. 18 auf Seite 56 dargestellt.

7.2 Schematischer Ablauf zur Bewertung von Berechtigungsgruppen

Durch eine falsche Einschätzung der geforderten Berechtigungsgruppen einer Anwendung besteht, wie schon erwähnt, die Möglichkeit, dass Schadsoftware auf dem Gerät installiert wird. Nach der Installation einer Anwendung kann diese durch eine eigens entwickelte App auf ihre Berechtigungen geprüft werden. Dieses wird in Abbildung 18 dargestellt.

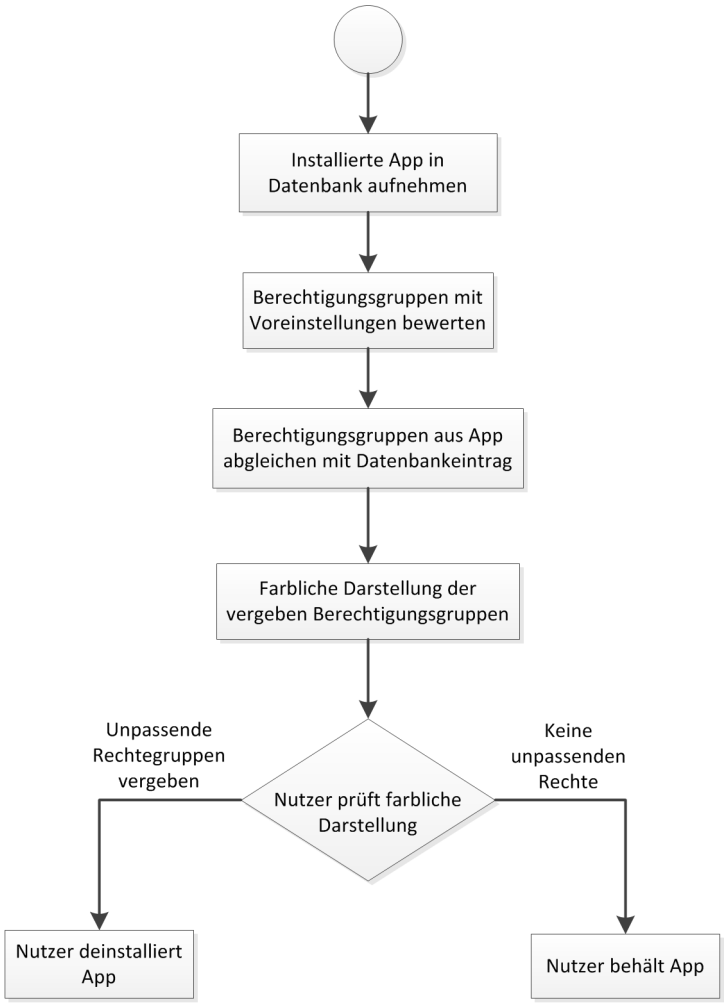

Abb. 18: Prüfen der Berechtigungsgruppen

Für die zu prüfende App muss ein Datenbankeintrag vorgenommen werden, indem alle Berechtigungsgruppen, welche im *Google Play Store* vorhanden sind, bewertet werden müssen, wobei die Bewertung der unterschiedlichen Gruppen über Voreinstellungen realisiert werden kann. Nachdem ein Datenbankeintrag erfolgt ist, können die Nutzer die installierte App prüfen. Es werden dann die vergebenen Berechtigungsgruppen der App, wie sie aus dem *Play Store* bekannt sind, in ihrer Notwendigkeit farblich dargestellt. Dadurch wird den Nutzern ein besserer Eindruck über die vergebenen Berechtigungsgruppen gegeben. Anschließend können die Nutzer selbst entscheiden, ob sie die geprüfte App behalten wollen oder aber, ob sie die App löschen.

7.3 Entwicklung einer Anwendung zur Bewertung von Berechtigungsgruppen

Es wird eine Anwendung entwickelt, die Benutzern bei der Beurteilung von vergebenen Berechtigungen für installierte Anwendungen unterstützt. Bei dem Prüfen einer Anwendung werden die vergebenen Berechtigungsgruppen mit Hilfe eines Ampel-Systems dargestellt. Um das Ampel-System zu realisieren, muss mit der zu prüfenden App ein Eintrag in eine Datenbank erfolgen, in der die Berechtigungsgruppen, welche aus dem *Play Store* bekannt sind, eingestuft werden. Es sollen die Gruppen, die für die Funktion der ausgewählten App notwendig sind, mit „grün“ bewertet werden. Gruppen, die bedingt notwendig sind, mit „gelb“ und Gruppen, die nichts mit der Funktion zu tun haben, mit „rot“. Um die Beurteilung für Nutzer zu vereinfachen, wie wichtig oder unwichtig eine Berechtigungsgruppe für eine Anwendung ist, kann zwischen unterschiedlichen Kategorien gewählt werden. Hierfür stehen folgende Kategorien zur Verfügung: Taschenlampe, Navigation, Kommunikation, Nachschlagewerke, Fotografie und Sonstiges. Bei der Auswahl einer Kategorie werden voreingestellte Werte für die jeweiligen Berechtigungsgruppen geladen. Diese Werte können nachträglich von den Nutzern angepasst werden. Nachdem eine Anwendung in die Datenbank aufgenommen wurde, kann diese geprüft werden. Hierbei werden die Permissions (deutsch: Genehmigung), welche im *Android-Manifest* eingetragen sind, ausgelesen. Die ermittelten Permissions müssen in die Berechtigungsgruppen der „einfachen Berechtigung“ eingetragen werden, damit geprüft werden kann, welche Berechtigungsgruppen von der zu prüfenden Anwendung gefordert werden. Nach der Ermittlung dieser Gruppen wird ein Abgleich mit der Datenbank vorgenommen. Die benötigten Berechtigungsgruppen werden dann in Form eines Ampel-Systems so dargestellt, wie sie in der Datenbank eingetragen wurden.

7.4 Erstellung einer Datenbank für unterschiedlicher Anwendungen

Um eine Datenbank zum Prüfen der Berechtigungsgruppen zu erstellen, werden jeweils zwei Apps für jede Kategorie (Taschenlampe, Navigation, Kommunikation, Nachschlagewerke, Fotografie) aus dem *Play Store* untersucht. Hierbei wird bewusst darauf geachtet, dass in jeder Kategorie eine App vorhanden ist, die nur die Berechtigungen für die eigentliche Funktion einfordert und eine andere App, die mehr Berechtigungen verlangt, als sie für die eigentliche Funktion benötigt. Für die nachfolgende Untersuchung der unterschiedlichen App-Kategorien werden nur die Rechtegruppen dargestellt, welche den Nutzern bei einer Installation einer App aus dem *Play Store* angezeigt werden. Daraus folgt, dass die Berechtigungsgruppe „Sonstiges“ nicht mit aufgeführt ist, da diese nur unter „Berechtigungshinweise“ dargestellt wird. Für die spätere Darstellung durch das Ampel-System wird diese Berechtigungsgruppe natürlich mit berücksichtigt.

7.4.1 Taschenlampen-App

Eine Taschenlampen-App benötigt in der Regel nur Zugriff auf die Kamera bzw. das Blitzlicht. Aus dem *Play Store* wurden zwei Apps ausgewählt, wobei es sich um die Apps „Tiny Flashlight“ und „Brightest Taschenlampe“ handelt. Die nachfolgende Tabelle 3 zeigt eine Übersicht, welche Berechtigungsgruppen für die Installation der jeweiligen App von den Nutzern bestätigt werden müssen:

Name der App	**Kamera**	**Standort**	**Daten**	**W-LAN**	**Geräte-ID**
Tiny Flashlight	ja	-	-	-	-
Brightest Taschenlampe	ja	ja	ja	ja	ja

Tab. 3: Rechte der Taschenlampen-Apps

Aus der Tabelle 3 ist zu entnehmen, dass die Rechtegruppen für „Tiny Flashlight“ optimal sind. Eine App, die als Taschenlampe funktionieren soll, benötigt keine weiteren Rechte bis auf die Berechtigungsgruppe Kamera/Mikrofon, welche hier als Kamera dargestellt ist. Die App „Brightest Taschenlampe“ hat ebenfalls Zugriff auf Kamera/Mikrofon, aber verlangt zudem den Zugriff auf Standort, Fotos/Medien/Dateien, W-LAN-Verbindungsinformationen und den Zugriff auf die Geräte-ID und Anrufinformationen.
Alle Berechtigungsgruppen, bis auf Kamera/Mikrofon könnten dazu dienen, um Nutzerdaten zu sammeln und zu versenden. Über die Geräte-ID können speziell für das verwendete Mobilgerät Informationen gesammelt werden, die dann an Server von Drittanbietern gesendet werden.

7.4.2 Navigations-App

Für die Ortung einer Navigations-App werden in der Regel die Berechtigungsgruppen für den Standort und die W-LAN-Verbindungsinformation benötigt, da mit diesen Gruppen die eigentliche Funktion einer Navigations-App realisiert werden kann. Es wurden die Apps „SNAV Navigator kostenlos“ und „GPS Navigation Sygic“ ausgewählt, welche beide im *Play Store* kostenlos erhältlich sind. In den Tabellen 4 und 5 ist zu sehen, welche Berechtigungsgruppen die jeweilige App bei der Installation fordert:

Name der App	**Standort**	**Daten**	**Kamera**	**W-LAN**	**In-App**
SNAV Navigator kostenlos	ja	ja	ja	ja	-
GPS Navigation Sygic	ja	ja	ja	ja	ja

Tab. 4: Rechte der Navigations-Apps

Name der App	**Verlauf**	**Kontakte**	**Geräte-ID**
SNAV Navigator kostenlos	-	-	-
GPS Navigation Sygic	ja	ja	ja

Tab. 5: Rechte der Navigations-Apps

Aus Tabelle 4 und 5 ist zu entnehmen, dass die Rechte für die App „SNAV Navigation kostenlos“ ausreichend sind. Die Berechtigungsgruppen Standort und W-LAN-Verbindungsinformationen werden für die Ortung benötigt. Die Gruppen Kamera/Mikrofon sowie Fotos/Medien/Dateien werden für Zusatzfunktionen benötigt, um bestimmte Plätze mit

selbst gemachten Fotos von z.B. Sehenswürdigkeiten zu markieren. Bei der App „GPS Navigation Sygic“ müssen diese Berechtigungen ebenfalls bestätigt werden. Zudem werden jedoch die Gruppen für In-App-Käufe, Geräte- und App-Verlauf, Kontakte, Kalender sowie Geräte-ID und Anrufinformationen gefordert. Die Berechtigungen In-App-Käufe und Kontakte werden ebenfalls für Zusatzfunktionen verwendet, wodurch in der App Zusatzfunktionen angeboten werden oder das Navigieren zu einer Person aus der Kontaktliste ermöglicht wird. Die Berechtigungsgruppen für Geräte-ID und Anrufinformationen sowie für Geräte- und App-Verlauf werden für die Funktion nicht benötigt. Es entsteht eher der Eindruck, dass dadurch ein Bewegungsprofil angelegt wird.

7.4.3 Kommunikations-App

Es wurden als Kommunikations-Apps einmal der „Facebook Messanger“ und der Messanger „Threema“ ausgewählt. Bei diesen Apps ist schwer zu unterscheiden, ob sie bestimmte Berechtigungsgruppen für die Funktion benötigen, da in diesen eine große Anzahl an Zusatzfunktionen integriert sind. Sicher ist, dass diese zunächst eine Internetverbindung benötigen, um Texte zu senden und empfangen. Des Weiteren ist es auch möglich Bilder oder den aktuellen Standort zu senden. In den Tabelle 6 und 7 sind beide Messanger mit den jeweiligen Berechtigungsgruppen dargestellt:

Tab. 6: Rechte der Kommunikation-Apps

Name der App	**Identität**	**Kontakte**	**Standort**	**SMS**	**Telefon**
Facebook Messanger	ja	ja	ja	ja	ja
Threema	ja	ja	ja	ja	-

Tab. 7: Rechte der Kommunikation-Apps

Name der App	**Daten**	**Kamera**	**W-LAN**	**Geräte-ID**
Facebook Messanger	ja	ja	ja	ja
Threema	ja	-	-	ja

Es ist zu erkennen, dass der Messanger „Threema “ im Gegensatz zum „Facebook Messanger“ auf die Berechtigungen für Telefon, Kamera/Mikrofon sowie die W-LAN-Verbindungsinformationen verzichtet, wobei beide Messanger den Zugriff auf die Kontakte fordern. Dies wird dafür benötigt, um zu prüfen, welche Einträge in der Kontaktliste ebenfalls den Messanger benutzen. Der Zugriff auf den aktuellen Standort wird dafür verwendet, um den aktuellen Standort einer anderen Person mitteilen zu können. Mit der Berechtigung für Fotos/Medien/Dateien können Bilder oder Dateien an andere Personen gesendet werden. Bei dem „Facebook Messanger“ dient dafür ebenfalls die Berechtigung für Kamera/Mikrofon, da somit ein Bild direkt im Chat gemacht und versendet werden kann. Über die Geräte-ID und Anrufinformationen stellen beide Messanger die Telefonnummer des verwendeten Geräts fest, was allerdings von dem „Facebook Messanger“ nicht benötigt wird, da für die Nutzung ein „Facebook-Account“ notwendig ist. Für die Funktion der jeweiligen Messanger ist der Zugriff auf SMS nicht notwendig. Beide Messanger benötigen nicht die Berechtigung, um SMS zu senden, empfangen oder die SMS-Einträge lesen zu können. Für den „Facebook Messanger“ ist es ebenfalls nicht notwendig, dass dieser die Berechtigungsgruppe Telefon hat. Der Messanger ist damit in der Lage, selbständig Anrufe zu tätigen oder umzuleiten.

7.4.4 Nachschlagewerk-App

Als Nachschlagewerke wurden die Apps „Wikipedia“ und „Kiwix“ ausgewählt, die in der Regel keine Berechtigungen benötigen, da es sich jeweils um ein „offline“-Nachschlagewerk (benötigen keine Internetverbindung) handelt. Obwohl diese Anwendungen keine Berechtigung benötigen, fordern diese dennoch welche. Diese sind in der Tabelle 8 zu entnehmen:

Name der App	**Standort**	**Daten**
Wikipedia	ja	-
Kiwix, Wikipedia ohne Internet	-	ja

Tab. 8: Rechte der Nachschlagewerk-Apps

Aus der Tabelle 8 ist zu entnehmen, dass „Wikipedia“ den Zugriff auf den Standort des Gerätes verwenden will. Wofür diese Berechtigungsgruppe bei der App benötigt wird, ist nicht bekannt. Es kann dazu dienen, um gezielt Werbung für die Nutzer anzuzeigen. Weitere Rechte verlangt diese App nicht. Die „Kiwix-App“ fordert den Zugriff auf Fotos/Medien/Dateien. Wofür diese Berechtigung dient, ist ebenfalls nicht bekannt. Da diese beiden Apps „offline“ funktionieren und nur Daten nachgesehen werden, benötigen diese somit gar keine Berechtigungen.

7.4.5 Fotografie-App

Bei den Apps für Fotografie handelt es sich um Programme, mit denen Bilder bearbeitet werden können. Hierzu wurden die Apps „Photo Grid - Collage Maker“ und „Camera 360 Ultimate“ gewählt. Diese Apps sollten in der Regel Zugriff auf die Kamerafunktion haben, da somit aktuelle Aufnahmen verarbeitet werden können. Zudem sollte die Berechtigung vorhanden sein, dass auf die vorhanden Bilder zugegriffen werden kann. Dies geschieht über die Berechtigung von Fotos/Medien/Dateien. In den nachfolgenden Tabellen 9 und 10 sind die jeweiligen Rechtegruppen dargestellt, die bei der Installation der Anwendungen von den Nutzern bestätigt werden müssen:

Tab. 9: Rechte der Fotografie-Apps

Name der App	**In-App**	**Verlauf**	**Daten**	**W-LAN**
Photo Grid Collage Maker	ja	ja	ja	ja
Camera 360 Ultimate	-	ja	ja	ja

Tab. 10: Rechte der Fotografie-Apps

Name der App	**Standort**	**Kamera**	**Geräte-ID**
Photo Grid Collage Maker	-	ja	-
Camera 360 Ultimate	ja	ja	ja

Bei „Photo Grid - Collage Maker“ ist zu entnehmen, dass diese App die Berechtigung für In-App-Käufe verlangt. Dieses wird benötigt, um Zusatzpakete in der App anbieten zu können. Zudem fordert die App den Zugriff auf Geräte-und App-Verlauf sowie W-LAN-Verbindungsinformationen an. Diese beiden Berechtigungen werden für die Funktion nicht benötigt. Die App „Camera 360 Ultimate“ benötigt mehr Berechtigungen als „Photo Grid - Collage Maker“. Es wird auf die In-App-Käufe verzichtet, dafür fordert die App jedoch

die Berechtigungsgruppen Standort und Geräte-ID und Anrufinformationen an. Hiermit könnte zu einem gemachten Foto der Standort mit abgespeichert wird. Es besteht dadurch allerdings auch die Möglichkeit, dass durch die App ein Bewegungsprofil angelegt wird.

7.5 Zuordnung der Rechte

In diesem Abschnitt werden die Berechtigungsgruppen der jeweiligen App-Kategorien eingeordnet, welche in der Datenbank für Voreinstellungen benötigt werden, um die Bewertung der Gruppen den Nutzern zu erleichtern. Hierbei werden Rechtegruppen mit „grün“ deklariert, die für die Funktion der App notwendig sind. Mit „gelb“ werden Gruppen eingestuft, die nicht zwangsläufig etwas mit der Funktion zu tun haben aber ggf. für Zusatzfunktionen verwendet werden. Mit „rot“ werden die Berechtigungsgruppen eingestuft, die nichts mit der Funktion der App zu tun haben.

Taschenlampen-App: Da die Funktion der Taschenlampe mit dem Zugriff auf Kamera/Mikrofon komplett realisiert werden kann, wird dieser Punkt mit „grün“ gewertet. Die anderen Berechtigungen werden mit „rot“ gewertet, da sie nicht für die Funktion benötigt werden.

Navigations-App: Damit eine App navigieren kann, wird der Zugriff auf die Kategorien Standort und W-LAN-Verbindungsinformationen benötigt, weshalb diese mit „grün“ gewertet werden. Der Zugriff auf die Fotos/Medien/Dateien, Kamera/Mikrofon, In-App-Käufe, Kontakte und Kalender werden mit „gelb“ gewertet, da diese Berechtigungen für Zusatzfunktionen verwendet werden, was z.B. das Navigieren zu einer Person aus den Kontaktdaten sein kann. Über In-App-Käufe können Zusatzpakete zu der jeweiligen App angeboten werden. Der Zugriff auf Geräte-ID und Anrufinformationen und Geräte- und App-Verlauf werden als „rot“ gewertet, da diese für die Funktion nicht benötigt werden.

Kommunikations-App: Zur Kommunikation über einen Messanger wird eine Internetverbindung benötigt, was über eine mobile Datenverbindung oder über das W-LAN geschehen kann. Da sich der Punkt für „vollen Internetzugriff“ in der Kategorie „Sonstiges“ befindet (mehr dazu unter „Untersuchung der Berechtigungsgruppen aus dem *Play Store*“ auf Seite 63), wird nur diese mit „grün“ bewertet. Der Zugriff auf Kontakte, Kalender, Identität, Standort, Fotos/Medien/Dateien, Geräte-ID und Anrufinformationen, W-LAN-Verbindungsinformationen und die Kamera/Mikrofon wird mit „gelb“ gewertet. Diese Berechtigungsgruppen können in den Messangern für Zusatzfunktionen benötigt werden, um z.B. jemand den aktuellen Standort zu senden oder ein Foto zu schicken. Die Gruppen SMS und Telefon werden mit „rot“ eingestuft, da ein Messanger die Funktionen für SMS lesen, senden und empfangen, oder selbständig Nummern anrufen, nicht benötigt.

Nachschlagewerk-App: Apps die als „offline“-Nachschlagewerk dienen, benötigen in der Regel für ihre Funktion keine Berechtigungen. Dadurch werden alle Gruppen, die solch eine App fordert, mit ”rot“ deklariert.

Fotographie-App: Apps, mit denen Bilder bearbeitet werden können, benötigen den Zugriff auf die Kamera/Mikrofon sowie Fotos/Medien/Dateien. Somit werden diese Berechtigungsgruppen mit „grün“ festgelegt. In-App-Käufe würden mit „gelb“ gewertet werden, da diese dazu dienen, um Zusatzpakete zu der App zu erwerben. Der Zugriff auf W-LAN-Verbindungsinformationen, Standort, Geräte-ID und Anrufinformationen sowie

Geräte-und App-Verlauf werden als ”rot“ gewertet, da diese nichts mit der eigentlichen Funktion zu tun haben und ggf. zur Spionage oder zum Anlegen eines Bewegungsprofils genutzt werden könnten.

7.6 Theoretischer Aufbau der Datenbank

Der Aufbau der Datenbank erfolgt über die jeweiligen Kategorien, die im Vorfeld festgelegt wurden, für die die verschiedenen Berechtigungsgruppen bewertet und angelegt werden. Um dieses darzustellen, müssen alle Berechtigungsgruppen beachtet werden, die bei *Android* vergeben werden können. In den Tabellen 11, 12 sowie 13 wird „grün“ mit einer „1“ angegeben. Das bedeutet, dass diese Rechte für die eigentlich Funktion der App benötigt werden. Für Berechtigungsgruppen, welche bedingt für die Funktion einer App notwendig sind, d.h. für Zusatzfunktionen, wird eine „2“ eingetragen. Rechte, die nicht für die Funktion einer Anwendung notwendig sind, werden mit „rot“ gekennzeichnet. Es wird in das Feld eine „3“ eingetragen.

Kategorie	Identität	Kamera	SMS	Kontakt	Kalender	Geräte-ID
Taschenlampe	3	1	3	3	3	3
Navigation	3	2	3	2	3	3
Kommunikation	2	2	3	2	3	2
Nachschlagewerk	3	3	3	3	3	3
Fotografie	3	1	3	3	3	3

Tab. 11: Rechte der Kategorien

Kategorie	W-LAN	In-App	Verlauf	Mobilfunk	Telefon	Bluetooth
Taschenlampe	3	3	3	3	3	3
Navigation	1	2	3	3	3	3
Kommunikation	2	2	3	3	3	3
Nachschlagewerk	3	3	3	3	3	3
Fotografie	3	2	3	3	3	3

Tab. 12: Rechte der Kategorien

Kategorie	Standort	Dateien	Sensor	Sonstiges
Taschenlampe	3	3	3	3
Navigation	1	2	3	3
Kommunikation	2	2	3	1
Nachschlagewerk	3	3	3	3
Fotografie	3	1	3	3

Tab. 13: Rechte der Kategorien

Die Einträge, welche mit einer „2“ bewertet wurden, werden später im entwickelten Programm mit „gelb“ dargestellt, worauf in den Tabellen verzichtet wurde, da es schlecht lesbar ist.

Nach der Installation einer App wird mit Hilfe der Datenbank geprüft, welche Berechtigungsgruppen für eine Anwendung benötigt, welche bedingt benötigt und welche absolut nicht benötigt werden. Um einen Abgleich mit der Datenbank realisieren zu können, müssen für jede App die Rechte aus dem *Android-Manifest* ausgelesen und in die unterschiedlichen Berechtigungsgruppen eingetragen werden. Die ermittelten Rechtegruppen werden dann mit der Datenbank aus Tabelle 11, 12 und 13 verglichen. Wenn z.B. eine Navigations-App die Berechtigungsgruppe für SMS gefordert hat, würde in der Kategorie Navigation geprüft werden, was an der Stelle eingetragen ist. Die Vergabe der Gruppe für SMS würde in diesem Fall „rot“ hinterlegt sein, da sich dort eine „3“ befindet. In der Darstellung sollte dann ebenfalls der Punkt „SMS“ mit „rot“ markiert sein.

7.7 Untersuchung der Berechtigungsgruppen aus dem Play Store

Um die Einteilung korrekt zu bestimmen, wurde für jede untersuchte App der Abgleich mit den Einträgen des *Android-Manifests* und der Anzeige der Gruppen aus den Berechtigungsdetails des *Play Stores* vorgenommen, da die Zuteilung der Rechte in die Berechtigungsgruppen von *Google* nicht bekannt gegeben wird. In dieser Gruppierung wurden nur die Rechte betrachtet, welche in dem *Android-Manifest* der untersuchten Apps vergeben wurden. Es ergibt sich folgende Einteilung:

- **Identität:** GET_ACCOUNTS, READ_PROFIL
- **Kamera/Mikrofon:** CAMERA, FLASHLIGHT, RECORD_VIDEO, RECORD_AUDIO
- **SMS:** RECEIVE_MMS, READ_SMS, WRITE_SMS, SEND_SMS, RECEIVE_SMS
- **Kontakte:** READ_CONTACTS
- **Geräte-ID und Anrufinformationen:** READ_PHONE_STATE, READ_SETTINGS, READ_CALL_LOG, READ_LOGS
- **W-LAN Verbindungsinformationen:** ACCESS_WIFI_STATE
- **In-App-Käufe:** BILLING
- **Geräte- und App-Verlauf:** GET_TASKS, CHECK_LICENSE
- **Telefon:** CALL_PHONE
- **Standort:** ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_LOCATION_EXTRA_COMMANDS
- **Fotos/Medien/Dateien:** READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE, MOUNT_UNMOUNT_FILESYSTEMS
- **Sonstiges:** RECEIVE_BOOT_COMPLETED, WAKE_LOCK, VIBRATE, INTERNET, ACCESS_NETWORK_STATE, CONTROL_LIGHT, STATUS_BAR, INSTALL_SHORTCUT, UNINSTALL_SHORTCUT, CHANGE_WIFI_STATE, MODIFY_AUDIO_SETTINGS, BROADCAST_STICKY, BROADCAST_BADGE, WRITE_BADGES, CROSS_PROCESS_BROADCAST, BATTERY_STATS, RECEIVE, C2D_MESSAGE, CHECK_LICENSE, SYSTEM_ALERT_WINDOW, WRITE, READ

Aus dieser Gruppierung ist gut zu erkennen, dass die meisten vergebenen Rechte von *Google* unter dem Punkt „Sonstiges“ eingefügt wurden. Die Berechtigungskategorie „Sonstiges“ beinhaltet alle Rechte, die nicht in die anderen Rechtegruppen passen. Diese Kategorie müsste weiter aufgeteilt werden, damit die Rechtevergabe an Apps besser beurteilt werden kann. Zudem ist der Punkt „Sonstiges“ der einzige Punkt, der bei der Installation einer App aus dem *Play Store* nicht angezeigt wird. Dieser ist nur ersichtlich unter den Berechtigungsdetails im *Play Store*. Am Beispiel der Taschenlampen-App „Tiny Flashlight“ soll dieses verdeutlicht werden.
Nach der Betätigung des Buttons „Installieren“ im *Play Store* erscheint für die Nutzer das Fenster aus Abbildung 19 mit den benötigten Berechtigungsgruppen für diese App, welche für die Installation akzeptiert werden müssen. Die Rechtekategorie „Kamera“ macht für die Funktion der Taschenlampe Sinn, da die App auf den Blitz der Kamera zugreifen muss, um die Funktionalität sicher zu stellen. Durch die Darstellung der Rechtekategorie in Abbildung 19 wird den Nutzern vermittelt, dass diese Anwendung nur die Berechtigung für den Zugriff auf die Kamera erhält. Mit dem Akzeptieren dieser Rechtegruppe, wird die App daraufhin auf dem Gerät installiert.

Abb. 19: Anzeige bei Installation

Abb. 20: Anzeige Berechtigungshinweises

In der Abbildung 20 wird das Fenster dargestellt, welches im *Play Store* unter dem Punkt „Berechtigungshinweis“ zu finden ist. Es ist etwas detaillierter beschrieben, welche Rechte der Taschenlampen-App „Taschenlampe Tiny Flashlight“ wirklich zugeteilt werden. Aus dieser Abbildung ist zu entnehmen, dass nicht nur Zugriff auf die Kamera besteht, sondern auch auf weitere Rechte aus der Kategorie „Sonstiges“. Die Vergabe von den Rechten aus der Kategorie „Sonstiges“ wurde den Nutzern bei der Installation nicht mitgeteilt und würde somit, unwissentlich, ebenfalls akzeptiert werden. Dadurch, dass diese Berechtigungsgruppe nicht angezeigt wird, wird den Nutzern bei der Installation einer Anwendung, eine scheinbare Sicherheit vermittelt. Um zu Überprüfen, welche Rechte im *Android-Manifest* der Taschenlampen-App wirklich vergeben wurden, wurden diese ausgelesen.
In Abbildung 21 sind die Permissions aus dem *Android-Manifest* der Taschenlampen-App dargestellt. Aus dieser Abbildung ist deutlich zu entnehmen, dass die Rechte, welche unter

„Berechtigungshinweise“, siehe Abb. 20, im *Manifest* der App vergeben wurden. Für diese App ist die Aufteilung der Rechte in die Berechtigungskategorien wie folgt:

Kamera: CAMERA, FLASHLIGHT

Sonstiges: RECEIVE_BOOT_COMPLETED, WAKE_LOCK, VIBRATE, INTERNET, ACCESS_NETWORK_STATE, CONTROL_LIGHT

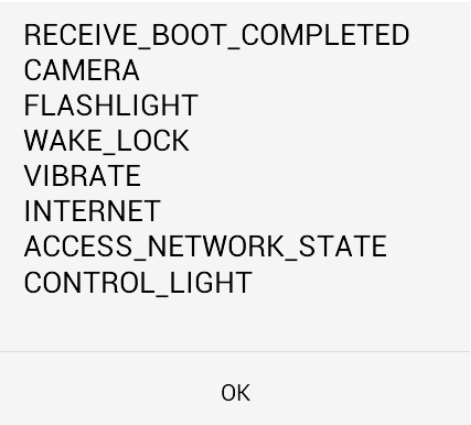

Abb. 21: Rechte aus *Manifest*

Inwieweit die vergebenen Rechte aus dem *Android-Manifest* von der App genutzt werden, ist nicht bekannt.
Nach der Installation dieser App wurden von den Nutzern somit die Berechtigungsgruppen „Kamera“ und „Sonstiges“ akzeptiert. Bei einem Update für diese Anwendung besteht die Möglichkeit, dass die App sich weitere Rechte aus den bestätigten Gruppen sichert, ohne dass dies die Nutzer mit bekommen[1]. Dieses automatische Hinzufügen von weiteren Rechten durch ein Update bestätigt der Text, welche in Abb. 20 ganz unten steht. Der Grund dafür ist, dass nicht mehr einzelne Rechte, sondern Berechtigungsgruppen bestätigt werden.

7.8 Umsetzung für die Entwicklung einer Anwendung zur farblichen Darstellung der Berechtigungsgruppen

Für die Umsetzung zur farblichen Darstellung der Berechtigungsgruppen einer App wurde mit Eclipse und dem *Android*-SDK eine Anwendung für *Android*-Betriebssysteme entwickelt. Mit dieser Anwendung ist es möglich, installierte Apps auf ihre Rechte zu prüfen. Um eine App zu prüfen, muss hierfür zunächst ein Datenbankeintrag angelegt werden, wobei jede Berechtigungsgruppe von *Google* bewertet wird. Es muss eingetragen werden, ob die zu prüfende App die unterschiedlichen Berechtigungsgruppen für die eigentliche Funktion benötigt, bedingt benötigt oder absolut nicht benötigt. Für die Bewertung gilt, eine „1“ wird mit „grün“ dargestellt und bedeutet, dass die Berechtigungsgruppe für die Funktion benötigt wird. Eine „2“ wird mit „gelb“ dargestellt und bedeutet, dass die Berechtigungsgruppe bedingt benötigt wird. Mit einer „3“ soll ausgesagt werden, dass die Berechtigungsgruppe absolut nicht für die eigentliche Funktion benötigt wird. Dieses wird mit „rot“ markiert. Um diese Bewertung etwas einfacher zu gestalten wurden unterschiedliche Kategorien angelegt, welche Taschenlampe, Navigation, Kommunikation, Nachschlagewerke, Fotografie und Sonstiges sind. Mit der Auswahl einer Kategorie werden Voreinstellungen für die Bewertung der Berechtigungsgruppen geladen, die unter „Theoretischer Aufbau der Datenbank“ auf Seite 62 angelegt wurden. Diese können nachträglich vom Benutzer angepasst werden. Diese Applikation ist ab einem API-Level 8 auf *Android*-Geräten nutzbar. D.h. es muss mindestens die *Android*-Version 2.2.x *Froyo* installiert sein. Damit diese App eine Datenbank anlegen und verwalten kann, wurden dieser die Berechtigung zum Lesen und Schreiben auf dem Speicher erteilt. In der nachfolgenden Abbildung 22 ist der Aufbau des Projekts dargestellt. Die sichtbaren Klassen (Activities) werden in grün dargestellt. Eine weitere Klasse, welche nicht sichtbar ist, wird in gelb angezeigt.

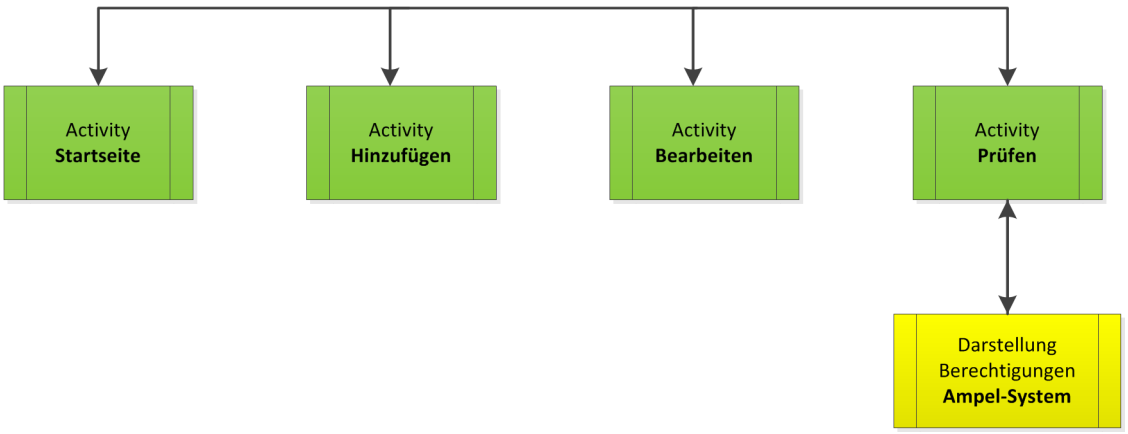

Abb. 22: Darstellung der Activities

Für die Umsetzung der Anwendung wurden vier Activities benötigt, die sich alle auf einer Ebene befinden. Es existiert die *Activity Startseite*, die angezeigt wird, wenn die entwickelte App gestartet wird. Durch die *Activity Hinzufügen* können installierte Anwendungen in die Datenbank aufgenommen werden. Über die *Activity Bearbeiten* können bestehende Einträge der Datenbank geändert werden. Mit der *Activity Prüfen* können installierte Apps auf ihre Berechtigungsgruppen untersucht werden. Von der Klasse *Darstellung Berechtigungen* wird ein Fenster erzeugt, indem die Berechtigungsgruppen einer App mit den Einstellungen der Datenbank, in Form eines Ampel-Systems, angezeigt werden.

7.9 Activity Startseite

Nach dem Start der erstellten Anwendung zum Prüfen von Apps wird die *Activity Startseite*, siehe Abb. 23, dargestellt. Von dieser Seite aus besteht die Möglichkeit, zu allen anderen Activities dieser App zu wechseln.
Über den Button „hinzufügen“ gelangt man zu der *Activity Hinzufügen* (siehe Abb. 25, Seite 69), um eine neue App in die Datenbank aufzunehmen. Wenn bereits eine Datenbank vorhanden ist, können diese Einträge nachträglich bearbeitet werden, indem der Button „bearbeiten“ betätigt wird, womit die *Activity Bearbeiten*, siehe 30, Seite 74, aufgerufen und dargestellt wird. Nachdem installierte Applikationen in die Datenbank aufgenommen wurden, können diese geprüft werden. Dieses geschieht über den Button „prüfen“, womit man zur *Activity Prüfen* aus Abb. 34, Seite 78 wechselt. Über den Button „löschen“ kann die bereits vorhanden Datenbank komplett gelöscht werden. Bei Betätigung erscheint noch eine Sicherheitsabfrage, ob man sich wirklich sicher ist, die Datenbank zu löschen.

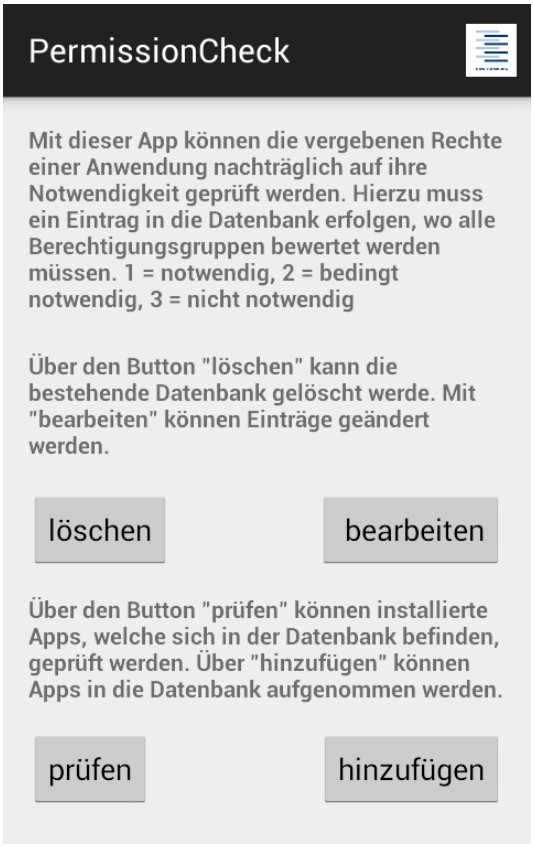

Abb. 23: *Activity Startseite*

7.9.1 Schematischer Ablauf der Activity Startseite

In der Abbildung 24 auf Seite 68 ist der Ablauf der Startseite schematisch dargestellt, indem zu erkennen ist, dass nach dem Start des Programms die verschiedenen Buttons initialisiert und so eingestellt werden, dass diese auf das Event *onClickListener()*, welches in der Abbildung „blau“ dargestellt ist, reagieren. Zudem wird der Text, der in Abbildung 23 zu sehen ist, angezeigt. Wenn ein Button betätigt wird, wird in der Methode *onClickListener()* geprüft, welcher gedrückt wurde. Durch die Betätigung der Buttons werden die unterschiedlichen Activities aufgerufen oder es wird nach einer Sicherheitsabfrage die bereits bestehende Datenbank gelöscht.

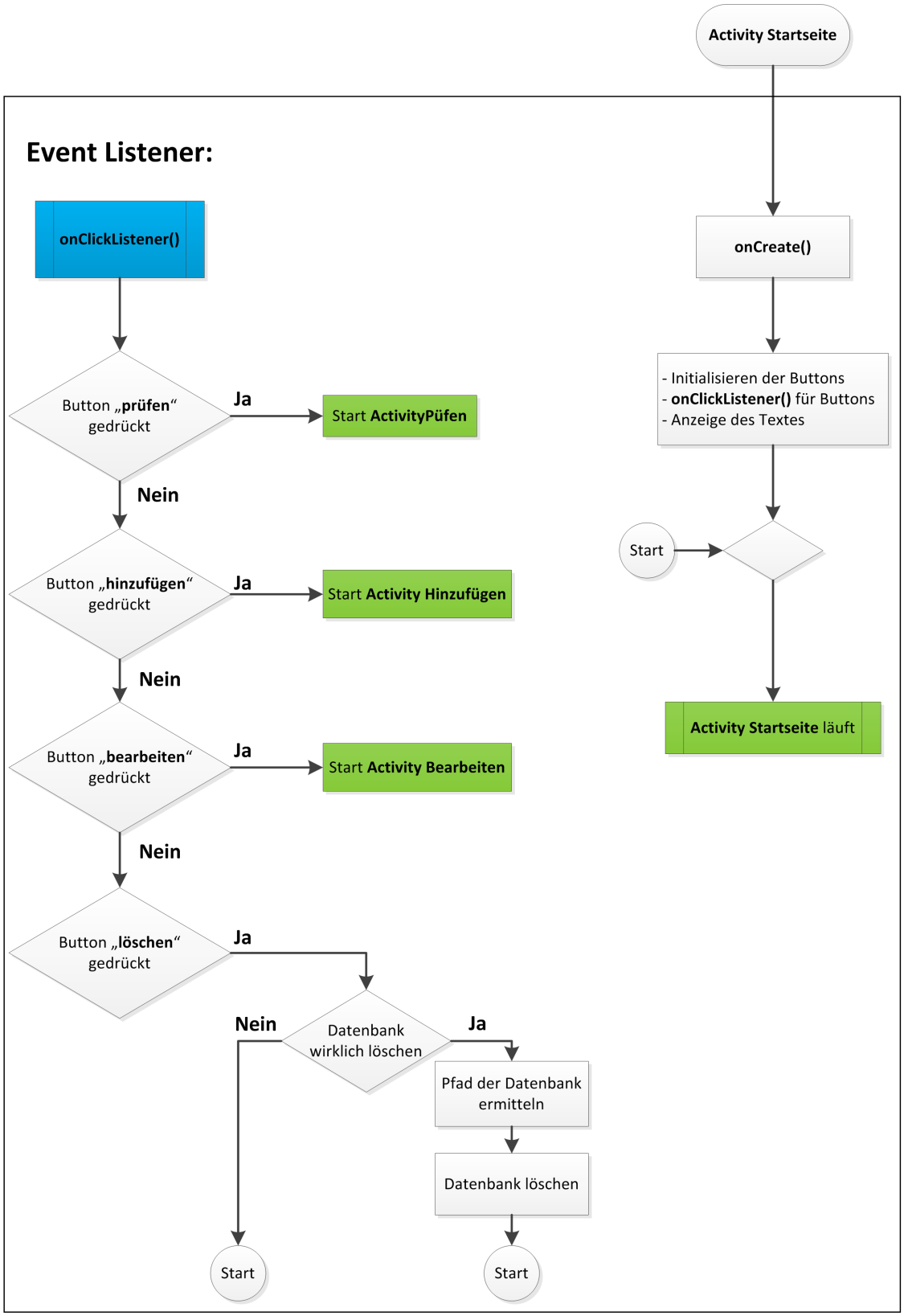

Abb. 24: Ablaufdiagramm der *Activity Startseite*

7.10 Activity Hinzufügen

Über diese Activity können installierte Apps in die Datenbank aufgenommen werden. Durch das obere Drop-Down-Menü kann eine App, die auf dem Mobilgerät installiert ist, ausgewählt werden. Es wurde in Abbildung 25 „Navigation Free“ ausgewählt, wobei es sich um die Anwendung „SNAV Navigator kostenlos“ handelt. Es werden die App-Namen angezeigt, wie sie im *Android*-System unter Einstellungen -> Anwendungsmanager dargestellt werden. Bei dem unteren Drop-Down-Menü kann eine Kategorie ausgewählt werden, die hier auf „Navigation“ gesetzt wurde, da es sich um eine Navigationsanwendung handelt. Durch die Auswahl werden Voreinstellungen zur Bewertung der Berechtigungsgruppen festgelegt, welche von den Nutzern nachträglich geändert werden können. Für diese Anwendung sind die Punkte „Standort“ und „W-LAN-Verbindung“ wichtig, da über die beiden Berechtigungen der aktuelle Standort ermittelt werden kann. Die Gruppen, die „gelb“ gekennzeichnet sind, sind für eine Navigations-App und deren Funktion bedingt erforderlich. Es besteht die Möglichkeit, dass z.B. an bestimmten Standorten Fotos oder Sprachnachrichten verknüpft werden. Des Weiteren kann durch die App die Navigation zu Personen aus der Kontaktliste erfolgen. Aufgrund dieser Tatsache sind „Kamera/Mikrofon“, „Kontakte“, „Medien/Dateien“ mit „gelb“ markiert. Der Punkt „In-App-Käufe“ kann dazu dienen, den Nutzern in der Anwendung Zusatzprodukte anzubieten. Die restlichen Berechtigungen sind „rot“ gekennzeichnet, da sie für die Funktion nicht notwendig sind. Über den Button „hinzufügen“ wird zunächst überprüft, ob bereits ein Eintrag für diese App in der Datenbank vorhanden ist. Sollte dies der Fall sein, dann kann dieser Eintrag bearbeitet werden. Wenn kein Eintrag mit der gewählten App in der Datenbank besteht, wird diese hinzugefügt. Über „abbrechen“ werden die Einstellungen verworfen und es wird zur Startseite gewechselt.

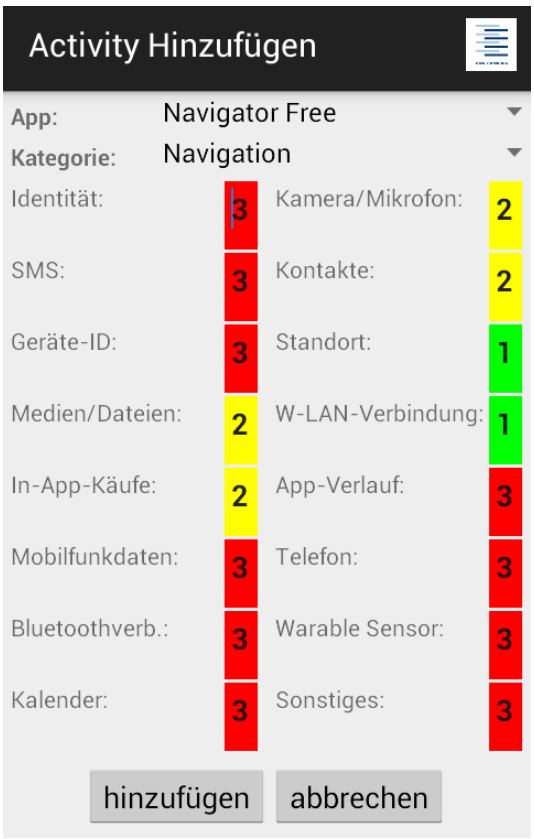

Abb. 25: *Activity Hinzufügen*

7.10.1 Schematischer Ablauf der Activity Hinzufügen

In Abbildung 26 ist der Ablauf der *Activity Hinzufügen* dargestellt. Diese Activity beinhalten die Events *onItemSelected()*, *onClickListener()* und *TextWatcher()*, die beim Start initialisiert werden. Zudem wird das Drop-Down-Menü mit den installierten Apps sowie das Drop-Down-Menü mit den Kategorien gefüllt. Es wird geprüft, ob die Activity von der *Activity Prüfen* aufgerufen wurde. Dies ist der Fall, wenn eine App geprüft werden soll, die sich noch nicht in der Datenbank befindet. Hierbei wird beim Aufruf der *Activity Hinzufügen* der Index für das Drop-Down-Menü der gewählten App übergeben. Somit kann der Eintrag des Menüs auf die App gesetzt werden, welche geprüft werden soll.

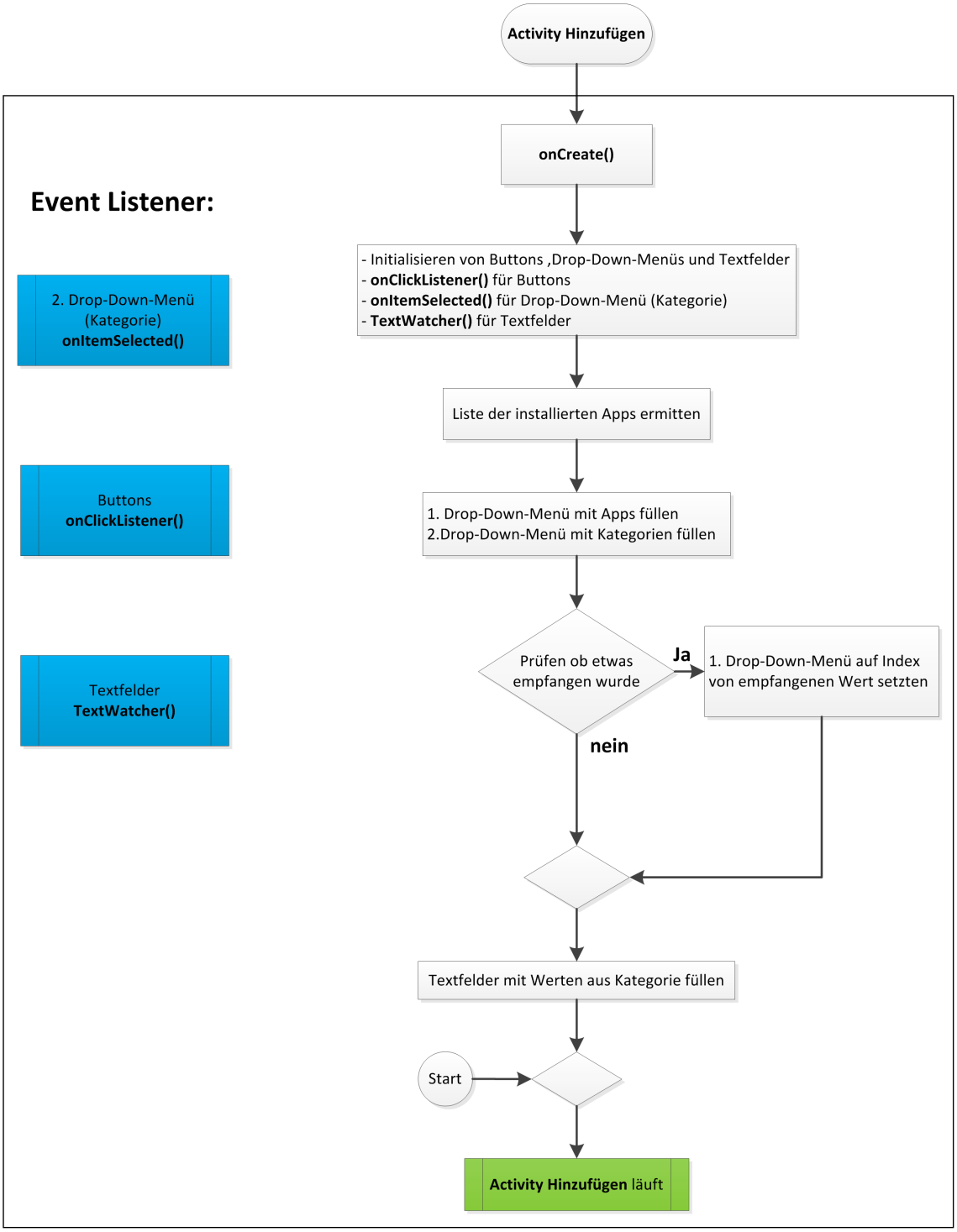

Abb. 26: Schematischer Ablauf der *Activity Hinzufügen*

Die Buttons „hinzufügen“ und „abbrechen“, welche in 25 zu sehen sind, rufen bei Betätigung die Methode *onClickListener()* auf. In der Methode wird nach Betätigung geprüft, welcher der Buttons „gedrückt“ wurde. Beim Button „abbrechen“ wird zur Startseite gewechselt. Bei Betätigung des Buttons „hinzufügen“ werden die Einträge aus den beiden Drop-Down-Menüs und die Einträge aus den Textfeldern entnommen und zu einem Datensatz zusammengefasst.

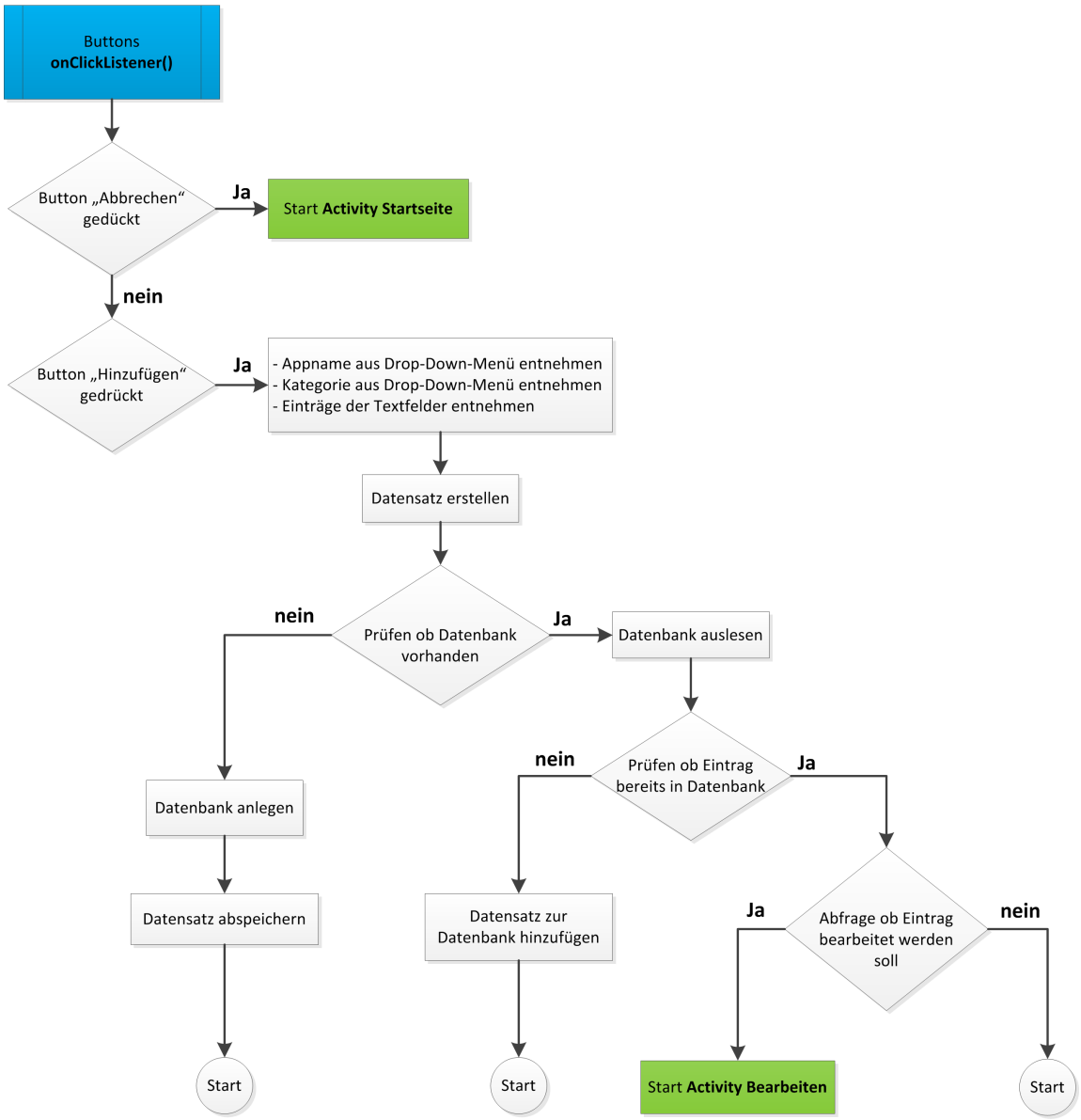

Abb. 27: Schematischer Ablauf der *Activity Hinzufügen onClickListener()*

Daraufhin wird geprüft, ob eine Datenbank bereits existiert. Wenn keine vorhanden ist, wird eine neue angelegt und der Datensatz wird gespeichert. Ist eine Datenbank bereits vorhanden, wird geprüft, ob ein Eintrag mit der gewählten App bereits vorhanden ist. Wenn ja, kann dieser Eintrag bearbeitet werden, indem die *Activity Bearbeiten* aufgerufen wird. Wenn keine Eintrag mit der gewählten App besteht, wird der erstellte Datensatz zu der bestehenden Datenbank hinzugefügt.

Das Menü, welches die unterschiedlichen Kategorien beinhaltet, reagiert auf eine Indexänderung des Drop-Down-Menüs, wodurch die Methode *onItemSelected()* aufgerufen wird. Beim Aufruf der Methode wird geprüft, welchen neuen Index das Drop-Down-Menü hat. Daraufhin werden Voreinstellungen für die jeweilige Kategorie geladen.

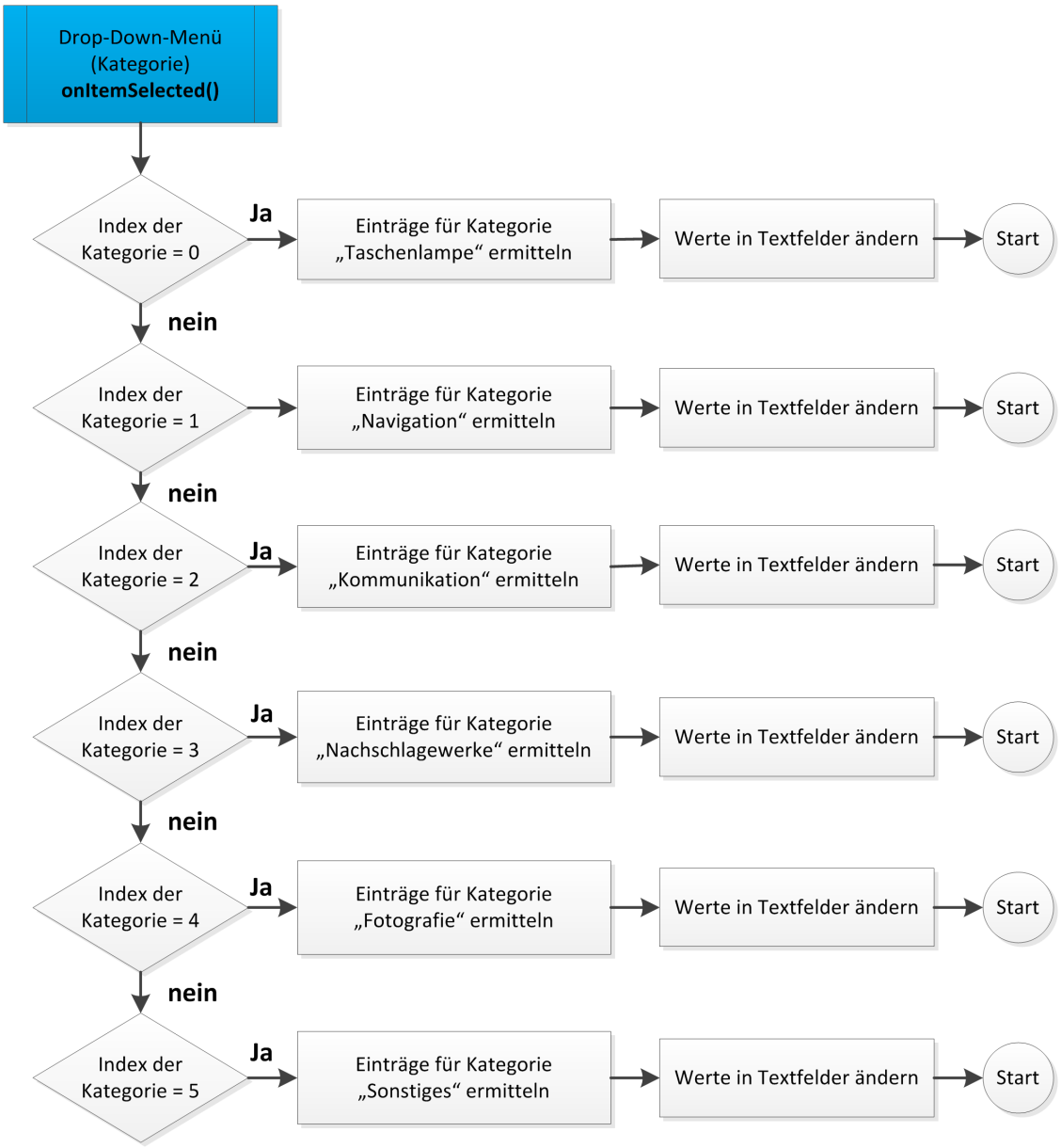

Abb. 28: Schematischer Ablauf der *Activity Hinzufügen onItemSelected()*

Wenn z.B. der Index für das Menü auf 0 steht, wurde die Kategorie „Taschenlampe“ ausgewählt. In diesem Fall werden die Voreinstellungen für die Taschenlampen-App geladen und in den Textfeldern der unterschiedlichen Berechtigungsgruppen angezeigt.

Bei Änderung der Bewertung für die jeweiligen Berechtigungsgruppen wird die Methode *TextWatcher()* aufgerufen. In dieser Methode wird geprüft, ob der Eintrag zwischen 1 - 3 liegt. Ist dies nicht der Fall, erscheint ein Text, dass es kein korrekter Eintrag ist. In das Feld wird daraufhin eine 3 eingetragen und der Hintergrund wird „rot“ dargestellt.

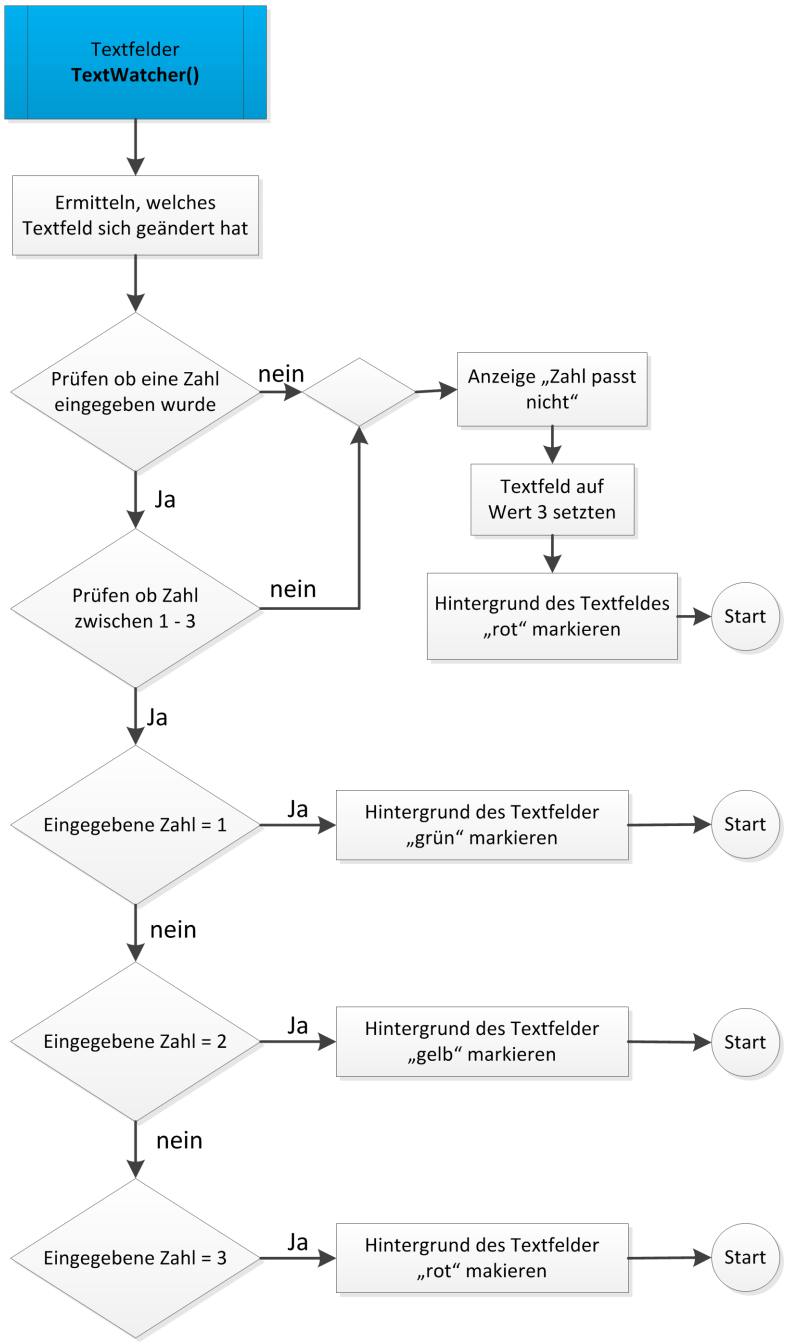

Abb. 29: Schematischer Ablauf der *Activity Hinzufügen TextWatcher()*

Sollte es sich bei dem Eintrag um einen Wert zwischen 1 - 3 handeln, wird geprüft, welche Zahl eingetragen wurde, woraufhin die eingegebene Zahl in dem Textfeld dargestellt und der Hintergrund für den Eintrag angepasst wird.

7.11 Activity Bearbeiten

In dieser Activity können bestehende Einträge der Datenbank nachträglich bearbeitet werden. Bei dem Aufruf des Fensters wird zu Beginn geprüft, ob eine Datenbank existiert. Wenn beim Aufruf dieser Activity keine Datenbank vorhanden ist, wird ein Textfeld generiert, dass den Nutzer darüber informiert. Durch die Bestätigung des Textes, wird zur Startseite gewechselt. Wenn Datenbankeinträge bestehen, werden diese ermittelt und in das Drop-Down-Menü eingetragen. Bei der Auswahl einer App aus dem Drop-Down-Menü, werden dann die eingestellten Parameter geladen und in den entsprechenden Feldern dargestellt. Diese geladenen Einträge können daraufhin von den Nutzern bearbeitet bzw. angepasst werden. Nachdem Einstellungen geändert wurden, können diese über den Button „speichern“ in der Datenbank abgespeichert werden. Es besteht in dieser Activity ebenfalls die Möglichkeit Einträge aus der Datenbank zu löschen. Über den Button „löschen“ wird der gewählte Eintrag aus dem Drop-Down-Menü mit den Einstellungen aus der Datenbank gelöscht. Durch den Button „zurück“ wird zur Startseite gewechselt.

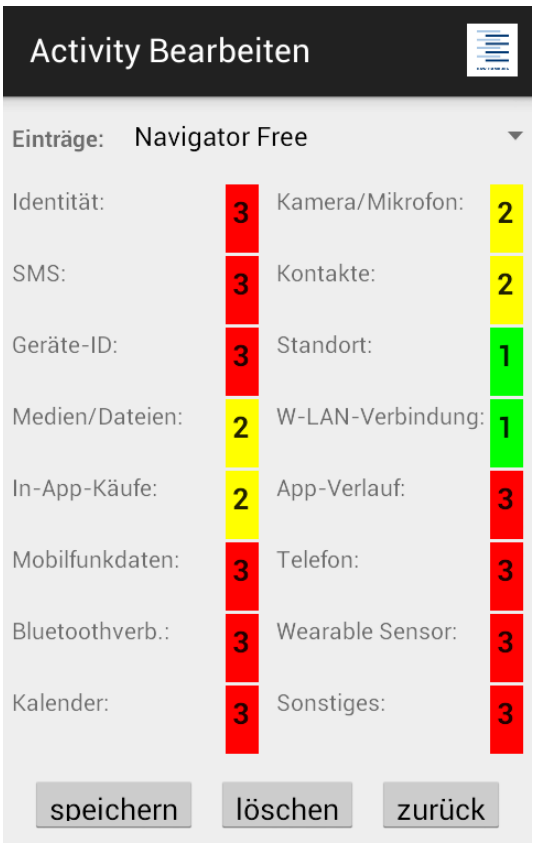

Abb. 30: *Activity Bearbeiten*

7.11.1 Schematische Darstellung der Activity Bearbeiten

In der Abbildung 31 auf Seite 75 ist der Ablauf der *Activity Bearbeiten* dargestellt. Beim Start dieser Activity werden die Events *onClickListener()*, *onItemSelected()* und *TextWatcher()* initialisiert. Des Weiteren wird überprüft, ob eine Datenbank vorhanden ist. Wenn keine Datenbank existiert, wird dieses durch eine Meldung angezeigt. Mit Bestätigung dieser Meldung wird *Activity Startseite* aufgerufen. Wenn eine Datenbank vorhanden ist, werden die Einträge ermittelt und in das Drop-Down-Menü, siehe Abbildung 30 eingetragen. Es wird geprüft, ob die Activity von der *Activity Hinzufügen* aufgerufen wurde. Bei dem Aufruf durch die Activity wird der Eintrag im Drop-Down-Menü auf den empfangenden Eintrag gesetzt, woraufhin die Bewertungen der Berechtigungsgruppen aus der Datenbank geladen und in den Textfeldern dargestellt werden.

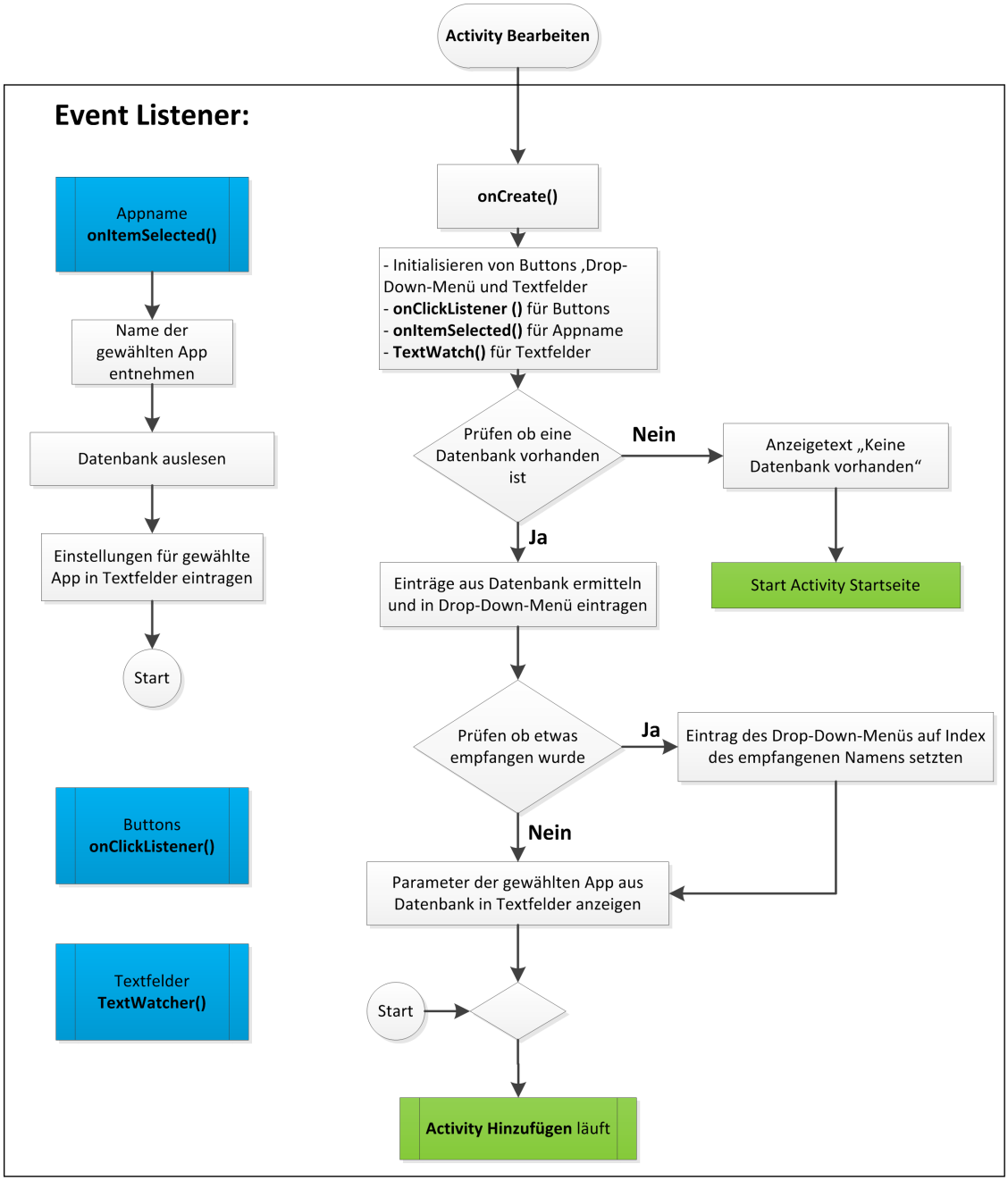

Abb. 31: Schematischer Ablauf der *Activity Bearbeiten*

Wenn einer der Buttons aus Abbildung 30 gedrückt wird, wird die Methode *onClickListener()* aufgerufen und es wird geprüft, welcher Button betätigt wurde. Mit dem Button „zurück“ wird zur Startseite gewechselt. Bei Betätigung des Button „löschen“ wird der App-Name aus dem Drop-Down-Menü ermittelt und der Eintrag in der Datenbank gesucht. Daraufhin wird der Eintrag mit den dazugehörigen Einstellungen gelöscht. Die Datenbank wird neu abgespeichert und die Activity wird neu gestartet, damit die Änderungen auch in dem Drop-Down-Menü sichtbar sind.

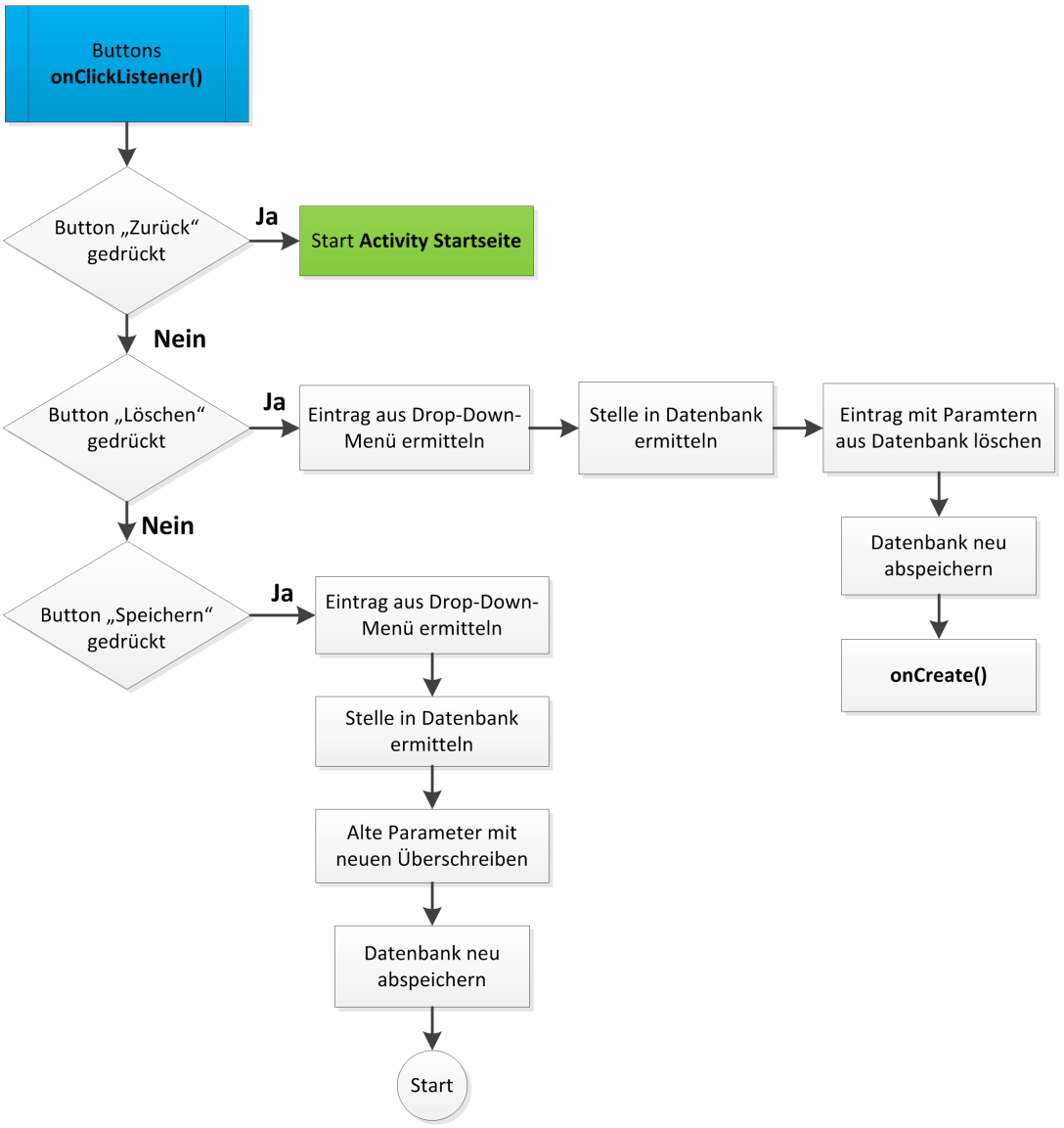

Abb. 32: Schematischer Ablauf der *Activity Bearbeiten onClickListener()*

Wenn für eine App aus der Datenbank die Bewertungen der jeweiligen Berechtigungsgruppen vorgenommen wurde, können diese gespeichert werden, was über den Button „speichern“ geschieht. Es wird hierbei der Eintrag aus dem Drop-Down-Menü ermittelt und die Stelle in der Datenbank gesucht. Die bestehenden Werte in der Datenbank werden durch die neuen Werte überschrieben, woraufhin diese abgespeichert wird.

Der Ablauf der Textfelder in der *Activity Bearbeiten* ist identisch mit den Textfeldern aus der *Activity Hinzufügen*, siehe Abbildung 29 Seite 73. Nach einer Änderung in einem Textfeld wird geprüft, welches sich geändert hat, woraufhin untersucht wird, ob eine Zahl zwischen 1 - 3 eingegeben wurde. Wurde keine Zahl zwischen 1 - 3 eingetragen, wird ein Text angezeigt, der die Nutzer darüber informiert und es wird der Eintrag mit einer 3 überschrieben sowie die Hintergrundfarbe „rot“ dargestellt.

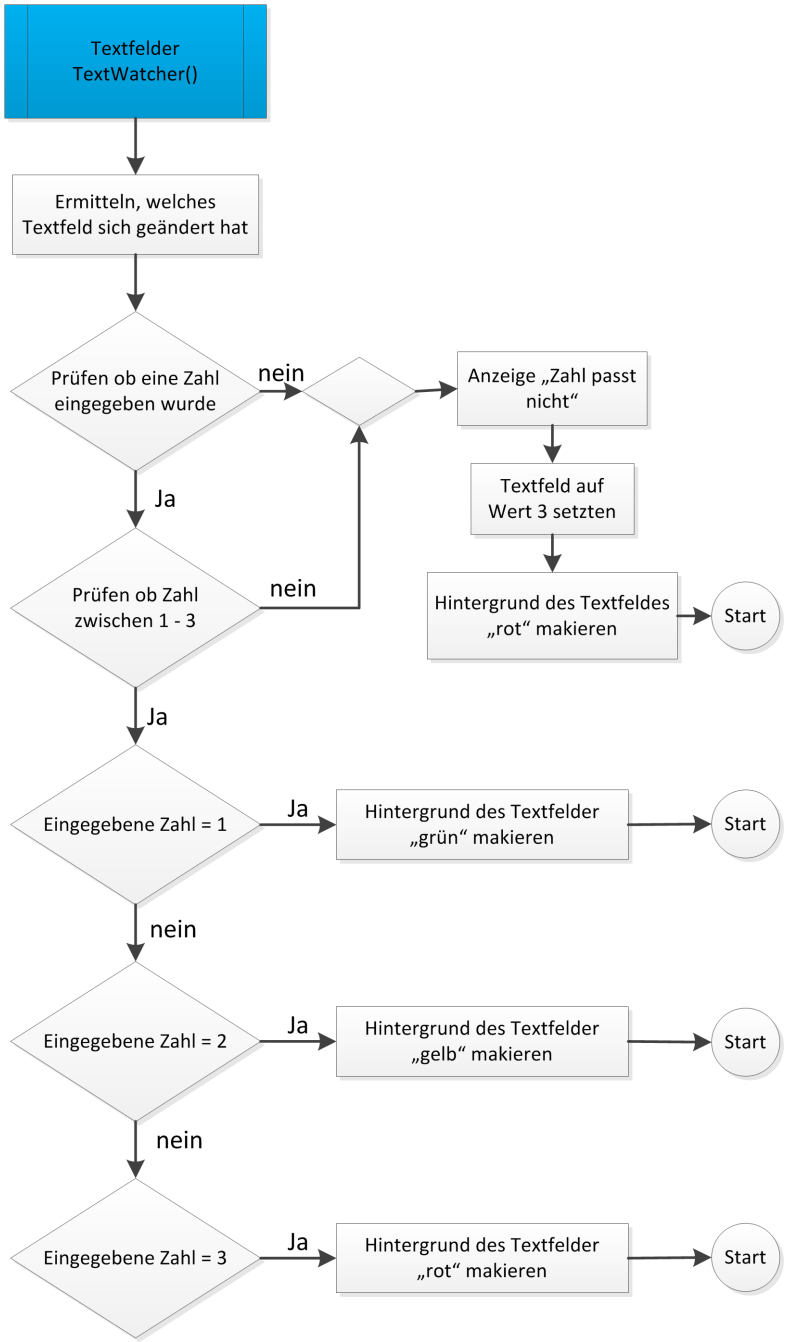

Abb. 33: Schematischer Ablauf der *Activity Bearbeiten TextWatcher()*

Wird eine Zahl zwischen 1 - 3 eingegeben, wird untersucht, um welche Zahl es sich handelt, woraufhin die Zahl und die passende Hintergrundfarbe dargestellt wird.

7.12 Activity Prüfen

In der *Activity Prüfen* können installierte Apps mit den Einträgen aus der Datenbank verglichen werden. Zudem besteht die Möglichkeit, sich die Rechte, welche im *Android-Manifest* einer App eingetragen sind, anzeigen zu lassen. Bei dem Start dieser Activity werden alle installierten Anwendungen, die sich auf dem Mobilgerät befinden, ermittelt und in dem Drop-Down-Menü dargestellt.

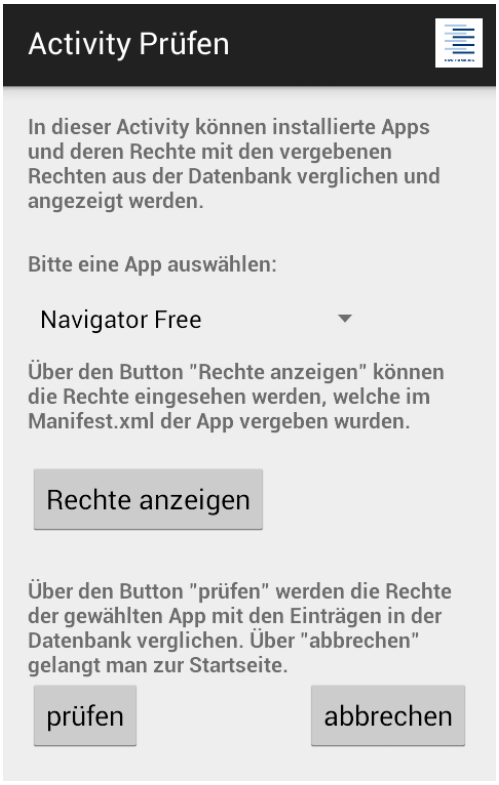

Abb. 34: *Activity Prüfen*

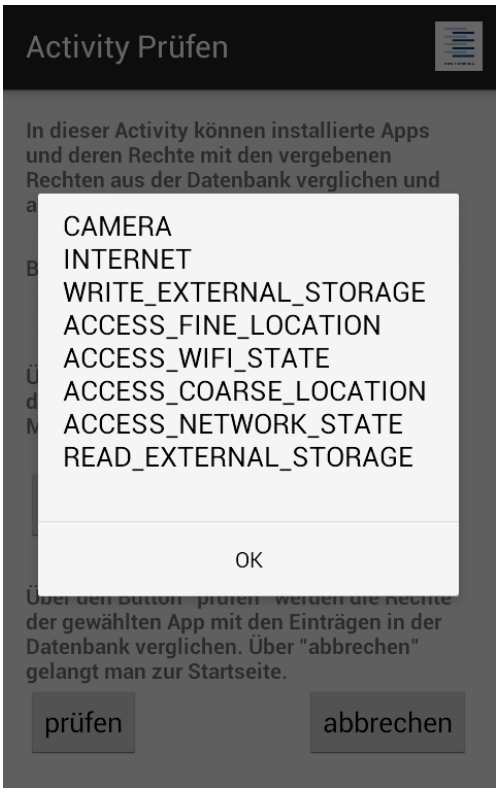

Abb. 35: Anzeige der Rechte

Über das Drop-Down-Menü aus Abbildung 34 können die installierten Apps, welche auf dem Gerät installiert sind, ausgewählt werden. Mit dem Button „Rechte anzeigen“ können sich die Rechte aus dem *Android-Manifest* der ausgewählten Anwendung angesehen werden. Dieses ist in Abbildung 35 dargestellt. Um sich die Rechte aus dem *Android-Manifest* anzeigen zu lassen wird kein Datenbankeintrag benötigt. Bei dieser Anzeige handelt es sich um die Rechte einer App, die noch nicht in die Berechtigungsgruppen unterteilt wurden. Mit dem „Klicken“ auf „OK“ wird das Fenster aus Abbildung 35 wieder geschlossen.

Über den Button „prüfen“ können die vergebenen Rechte aus dem *Android-Manifest* der Anwendung mit den Einstellungen aus der Datenbank verglichen werden.

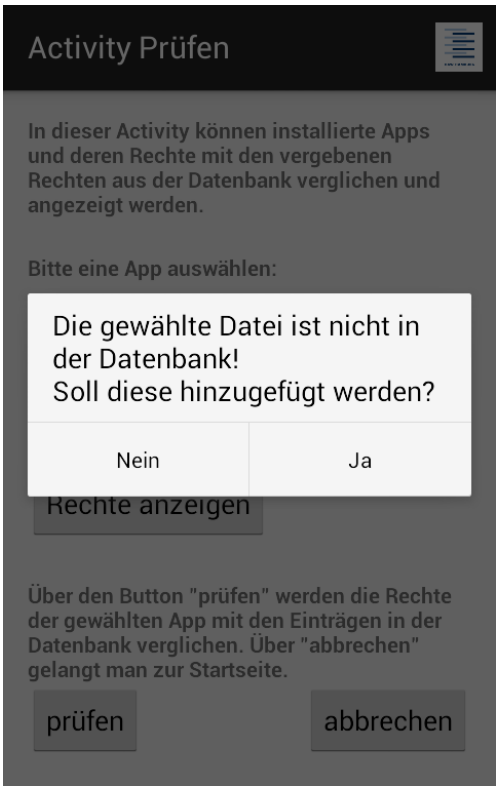

Abb. 36: App nicht in Datenbank

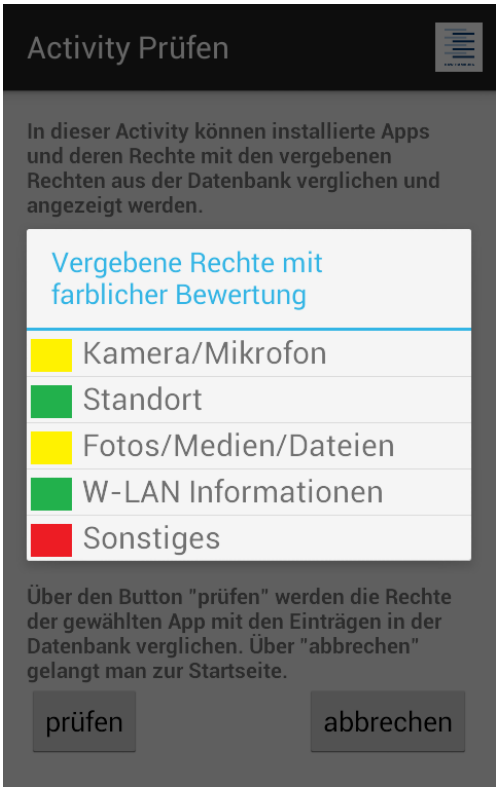

Abb. 37: Darstellung der Rechte

Wenn eine Anwendung zum Prüfen ausgewählt wird und noch keine Datenbank vorhanden oder die Anwendung nicht in der Datenbank eingetragen ist, wird ein Fenster, siehe Abb. 36, mit dem Informationstext dargestellt. Über dieses Fenster kann angegeben werden, ob die gewählte App in die Datenbank aufgenommen werden soll. Bei Bestätigen mit „Ja“ wird die ausgewählte App an die *Activity Hinzufügen* übergeben, in der diese in die Datenbank aufgenommen werden kann. Wenn eine App, die sich in der Datenbank befindet, geprüft werden soll, werden zunächst die Rechte aus dem *Android-Manifest* ausgelesen, die daraufhin in die Berechtigungsgruppen der „einfachen Berechtigung“ von *Google* eingetragen werden. Dieser Vorgang muss erfolgen, damit die Berechtigungsgruppen der installierten App mit dem Eintrag der Datenbank verglichen werden kann. Nach dem Vergleichen werden die benötigten Berechtigungsgruppen mit der Bewertung aus der Datenbank dargestellt. Dieses ist in Abbildung 37 zu sehen.

Ein direkter Vergleich zwischen den Einträgen in der Datenbank, den Rechten aus dem *Android-Manifest* und der farblichen Darstellung, welche in Abbildung 37 zu sehen ist, ist auf Seite 82 unter dem Abschnitt **Direkter Vergleich des Datenbankeintrags und der farblichen Darstellung der Berechtigungsgruppen** noch einmal besser dargestellt.

7.12.1 Schematischer Ablauf der Activity Prüfen

In der nachfolgenden Abbildung 38 ist der Ablauf der *Activity Prüfen* dargestellt. In dieser Activity wird nur das Event *onClickListener()* für die drei Buttons, siehe Abbildung 34 Seite 78, benötigt. Beim Start dieser Activity werden alle Apps, welche auf dem Mobilgerät installiert sind, ermittelt und in dem Drop-Down-Menü dargestellt.

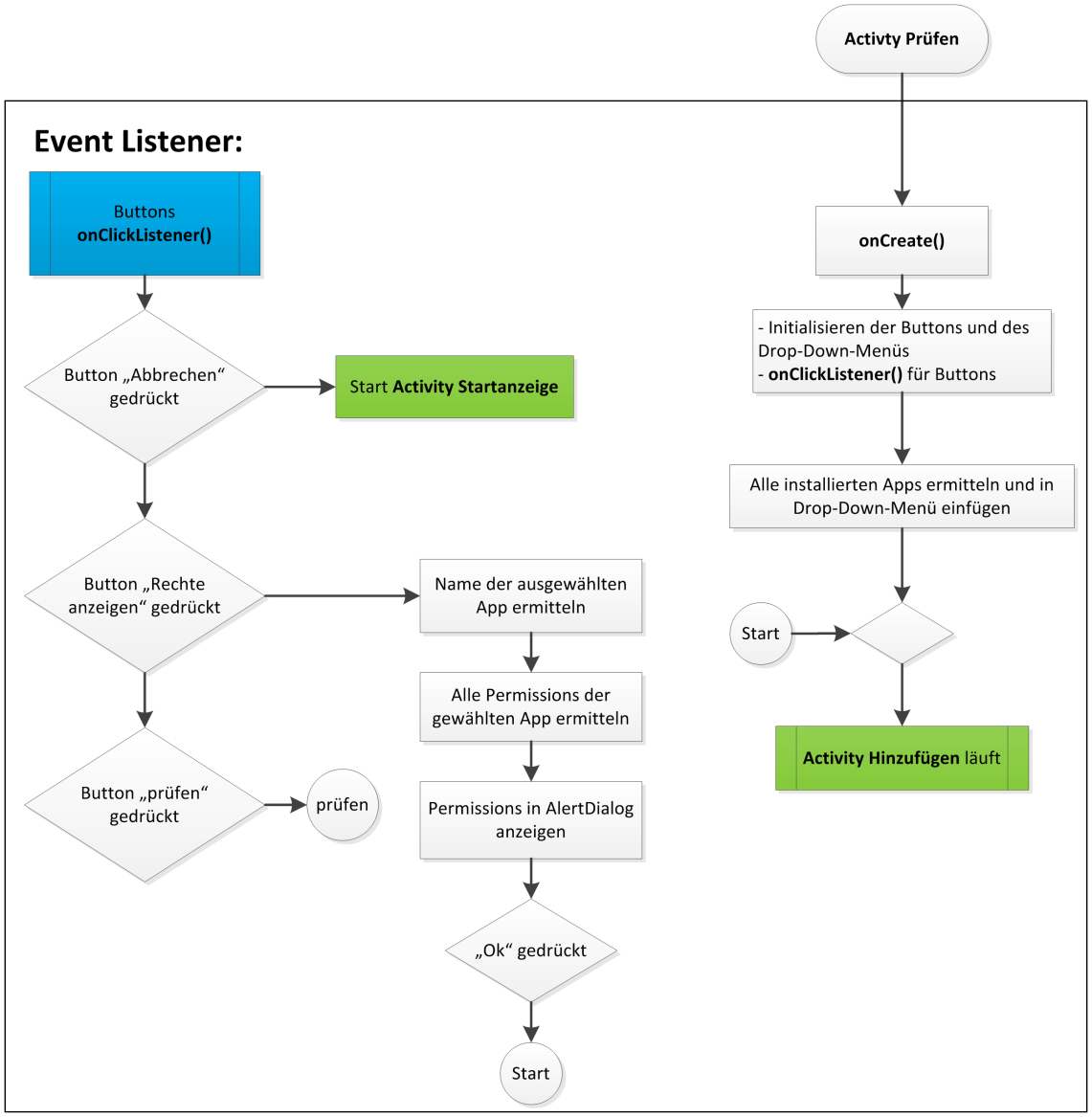

Abb. 38: Schematischer Ablauf der *Activity Prüfen*

Beim Drücken auf den Button „abbrechen“ wird zur Startseite gewechselt. Durch Betätigung des Buttons „Rechte anzeigen“ wird der App-Name aus dem Drop-Down-Menü entnommen und es werden für diese App die Rechte aus dem *Android-Manifests* ausgelesen und in einem sogenannten AlertDialog angezeigt (siehe Abbildung 35 Seite 78).

Durch Betätigung des Buttons „prüfen“, wird zunächst der gewählte Eintrag aus dem Drop-Down-Menü ermittelt, woraufhin untersucht wird, ob eine Datenbank vorhanden ist. Sollte keine Datenbank vorhanden sein, kann diese mit der gewählten App angelegt werden. Hierzu wird die *Activity Hinzufügen* aufgerufen. Wenn die Datenbank vorhanden ist aber es keinen Eintrag mit der gewählte App gibt, kann diese in die Datenbank ebenfalls über *Activity Hinzufügen* aufgenommen werden.

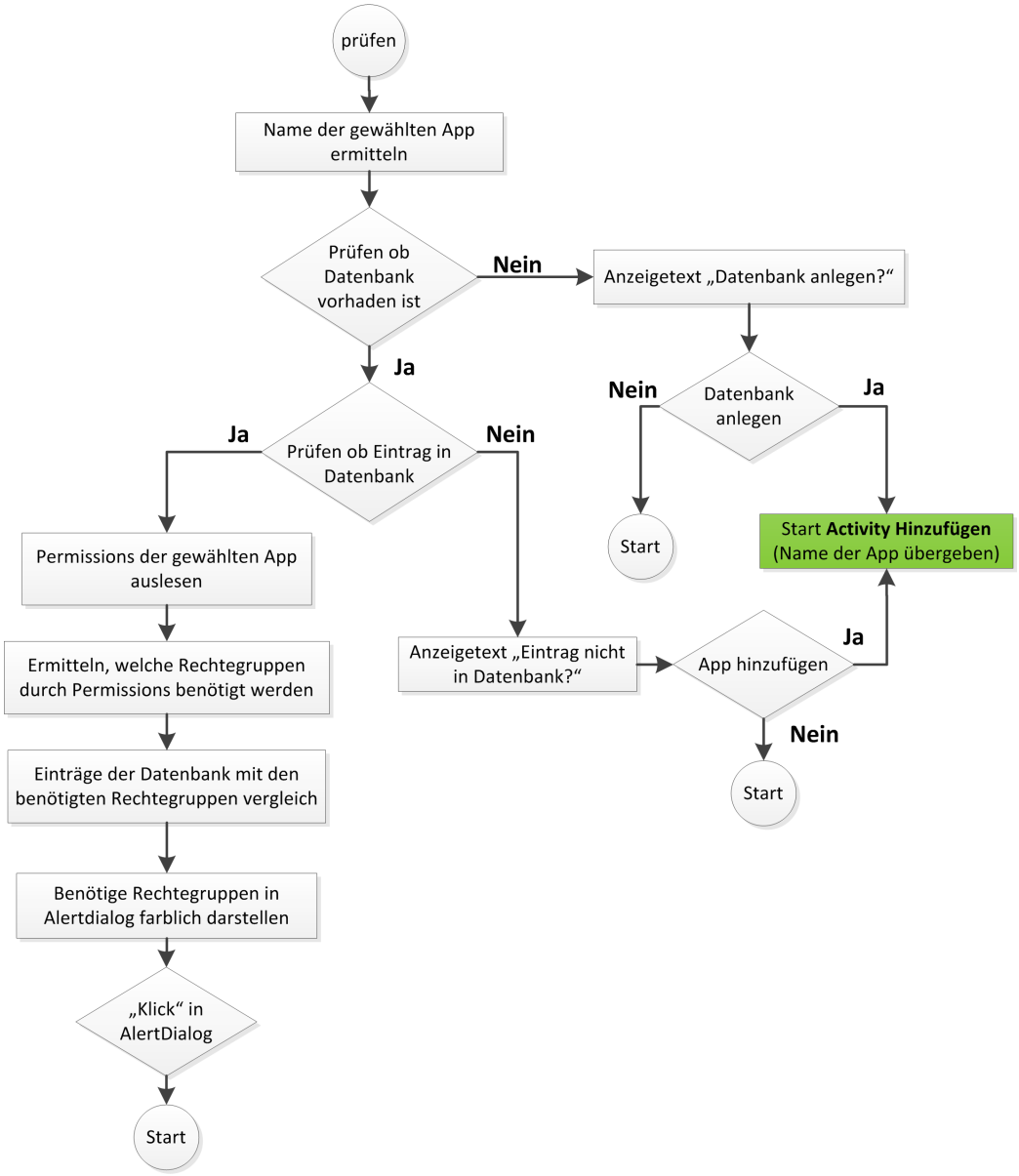

Abb. 39: Schematischer Ablauf der *Activity Prüfen*

Besteht ein Eintrag mit der gewählten App in der Datenbank, werden alle Rechte aus dem *Android-Manifest* dieser App ausgelesen. Mit diesen Rechten wird dann geprüft, welche Berechtigungsgruppen für die Anwendung benötigt werden. Daraufhin werden die ermittelten Berechtigungsgruppen der App mit den Einträgen der Datenbank verglichen. Danach werden die Berechtigungsgruppen der App, wie in der Datenbank angelegt, mit dem Ampel-System dargestellt (siehe Abbildung 37).

7.13 Direkter Vergleich des Datenbankeintrags und der farblichen Darstellung der Berechtigungsgruppen

Im Vorfeld wurden die einzelnen Activities der entwickelten Anwendung dargestellt. In diesem Abschnitt wird noch einmal der direkte Zusammenhang zwischen dem Eintag der Datenbank, den Rechten aus dem *Android-Manifest* und der farblichen Darstellung der Berechtigungsgruppen gezeigt.

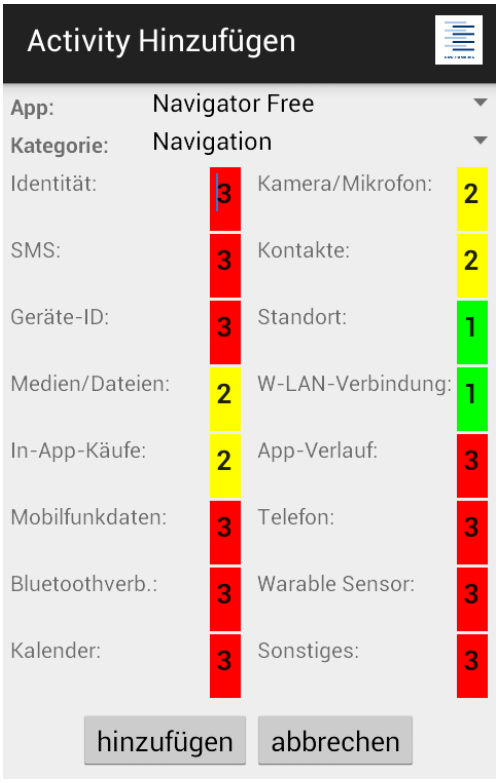

Abb. 40: Eintrag in der Datenbank

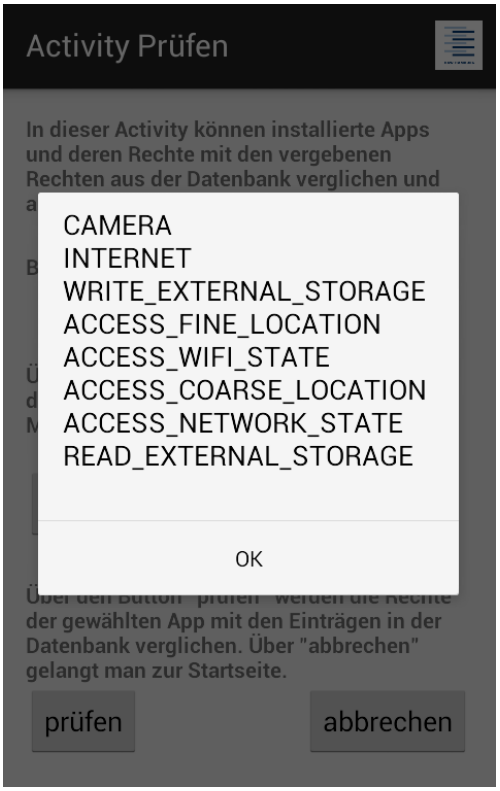

Abb. 41: *Android-Manifest* Rechte

Wie in Abbildung 40 zu erkennen ist, wurde beim Hinzufügen einer Anwendung in die Datenbank jede Berechtigungsgruppe bewertet und somit eingetragen, ob die jeweilige Berechtigungsgruppe für die Anwendung notwendig, bedingt notwendig oder überhaupt nicht notwendig ist. Wenn eine Anwendung in die Datenbank aufgenommen wird, muss jede dieser Gruppen bewertet werden, da noch nicht bekannt ist, welche Rechte im *Android-Manifest* vergeben wurden. In Abbildung 41 sind die Rechte dargestellt, welche im *Android-Manifest* der Navigations-App eingetragen sind. Um diese Rechte mit dem Eintrag der Datenbank Vergleich zu können, muss geprüft werden, in welchen Berechtigungsgruppen der „einfachen Berechtigung“ von *Google* diese Rechte eingetragen sind.

Für die Navigations-App ergeben sich aus den ermittelten Rechten folgende Berechtigungsgruppen:

- **Kamera/Mikrofon** CAMERA
- **Fotos/Medien/Dateien** WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE
- **W-LAN-Verbindungsinformationen:** ACCESS_WIFI_STATE
- **Standort:** ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION
- **Sonstiges:** INTERNET, ACCESS_NETWORK_STATE

Durch diese Einteilung der Rechte aus dem *Android-Manifest* in die Berechtigungsgruppen der „einfachen Berechtigung“ kann ein Vergleich mit dem Datenbankeintrag gemacht werden.

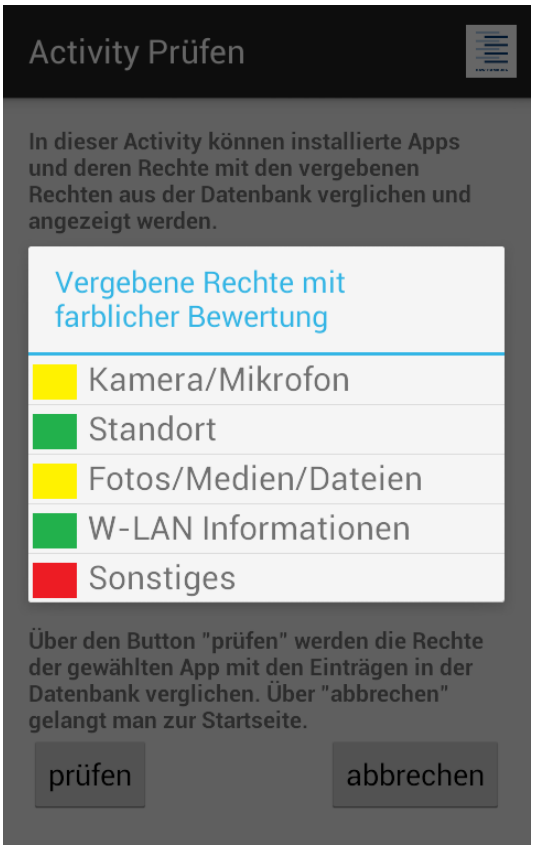

Abb. 42: Darstellung der Berechtigungsgruppen mit Ampel-System

Durch das Auslesen des *Android-Manifest* der zu prüfenden App konnte ermittelt werden, welche Berechtigungsgruppen diese App benötigt. Somit kann für diese Berechtigungsgruppen geprüft werden, was für diese App in der Datenbank eingetragen ist. Die Einträge der Datenbank für die Berechtigungsgruppen sind in Abb. 40 Seite 82 dargestellt. Es ist zu entnehmen, dass für die Berechtigungsgruppen Standort und W-LAN-Verbindungsinformationen eine 1 eingetragen wurde. Diese beiden Gruppen werden von der App auch wirklich benötigt und deshalb in Abb. 42 mit „grün“ dargestellt. Bei den Berechtigungsgruppen Kamera/Mikrofon sowie Fotos/Medien/Dateien wurde eine 2 in die Datenbank eingetragen, somit werden diese beiden Gruppen mit „gelb“ markiert. Eine 3 wurde bei der Gruppe Sonstiges in der Datenbank vergeben. Diese wird somit in Abb. 42 mit „rot“ dargestellt. Alle anderen Berechtigungsgruppen, welche in der Datenbank bewertet wurde, aber nicht von der Anwendung benötigt werden, werden nicht mit angezeigt.

8 Schlussbetrachtung

In diesem letzten Kapitel soll der Aufbau dieses Dokuments noch einmal zusammengefasst werden. Zudem soll erwähnt werden, wie das Ergebnis dieser Bachelorthesis mit Blick in die Zukunft noch verbessert werden kann. Abschließend erfolgt die eigene Meinung zu dieser Arbeit.

8.1 Zusammenfassung

Um in dieser Bachelorthesis die Implementierungsmöglichkeiten sicherheitstechnischer Funktionen mobiler Geräte und deren Betriebssysteme zu untersuchen, musste sich zu Beginn ein Überblick über die am meist vertretenden Systeme verschafft werden. Bei der Untersuchung der Betriebssysteme und deren Marktanteile wurde festgestellt, dass das *Android*-System die größte Marktpräsenz auf dem Sektor der mobilen Betriebssysteme aufweist. Um die verschiedenen Systeme besser zu verstehen, wurden die am stärksten am Markt vertretenen Betriebssysteme in Aufbau und Funktionsweise untersucht. Hierbei wurde festgestellt, dass Anwendungen in einer Sandbox ausgeführt werden, wodurch erreicht wird, dass unterschiedliche Applikationen und deren Speicherbereich von einander getrennt sind. Zudem wird damit sichergestellt, dass, wenn Fehler in einer Anwendung auftreten, dieses keine Auswirkung auf andere Anwendungen haben. Daraufhin wurden die Betriebssysteme bezüglich ihrer integrierten Sicherheitsfunktion untersucht und verglichen. Hierbei konnte festgestellt werden, dass die Systeme bereits Sicherheitsfunktionen beinhalten, um die Daten von Nutzern zu schützen. Durch diese Funktionen können Mobilgeräte beispielsweise mit einem Passwort geschützt werden. Außerdem werden von den Systemen die Benutzerdaten auf dem Gerät und bei der Übertragung verschlüsselt. Es besteht ebenfalls die Möglichkeit für den Nutzer, bei Verlust des mobilen Geräts, die Daten aus der Ferne zu löschen. Um zu überprüfen, wie Schadsoftware auf den Systemen installiert wird, wurde untersucht, welche Angriffe durch Schadprogramme es bereits gegen die jeweiligen Systeme gegeben hat. Bei der Untersuchung, wie diese Schadsoftware auf den jeweiligen Systeme installiert wird, wurde festgestellt, dass in der Regel die Installationen von den Nutzern selbständig durchgeführt werden. Der Grund dafür ist, dass die Schadsoftware meistens unter einem vertrauenswürdigen Decknamen in den Stores oder auf Internetseiten zum Download angeboten wird. Des Weiteren werden Berechtigungen, die von Benutzern für eine Anwendung bestätigt werden müssen, einfach akzeptiert oder nicht verstanden. Durch den hohen Marktanteil von *Android* und dadurch, dass sich die meiste Schadsoftware gegen das System richtet, wurde sich dafür entschieden, für dieses System eine Anwendung zu entwickeln, die die Nutzer bei der Erkennung von Schadprogrammen unterstützt. Um die Berechtigungsgruppen besser bewerten zu können, werden durch die entwickelte Anwendung, die Gruppen einer installierten *Android*-App in Form eines Ampel-Systems dargestellt. Mit dieser entwickelten Anwendung besteht für die Nutzer die Möglichkeit, eine installierte App nachträglich auf die Berechtigungsgruppen und deren Notwendigkeit zu prüfen. Hierfür muss für die zu prüfende App ein Eintrag in einer Datenbank erfolgen, indem die Rechtegruppen in Form des Ampel-Systems bewertet werden. Hierbei wird festgelegt, welche Gruppen für die installierte App wichtig für die eigentliche Funktion, welche bedingt wichtig und welche überhaupt nicht notwendig sind. Um diese Bewertung den Nutzern zu erleichtern, wurden für unterschiedliche Anwendungen Kategorien angelegt. Durch die Auswahl dieser Kategorien werden für eine App Voreinstellungen geladen, welche nachträglich von den Nutzern bearbeitet werden können. Durch die erstellte Anwendung haben die Nutzer die Möglichkeit, sich die Rechte aus dem *Android-Manifest* der

installierten App anzeigen zu lassen. Für das Prüfen einer App werden die Einträge aus dem *Android-Manifest* ermittelt und in die Rechtekategorien eingeteilt, welche ebenfalls im *Play Store* angezeigt werden. Durch den Abgleich der Berechtigungsgruppen der installierten App sowie des Datenbankeintrag werden die vergebenen Gruppen der App in einem Ampel-System farblich dargestellt. Dadurch können die Nutzer die Berechtigungen einer App besser bewerten. Danach ist es den Nutzern selbst überlassen, ob sie die geprüfte Anwendung auf ihrem Mobilgerät installiert lassen oder diese entfernen.

8.2 Ausblick

Die in dieser Bachelorthesis entstandene Applikation bietet den Nutzern, mit Hilfe des Ampel-Systems, eine bessere Einschätzung, ob eine installierte App bei der Installation ausreichend oder zu viele Berechtigungen verlangt hat. Um Anwendungen auf geforderte Rechtegruppen zu überprüfen, müssen die Apps installiert sein. In Form einer Weiterentwicklung könnte das aktuelle Programm so geändert werden, dass die zu prüfenden Apps nicht installiert werden müssen, sondern das hierbei das *Android-Manifest* aus der vorhandenen .apk-Datei ausgelesen werden kann. Dieses war in der Bearbeitungszeit dieser Thesis nicht realisierbar, da das Auslesen des *Android-Manifests* einer .apk-Datei einen beachtlichen Aufwand mit sich bringt. Der Grund dafür ist, dass die .apk-Datei bereits kompiliert ist. Außerdem besteht die Möglichkeit für Downloads von .apk-Dateien aus dem *Play Store* nicht. Diese Dateien werden in einem Ordner auf dem mobilen Gerät gespeichert, für den man ohne root-Zugriff keine Berechtigung hat. Somit könnte zunächst nur das Prüfen von .apk-Dateien aus Quellen von Drittanbietern geschehen. Da die entwickelte Anwendung bislang nur ein farbliche Darstellung der unterschiedlichen Berechtigungsgruppen bietet, könnte ein weiterer Punkt für eine Weiterentwicklung sein, dass installierten Apps nachträglich Berechtigungen entzogen oder diese blockiert werden können. Für das Bearbeiten von Anwendungen und derer Berechtigungen wird ebenfalls der root-Zugriff benötigt. Dadurch ist das Bearbeiten aktuell nicht möglich. Die entwickelte Anwendung könnte auch dahingegen weiter Entwickelte werden, indem die Anwendung bereits eine bestehende Datenbank mit verschiedenen Apps beinhaltet, da somit die Nutzer für Apps keinen Datenbankeintrag mehr vornehmen müssten. Des Weiteren müssen, damit mit der entwickelten Anwendung jede App geprüft werden kann, alle weiteren Rechte, welche im *Android-Manifest* vergeben werden können, in das Programm eingepflegt werden.

8.3 Eigene Meinung

Abschließend zu dieser Thesis möchte ich meine persönlichen Eindrücke schildern.

Zu Beginn dieser Bachelorthesis war mir noch nicht bekannt, in welche Richtung sich diese Arbeit bewegt. Nach den ersten Recherchen über die Verbreitung der unterschiedlichen mobilen Betriebssysteme auf dem Markt, wurde festgestellt, dass das *Android*-System den Markt beherrscht. Bei der Untersuchung des Aufbaus und der Funktionsweise der am Markt am stärksten vertretenen Systeme wurde erkannt, dass es über die freien Systeme, wie *Android* und *Symbian*, sehr gute Informationen gibt. Die Informationsbeschaffung der Systeme *Apple iOS*, *Windows Phone* und *Blackberry OS* war deutlich schwieriger. Die Hersteller dieser drei Systeme versuchen den Aufbau und deren Funktionsweise geheim zu halten. Dieses ist für die mobilen Betriebssysteme von Vorteil, da z.B. Hacker dadurch wenig Informationen über den Systemaufbau bekommen. Meiner Meinung nach legt die Firma *Blackberry* von den untersuchen Systemen am meisten Wert auf die Geheimhal-

tung des Systemaufbaus. Es sind über das System hauptsächlich Informationen über das Sicherheitskonzept zu finden aber wenig über das Betriebssystem selbst.
Nachdem die Informationen über die Funktionsweisen zusammen getragen wurden, wurde der Vergleich der Systeme vorgenommen. Hierbei wurde festgestellt, dass die Systeme Anwendungen in einer Sandbox ausführen, womit schon eine beachtliche Sicherheit geboten wird. Zudem wurde untersucht, welche Sicherheitsfunktionen die jeweiligen Betriebssysteme integriert haben. Dadurch wurde erkannt, dass jedes Betriebssystem, welches untersucht wurde, gute Sicherheitsfunktionen bereits besitzen. Bei der Überprüfung, wie Schadsoftware auf den Geräten installiert wird, wurde festgestellt, dass die Nutzer durch Unwissenheit oder blindem Vertrauen die Schadprogramme selbständig installieren, wodurch der Entschluss gefasst wurde, eine Anwendung zu entwickelt, die den Nutzern die geforderten Berechtigungen von Anwendungen verdeutlicht. Am Anfang der Entwicklung dieser Anwendung habe ich mich schwer damit getan, Eclipse und den Emulator korrekt einzurichten. Nachdem dieses jedoch geschehen war, konnte ich feststellen, dass der Aufbau der Oberfläche ähnlich wie bei Delphi ist, welches ich von der Arbeit her kenne. Dadurch konnte ich relativ schnell gute Ergebnisse mit der Entwicklung erzielen.
Ich denke, dass das entwickelte Programm den Nutzern sehr gut deutlich macht, welche Rechte bei einer Anwendung erforderlich sind und welche überhaupt nicht. Mit der Weiterentwicklung dieser Anwendung kann ich mir durchaus vorstellen, dass diese App von *Android*-Nutzern zum Überprüfen von installierten Anwendungen gerne genutzt wird.

Abbildungsverzeichnis

Tabellenverzeichnis

Stichwortverzeichnis

Abkürzungsverzeichnis

AES	**A**dvanced **E**ncryption **S**tandard
API	**A**pplication **P**rogramming **I**nterface
App	**App**likation
ASLR	**A**ddress **S**pace **L**ayout **R**andomization
BES	**B**lacberry **E**nterprise **S**erver
CLR	**C**ommon **L**anguage **R**untime
DLL	**D**ynamic **L**ink **L**ibrary
DNS	**D**omain **N**ame **S**erver
DoS	**D**enial **o**f **S**ervice
DVM	**D**alvik **V**irtual **M**achine
EAS	**E**xchange **A**ctive**S**ync
GUI	**G**raphical **U**ser **I**nterface
HTTP	**H**ypertext **T**ransfer **P**rotocol
HTTPS	**H**ypertext **T**ransfer **P**rotocol **S**ecure
IANA	**I**nternet **A**ssigned **N**umbers **A**uthority
IMEI	**I**nternational **M**obile **E**quipment **I**dentity
IP	**I**nternet**p**rotokoll
JAR	**J**ava **Ar**chive
LAN	**L**ocal **A**rea **N**etwork
ME	**M**icro **E**dition
MIDP	**M**obile **I**nformation **D**evice **P**rofile
MMS	**M**ultimedia **M**essaging **S**ervice
MMU	**M**emory **M**anagement **U**nit
NOC	**N**etwork **O**perating **C**enter
OS	**O**perating **S**ystem
PIM	**P**ersonal **M**essenger **I**nformation
RAM	**R**andom **A**ccess **M**emory
RIM	**R**esearch **i**n **M**otion
ROM	**R**ead **O**nly **M**emory
RTOS	**R**eal **T**ime **O**perating **S**ystem
SD	**S**ecure **D**igital **M**emory
SMS	**S**hort **M**essage **S**ervice
SSH	**S**ecure **S**hell
TAN	**T**rans**a**ktions**n**ummer
VM	**V**irtual **M**achine
VPN	**V**irtual **P**rivate **N**etwork
WLAN	**W**ireless **L**ocal **A**rea **N**etzwork

Literaturverzeichnis

[1] Android-User.de: *Neue Play Store Berechtigungsgruppen schaffen Hintertür für böswillige Apps.* http://www.android-user.de/neue-play-store-berechtigungsgruppen-schaffen-hintertuer-fuer-boeswillige-apps/, 10.6.2014. – abgerufen am 14.1.2015, 14:09 Uhr.

[2] Android-User.de: *Was ist eine Firmware?* www.android-user.de/was-ist-eine-firmware/, 26.1.2012. – abgerufen am 10.2.2015, 7:24 Uhr.

[3] Android.com: *App Manifest.* http://developer.android.com/guide/topics/manifest/manifest-intro.html, o.J.. – abgerufen am 5.2.2015, 13:11 Uhr.

[4] Apfelwiki.de: *Mac OS X.* http://www.apfelwiki.de/Main/MacOSX, 1.12.2014. – abgerufen am 26.1.2015, 11:19 Uhr.

[5] Beck, A. ; Pant, M.: *Android - Grundlagen und Programmierung, 1. Auflage.* dpunkt.verlag, 2009.

[6] Borges, G. ; Schwenk, J. ; Stuckenberg, C.-F. ; Wegener, C.: *Identitätsdiebstahl und Identitätsmissbrauch im Internet.* Springer-Verlag, 2011.

[7] boxcryptor.com: *AES-256 Verschlüsselung.* https://www.boxcryptor.com/de/verschl%C3%BCsselung, o.J.. – abgerufen am 25.1.2015, 14:34 Uhr.

[8] Chan Yuk Ching Yuko, Lo Tsz Nga A. Ho Wing Man Anissa A. Ho Wing Man Anissa: *M-Commerce Application.* http://eportal.cityu.edu.hk/bbcswebdav/users/ycychan2/lab4_part3.pdf, o.J.. – abgerufen am 29.10.2014, 8:44 Uhr.

[9] Chip.de: *Was ist Malware.* http://praxistipps.chip.de/was-ist-malware_28542, 15.3.2014. – abgerufen am 26.1.2015, 11:37 Uhr.

[10] Chip.de: *Jailbreak – was ist das?* http://praxistipps.chip.de/jailbreak-was-ist-das_12321, 18.6.2014. – abgerufen am 26.1.2015, 14:38 Uhr.

[11] ComputerBase.de: *Android: Marktanteil in Deutschland auf über 50%.* http://www.computerbase.de/2012-11/android-marktanteil-in-deutschland-auf-ueber-50-prozent/, 05.11.2012. – abgerufen am 16.10.2014, 11:58 Uhr.

[12] Computerbetrug.de: *Viren, Trojaner, Würmer, Dos-Angriffe.* http://www.computerbetrug.de/sicherheit-im-internet/viren-trojaner-wurmer, o.J.. – abgerufen am 22.2.2015, 11:16 Uhr.

[13] Computerwoche.de: *Samsung KNOX - Sicherheit für Android.* http://www.computerwoche.de/a/samsung-knox-sicherheit-fuer-android,2552641, 9.1.2015. – abgerufen am 12.1.2015, 10:45 Uhr.

[14] DroidWiki.de: *Dalvik VM.* http://www.droidwiki.de/Dalvik_VM, 8.12.2014. – abgerufen am 26.1.2015, 9:37 Uhr.

[15] Dummies.com: *Enterprise Mobile Device Security: Personal Symbian Device Protection.* www.dummies.com/how-to/content/enterprise-mobile-device-security-personal-symbian.html, o.J.. – abgerufen am 10.2.2015, 8:00 Uhr.

[16] DUMMIES.COM: *Enterprise Mobile Device Security: Personal Symbian Device Protection*. http://www.dummies.com/how-to/content/enterprise-mobile-device-security-personal-symbian.html, o.J.. – abgerufen am 28.10.2014, 12:15 Uhr.

[17] ELEKTRONIK-KOMPENDIUM: *DoS - Denial of Service*. http://www.elektronik-kompendium.de/sites/net/1412091.htm, o.J.. – abgerufen am 22.2.2015, 10:00 Uhr.

[18] ELEKTRONIK-KOMPENDIUM.DE: *Cache (L1 / L2 / L3)*. http://www.elektronik-kompendium.de/sites/com/0309291.htm, o.J.. – abgerufen am 26.1.2015, 8:53 Uhr.

[19] ELEKTRONIK-KOMPENDIUM.DE: *VPN - Virtual Private Network*. http://www.elektronik-kompendium.de/sites/net/0512041.htm, o.J.. – abgerufen am 26.1.2015, 13:36 Uhr.

[20] ENZYKLO.DE: *Unixoides System*. http://www.enzyklo.de/Begriff/Unixoides%20System, o.J.. – abgerufen am 26.1.2015, 13:26 Uhr.

[21] EXCELSIORUSA.COM: *Schützen Sie Ihren Java-Code — durch Obfuscatoren und weitergehende Maßnahmen*. http://www.excelsior-usa.com/de/articles/java-obfuscatoren.html, 9.7.2015. – abgerufen am 26.1.2015, 9:01 Uhr.

[22] FSCKLOG.COM: *ikee: iPhone-Wurm attackiert jailbroken iPhones per SSH mit Rick Astley*. http://www.fscklog.com/2009/11/ikee-iphone-wurm-attackiert-jailbroken-iphones-per-ssh-mit-rick-astley.html, 9.11.2009. – abgerufen am 21.2.2015, 18:28 Uhr.

[23] FUTUREZONE.AT: *Bada*. http://futurezone.at/produkte/sms-kann-windows-phone-mango-lahmlegen/24.574.295, 13.12.11. – abgerufen am 17.11.2014, 7:55 Uhr.

[24] GISO.DE: *Prozessverwaltung*. http://rowa.giso.de/german/process.html, o.J.. – abgerufen am 26.1.2015, 12:41 Uhr.

[25] GOLEM.DE: *Android-Apps erhalten leichter mehr Berechtigungen*. http://www.golem.de/news/google-play-store-android-apps-erhalten-leichter-mehr-berechtigungen-1406-106856.html, 2.6.2014. – abgerufen am 12.1.2015, 10:45 Uhr.

[26] GOOGLE.COM: *App-Berechtigung prüfen*. https://support.google.com/googleplay/answer/6014972?hl=de, 2015. – abgerufen am 9.1.2015, 15:39 Uhr.

[27] GRUENDERSZENE.DE: *Application-Programming-Interface (API)*. http://www.gruenderszene.de/lexikon/begriffe/application-programming-interface-api, o.J.. – abgerufen am 26.1.2015, 8:25 Uhr.

[28] HANDELSBLATT.COM: *Google*. http://www.handelsblatt.com/themen/Google, o.J.. – abgerufen am 16.2.2015, 16:53 Uhr.

[29] HEISE.DE: *Android-Smartphones per Drive-by infiziert*. http://www.heise.de/security/meldung/Android-Smartphones-per-Drive-by-infiziert-1446758.html, 2.3.2012. – abgerufen am 2.2.2015, 11:09 Uhr.

[30] Inside-Handy: *Android laut Marktforschern weiter auf dem Vormarsch.* http://www.inside-handy.de/news/19173-android-laut-marktforschern-weiter-auf-dem-vormarsch, 10.9.2010. – abgerufen am 16.2.2015, 14:14 Uhr.

[31] IT-Wissen: *Smartphone-Betriebssystem.* http://www.itwissen.info/definition/lexikon/Smartphone-Betriebssystem-smartphone-operating-system.html, 10.01.2015. – abgerufen am 10.1.2015, 13:54 Uhr.

[32] ITWissen.de: *Smartphone Betriebssysteme.* http://www.itwissen.info/definition/lexikon/Smartphone-Betriebssystem-smartphone-operating-system.html, o.J.. – abgerufen am 26.1.2015, 14:00 Uhr.

[33] ITWissen.de: *Speichermanager.* http://www.itwissen.info/definition/lexikon/Speicherverwaltungseinheit-MMU-memory-management-unit.html, o.J.. – abgerufen am 26.1.2015, 11:49 Uhr.

[34] ITWissen.de: *SSL (secure socket layer).* http://www.itwissen.info/definition/lexikon/secure-socket-layer-SSL-SSL-Protokoll.html, o.J.. – abgerufen am 26.1.2015, 13:12 Uhr.

[35] ITWissen.de: *Thread.* www.itwissen.info/definition/lexikon/Thread-thread.html, o.J.. – abgerufen am 17.2.2015, 08:21 Uhr.

[36] ITWissen.info: *Bytecode.* http://www.itwissen.info/definition/lexikon/Bytecode-bytecode.html, o.J.. – abgerufen am 26.1.2015, 8:43 Uhr.

[37] ITWissen.info: *JVM (Java virtual machine).* www.itwissen.info/definition/lexikon/Java-virtual-machine-JVM-Java-virtuelle-Maschine.html, o.J.. – abgerufen am 19.2.2015, 8:54 Uhr.

[38] ITWissen.info: *Middleware.* www.itwissen.info/definition/lexikon/Middleware-middleware.html, o.J.. – abgerufen am 12.2.2015, 8:19 Uhr.

[39] ITWissen.info: *Mobilgerät.* http://www.itwissen.info/definition/lexikon/Mobilgeraet-mobile-equipment.html, o.J.. – abgerufen am 22.2.2015, 15:00 Uhr.

[40] Java.com: *Was ist J2ME oder Java ME?* https://www.java.com/de/download/faq/whatis_j2me.xml, o.J.. – abgerufen am 26.1.2015, 10:49 Uhr.

[41] Kaspersky.de: *Was ist eine Man-in-the-Middle-Attacke?* http://blog.kaspersky.de/was-ist-eine-man-in-the-middle-attacke/905/, 10.4.2013. – abgerufen am 2.2.2015, 12:33 Uhr.

[42] MAC & I: *WireLurker: Malware schleicht sich vom Mac auf iOS-Geräte.* http://www.heise.de/mac-and-i/meldung/WireLurker-Malware-schleicht-sich-vom-Mac-auf-iOS-Geraete-2443208.html, 6.11.2014. – abgerufen am 7.11.2014, 10:11 Uhr.

[43] madeyourweb.com: *Was sind Frameworks?* www.madeyourweb.com/webentwicklung/was-sind-frameworks.html, o.J.. – abgerufen am 17.2.2015, 10:03 Uhr.

[44] Merelewis.com: *Was sind Buffer Overflows?* www.merelewis.com/was-sind-buffer-overflows/, 9.6.2011. – abgerufen am 18.2.2015, 9:44 Uhr.

[45] Microsoft.com: *Die Entwicklung von Windows*. windows.microsoft.com/de-de/windows/history#T1=era0, 11.2013. – abgerufen am 19.2.2015, 11:31 Uhr.

[46] Microsoft.com: *Exchange ActiveSync*. https://technet.microsoft.com/de-de/library/aa998357%28v=exchg.150%29.aspx, 6.11.2014. – abgerufen am 26.1.2015, 10:29 Uhr.

[47] Müller, S.: *Application Provisioning im Rahmen des Seminars Mobile Java*. Westfälische Wilhelms-Universität Münster, o.J.

[48] MOBISTACK: *Denial of Service (DoS) Angriffe über den BlackBerry BrowserDenial of Service (DoS) Angriffe über den BlackBerry Browser*. http://mobistack.com/denial-of-service-dos-angriffe-uber-den-blackberry-browser/, 15.1.2011. – abgerufen am 11.11.2014, 9:58 Uhr.

[49] Monitor: *Angriff der Handy-Viren*. http://www.monitor.at/index.cfm/storyid/7285_Gefahr_fuer_Handys_und_PDAs-Angriff_der_Handy-Viren, 7.4.2005. – abgerufen am 11.11.2014, 12:00 Uhr.

[50] nakedsecurity: *First iPhone worm discovered - ikee changes wallpaper to Rick Astley photo*. https://nakedsecurity.sophos.com/2009/11/08/iphone-worm-discovered-wallpaper-rick-astley-photo/, 8.1.2009. – abgerufen am 13.2.2015, 10:10 Uhr.

[51] nethosting24.de: *BlackBerry Internet Service" und "BlackBerry Enterprise Server*. http://blog.nethosting24.de/blackberry/blackberry-internet-service.html, 24.6.2009. – abgerufen am 26.1.2015, 8:35 Uhr.

[52] Nokia.com: *Mail for Exchange*. http://i.nokia.com/blob/view/-/807284/data/5/-/Mail-for-Exchange-Datasheet-Symbian-Oct10-pdf.pdf, 2010. – abgerufen am 29.10.2014, 9:10 Uhr.

[53] Online, Spiegel: *Angriff per Bilddatei: Schwere Sicherheitslücke bei Blackberry-Servern*. http://www.spiegel.de/netzwelt/gadgets/angriff-per-bilddatei-schwere-sicherheitsluecke-bei-blackberry-servern-a-779865.html, 12.8.2011. – abgerufen am 11.11.2014, 9:44 Uhr.

[54] Oracle.com: *Programming the Blackberry with J2ME*. www.oracle.com/technetwork/java/index-139239.html, April 2005. – abgerufen am 10.2.2015, 11:47 Uhr.

[55] Paderborn, Universität: *ssh - Secure Shell*. https://www2.math.uni-paderborn.de/rechnerbetrieb/informationen-a-z/sicherheit/ssh-als-sichere-alternative.html, 7.1.2015. – abgerufen am 26.1.2015, 12:00 Uhr.

[56] PDF.io: *Strukturen und Datentypen*. http://www.pdfio.net/k-23403716.html#download_area, o.J.. – abgerufen am 21.10.2014, 9:36 Uhr.

[57] QNX.de: *QNX Neutrino RTOS*. www.qnx.de/products/neutrino-rtos/neutrino-rtos.html?lang=de, o.J.. – abgerufen am 10.2.2015, 10:16 Uhr.

[58] SamsungKnox.com: *KNOX Workspace*. https://www.samsungknox.com/de/products/knox-workspace/technical, o.J. – abgerufen am 12.1.2015, 10:17 Uhr.

[59] SEARCHDATACENTER.DE: *Dynamic Link Library (DLL).* http://www.searchdatacenter.de/definition/Dynamic-Link-Library-DLL, November 2014. – abgerufen am 26.1.2015, 10:12 Uhr.

[60] SECURE bbcom: *Zahlreiche weitere Angriffvektoren bedrohen die mobile Kommunikation.* https://www.bbcomsecure.de/Angriffswege/Die-wichtigsten-Angriffswege-auf-Mobiltelefone.html, o.J.. – abgerufen am 12.11.2014, 11:10 Uhr.

[61] SECURITY heise: *Zeus-Trojaner verstärkt Angriffe auf mTANs.* http://www.heise.de/security/meldung/Zeus-Trojaner-verstaerkt-Angriffe-auf-mTANs-1661773.html, 7.8.2012. – abgerufen am 11.11.2014, 9:47 Uhr.

[62] SOFTONIC: *Unsignierte Symbian-Software online signieren.* http://artikel.softonic.de/unsignierte-symbian-software-online-signieren, 7.7.2008. – abgerufen am 28.10.2014, 11:23 Uhr.

[63] SOLUTIONS, Proteus: *Wissenswert: Marktanteile mobiler Betriebssysteme weltweit.* http://www.google.de/imgres?imgurl=https%3A%2F%2Fwww.proteus-solutions.de%2F_system-Pics%2FNewsPics%2FProteus-Statistik-Marktanteile-mobile-OS-weltweit-2010.png&imgrefurl=https%3A%2F%2Fwww.proteus-solutions.de%2F~Unternehmen%2FNews-PermaLink%3AtM.F04!sM.NI41!Article.953275.asp&h=380&w=500&tbnid=VpdrNnnspp_j0M%3A&zoom=1&docid=ZR_bCmGDoBl-jM&ei=MZKuVPSsI9fYavTLgagK&tbm=isch&iact=rc&uact=3&dur=379&page=1&start=0&ndsp=33&ved=0CCUQrQMwAQ, 24.11.11. – abgerufen am 8.1.2015, 15:30 Uhr.

[64] STATIONAGENT.DE: *Smartphones - Blackberry (RIM) OS.* www.stationagent.de/smartphones-blackberry-rim-o-2112, o.J.. – abgerufen am 12.2.2015, 8:39 Uhr.

[65] STATISTA.COM: *Marktanteil von Symbian am Endkundenabsatz von Smartphones weltweit vom 1. Quartal 2009 bis zum 3. Quartal 2013.* http://de.statista.com/statistik/daten/studie/246459/umfrage/marktanteil-von-symbian-am-weltweiten-smartphone-absatz-nach-quartalen/, o.J.. – abgerufen am 16.2.2015, 11:41 Uhr.

[66] STATISTA.COM: *Marktanteile der Betriebssysteme an der Smartphone-Nutzung in Deutschland von Dezember 2011 bis Juni 2014.* http://de.statista.com/statistik/daten/studie/170408/umfrage/marktanteile-der-betriebssysteme-fuer-smartphones-in-deutschland/, o.J.. – abgerufen am 16.2.2015, 14:15 Uhr.

[67] STATISTA.COM: *Marktanteile der mobilen Betriebssysteme am Absatz von Smartphones in Deutschland von Juni bis August in den Jahren 2013 und 2014.* http://de.statista.com/statistik/daten/studie/198435/umfrage/marktanteile-der-smartphone-betriebssysteme-am-absatz-in-deutschland/, o.J.. – abgerufen am 15.10.2014, 8:13 Uhr.

[68] STATISTA.COM: *Prognose zu den Marktanteilen der Betriebssysteme am Absatz vom Smartphones weltweit in den Jahren 2014 und 2018.* http://de.statista.com/statistik/daten/studie/182363/umfrage/prognostizierte-marktanteile-bei-smartphone-betriebssystemen/, o.J.. – abgerufen am 15.10.2014, 8:08 Uhr.

[69] Szwillus, G.: *Vorlesung Grundlagen der Programmierung 2.* 2007

[70] T-Online.de: *Neuer Schadcode: Alte Sicherheitslücke in Android wird zum Problem.* http://community.t-online.de/community/forum/digital/Smartphones-und-Tablet-PC/neuer-schadcode:3A:-alte-sicherheitsl:C3::BC:cke-in-android-wird-zum-problem,76270627.html, 21.2.2014. – abgerufen am 13.11.2014, 12:36 Uhr.

[71] tecchannel.de: *Android ist Ziel von 97 Prozent der mobilen Malware.* http://www.tecchannel.de/kommunikation/news/2053774/android_ist_ziel_von_97_prozent_mobiler_malware/, 5.03.2014. – abgerufen am 17.11.2014, 13:03 Uhr.

[72] techfacts.de: *Android und iPhone Virenschutz.* http://www.techfacts.de/ratgeber/android-und-iphone-virenschutz, 16.05.2014. – abgerufen am 18.11.2014, 9:34 Uhr.

[73] techstage.de: *Android.* www.techstage.de/thema/Android, o.J.. – abgerufen am 10.2.2015, 7:09 Uhr.

[74] teltarif.de: *Eingestellte Eigenentwicklung: Samsung Bada.* http://www.teltarif.de/handy/betriebssysteme/bada.html, o.J.. – abgerufen am 26.1.2015, 8:17 Uhr.

[75] teltarif.de: *Handy-Sicherheit: So schützen Sie sich vor Malware and Datenlecks.* http://www.teltarif.de/handy/sicherheit.html, o.J.. – abgerufen am 17.11.2014, 8:52 Uhr.

[76] Test.de: *Die Betriebssysteme im Überblick.* https://www.test.de/Handys-und-Smartphones-im-Test-4222793-4222876/, 10.12.2014. – abgerufen am 15.10.2014, 7:41 Uhr.

[77] Teufl, Dipl.-Ing. P. ; Dietrich, Dipl.-Ing. K.: *Sicherheitsanalyse - Blackberry OS 5, Version 1.0.2,19. April 2010.* Zentrum für sichere Informationstechnologie – Austria, 2010.

[78] Thomas, E. ; Zeitz, J.: *Anwendungsentwicklung für Symbian OS.* 2006.

[79] Trojaner-Info.de: *Trojanische Pferde - Was ist das ?* http://www.trojaner-info.de/beschreibung.shtml, o.J.. – abgerufen am 22.2.2015, 11:22 Uhr.

[80] tuxanddroid.de: *Über den Sinn und Unsinn von Antiviren Apps für Android.* http://tuxanddroid.de/android/uber-den-sinn-und-unsinn-von-anti-viren-apps-fur-android, 1.7.2013. – abgerufen am 17.11.2014, 10:52 Uhr.

[81] Uni-Protokolle.de: *Darwin (Betriebssystem).* http://www.uni-protokolle.de/Lexikon/Darwin_%28Betriebssystem%29.html, o.J.. – abgerufen am 26.1.2015, 10:01 Uhr.

[82] Urrigshardt, F.: *Softwarearchitekturen Windows Vista und Windows Phone 7.* Universität Siegen, 2011

[83] Wall, P.: *Die Architektur von Mac OS X und iOS.* 1.7.2011.

[84] Welt, PC: *Die Sicherheit von Windows Phone 7*. http://www.pcwelt.de/ratgeber/Die-Sicherheit-von-Windows-Phone-7-1129284.html, 2014. – abgerufen am 18.11.2014, 13:39 Uhr.

[85] Welt, PC: *Schwachstellen in Windows Mobile*. http://www.pcwelt.de/news/Schwachstellen-in-Windows-Mobile-259431.html, 31.01.2007. – abgerufen am 11.11.2014, 13:28 Uhr.

[86] Wiesel, R.: *Mobile OS Security Architectures*. Technische Universität Berlin, 2010.

[87] Winfwiki.de: *Das Sicherheitskonzept von Blackberry-Endnutzergeräten in Hinblick auf sicheren E-Mail-Datenverkehr*. http://winfwiki.wi-fom.de/index.php/Das_Sicherheitskonzept_von_Blackberry-Endnutzerger%C3%A4ten_in_Hinblick_auf_sicheren_E-Mail-Datenverkehr#Betrachtung_der_Blackberry-Architektur, 30.1.2011. – abgerufen am 21.10.2014, 16:38 Uhr.

[88] ZDnet.de: *Android-Architektur: Wieviel Linux steckt in Googles OS?* www.zdnet.de/41553061/android-architektur-wieviel-linux-steckt-in-googles-os/, 18.5.2011. – abgerufen am 10.2.2015, 7:16 Uhr.

[89] ZDNet.de: *Praxis: Mac-Malware WireLurker erkennen und beseitigen*. http://www.zdnet.de/88210420/mac-malware-wirelurker-erkennen-und-beseitigen/, 7.11.2014. – abgerufen am 21.2.2015, 18:51 Uhr.

Anhang

Der Anhang dieser Bachelorthesis befindet sich auf DVD, einzusehen bei den Prüfern Prof.Dr.-Ing. R. Fitz oder Prof.Dr. H. Neumann.

Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 26. Februar 2015	
Ort, Datum	Unterschrift